TURBOMOLE

Program Package for *ab initio*

Electronic Structure Calculations

USER'S MANUAL

TURBOMOLE Version 7.8
October 12, 2023

Contents

Chapter 1

Preface and General Information

1.1 Contributions and Acknowledgements

TURBOMOLE [1] is a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH 1989-2007, TURBOMOLE GmbH, since 2007. The following people have made contributions:

Reinhart Ahlrichs, Josefine Andersen, Markus Klaus Armbruster, Rafał A. Bachorz, Hilke Bahmann, Alexander Baldes, Michael Bär, Hans–Peter Baron, Rüdiger Bauernschmitt, Martin Becker, Florian A. Bischoff, Stephan Böcker, Florian Bruder, Asbjörn M. Burow, Guo P. Chen, Sonia Coriani, Nathan Crawford, Peter Deglmann, Fabio Della Sala, Michael Diedenhofen, Sebastian Ehlert, Michael Ehrig, Karin Eichkorn, Simon Elliott, Daniil Fedotov, Yannick J. Franzke, Daniel Friese, Filipp Furche, James Furness, Sree Ganesh Balasubramani, Sebastian Gillhuber, Tino Gimon, Andreas Glöß, Nora Graf, Lukáš Grajciar, Robin Grotjahn, Frank Haase, Marco Häser, Christof Hättig, Arnim Hellweg, Benjamin Helmich, Sebastian Höfener, Christof Holzer, Hans Horn, Christian Huber, Waldemar Hujo, Uwe Huniar, Aaron D. Kaplan, Marco Kattannek, Max Kehry, Sascha Klawohn, Wim Klopper, Andreas Köhn, Christoph Kölmel, Markus Kollwitz, Katharina Krause, Michael Kühn, Roman Łazarski, Fabian Mack, Toni M. Maier, Klaus May, Nils Middendorf, Carolin Müller, Paola Nava, Christian Neiß, Christian Ochsenfeld, Holger Öhm, Mathias Pabst, Shane M. Parker, Holger Patzelt, Ansgar Pausch, Patrik Pollak, Dmitrij Rappoport, Kevin Reiter, Saswata Roy, Oliver Rubner, Ansgar Schäfer, Gunnar Schmitz, Uwe Schneider, Tobias Schwabe, Manas Sharma, Marek Sierka, Jianwei Sun, David P. Tew, Oliver Treutler, Barbara Unterreiner, Malte von Arnim, Vamsee K. Voora, Florian Weigend, Patrick Weis, Horst Weiss, Nina Winter, Jason M. Yu

We acknowledge help from

- Michael Dolg, University of Stuttgart, now: University of Cologne
- Jürgen Gauss, University of Mainz
- Christoph van Wüllen, University of Bochum, now: TU Kaiserslautern
- Stefan Grimme, University of Bonn
- Stefan Brode, BASF AG, Ludwigshafen
- Heinz Schiffer, HOECHST AG, Frankfurt
- the research groups of Ove Christiansen, Aarhus University, and Jacob Kongsted, University of Southern Denmark

and financial support by the University of Karlsruhe, BASF AG, BAYER AG, HOECHST AG, the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG), Carl-Zeiss-Stiftung, and Fonds der Chemischen Industrie.

Contact address:

Turbomole GmbH
Litzenhardtstrasse 19
76135 Karlsruhe
Germany
E-mail: support@turbomole.com
Web: https://www.turbomole.org
Support E-mail: support@turbomole.com

1.2 Features of TURBOMOLE

TURBOMOLE has been specially designed for UNIX workstations and PCs and efficiently exploits the capabilities of this type of hardware. TURBOMOLE consists of a series of modules; their use is facilitated by various tools.

Outstanding features of TURBOMOLE are

- semi-direct algorithms with adjustable main memory and disk space requirements
- full use of all point groups
- efficient integral evaluation
- stable and accurate grids for numerical integration
- low memory and disk space requirements

1.3 How to Quote Usage of TURBOMOLE

Please quote the usage of the program package under consideration of the version number:

> TURBOMOLE V7.8 2023, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007,
> TURBOMOLE GmbH, since 2007; available from
> https://www.turbomole.org.

A LaTeX template could look like this:

```
@misc{TURBOMOLE,
title = {{TURBOMOLE V7.8 2023}, a development of {University of Karlsruhe} and
         {Forschungszentrum Karlsruhe GmbH}, 1989-2007,
         {TURBOMOLE GmbH}, since 2007; available from \\
         {\tt https://www.turbomole.org}.}}
```

In addition, we kindly ask to cite the recent review of the TURBOMOLE project [2]

A LaTeX template for this review could look like this:

```
@Article{   Balasubramani.Chen.ea:TURBOMOLE.2020,
  author = {Balasubramani, Sree Ganesh  and Chen, Guo P.
             and Coriani, Sonia and Diedenhofen, Michael and
             Frank, Marius S. and Franzke, Yannick J. and
             Furche, Filipp and Grotjahn, Robin and Harding, Michael E.
             and H\"attig, Christof and Hellweg, Arnim and
             Helmich-Paris, Benjamin and Holzer, Christof and Huniar, Uwe
             and Kaupp, Martin and Marefat Khah, Alireza
             and Karbalaei Khani, Sarah and M\"uller, Thomas and Mack, Fabian
             and Nguyen, Brian D. and Parker, Shane M. and Perlt, Eva
             and Rappoport, Dmitrij and Reiter, Kevin and Roy, Saswata and
             R\"uckert, Matthias and Schmitz, Gunnar and Sierka, Marek
             and Tapavicza, Enrico and Tew, David P. and van W\"ullen, Christoph
             and Voora, Vamsee K. and Weigend, Florian and
             Wody{\'n}ski, Artur and Yu, Jason M.},
  title     = {TURBOMOLE: Modular program suite for \textit{ab initio}
               quantum-chemical and condensed-matter simulations},
  journal   = {J. Chem. Phys.},
  volume    = {152},
  issue     = {18},
  pages     = {184107},
  year      = {2020},
  url       = { https://doi.org/10.1063/5.0004635},
  DOI       = {10.1063/5.0004635}
}
```

Scientific publications require proper citation of methods and procedures employed. The output headers of `TURBOMOLE` modules include the relevant papers. One may also use the following connections between: method [module] number in the subsequent list (For module `ricc2` see also Section 10).

- Programs and methods
  - general program structure and features: I
  - HF-SCF [`dscf`, `ridft`]: II
  - DFT (quadrature) [`dscf`, `ridft`, `escf`, `aoforce`]: IV, VI (m grids), VII (a grids)
  - RI-DFT [`ridft`, `aoforce`, `escf`, `riper`]: V, VI, XXXIII (marij), X (`escf`), XXXIV (`aoforce`)
  - periodic DFT [`riper`]: XLVIII, L, LI, XLIX
  - MP2 [`mpgrad`]: III
  - RI-MP2 [`ricc2`]: energies and gradients XI, XXXIX, XII, and (static) polarizabilities XLV
  - PNO-MP2 [`pnoccsd`]: energies XLVI
  - stability analysis [`escf`]: VIII
  - electronic excitations with CIS, RPA, TD-DFT [`escf`]: IX, X, XXIX, XXXVII
  - excited state structures and properties with CIS, RPA, TD-DFT [`egrad`]: XXX, XXXVI, XXXVII
  - RI-CC2 [`ricc2`]:
    - singlet XXII and triplet excitation energies XXIII
    - transition moments and first-order properties of excited states XXV and first-order properties for triplet states XXIV
    - ground state geometry optimizations XXXI
    - excited state geometry optimizations and relaxed properties XXXII
    - parallelization XXXIX
    - spin-component scaled (SCS) variants XLI
    - frequency-dependent and static polarizabilities XLV
  - RI-ADC(2), RI-CIS(D) and RI-CIS(D_∞) [`ricc2`]: XXXVIII
  - SOS variants of MP2, CIS(D), CIS(D_∞), ADC(2) and CC2 with $\mathcal{O}(\mathcal{N}^4)$-scaling XLII
  - analytical second derivatives (force fields) [`aoforce`]: XXVI, XXVII
  - RI-JK [`ridft`]: XXVIII
  - NMR chemical shifts [`mpshift`]: XIII, XIV, XV, XVI, XVII (HF, DFT) XVIII, XIX (MP2)
  - parallel DFT [`ridft`]: XX
  - geometry optimization in redundant internal coordinates [`relax`]: XXI
  - RI integral evaluation: XXXV

- explicitly correlated F12 methods for ground state energies [`ccsdf12` and `pnoccsd`]: MP2-F12 XLIII, PNO-MP2-F12 XLVII, MP3-F12 XLIV, MP4(F12*) XLIV, CCSD(F12) XL, CCSD(F12*) XLIV, CCSD(F12)(T) XL, CCSD(F12*)(T) XLIV
- Relativistic approaches: LII, LIII, LIV, LV, LVI ([`dscf`, `ridft`, etc.]), LVI, LIV,([`grad`, `rdgrad`, etc.]), LVII, LVIII, LIX LX ([`mpshift`])
- Local hybrid calculations: LXI, LXVIII ([`ridft`]), LXII, LXVI ([`grad`, `rdgrad`]), LXIII, LXIV, LXV, LXVI ([`escf`]), LXVII ([`egrad`]),
- Seminumerical and pseudospectral methods: LIII, LIV ([`ridft`, `rdgrad`]), LXV ([`escf`, `egrad`, `aoforce`])

- Orbital and auxiliary basis sets
  - basis sets:
    - SV, SV(P), SVP, DZ (a), TZV, TZVP, TZVPP (b), TZVPP(Rb-Hg) (f), QZV, QZVP, QZVPP (i)
    - new balanced basis sets (with smaller ECPs, i.e. the def2 basis sets): j
    - all-electron basis sets for Rb to Xe (SVPall, SVPPall, TZVPall, TZVPPall): g
    - references for the correlation consistent basis sets (cc-pVXZ, etc.) can be found e.g. at
      http://tyr0.chem.wsu.edu/~kipeters/Pages/cc_append.html, or
      http://www.grant-hill.group.shef.ac.uk/ccrepo/, or
      http://www.emsl.pnl.gov/forms/basisform.html.
      Note, that most of the correlation consistent basis sets in the basis set exchange library of TURBOMOLE have been downloaded from the latter EMSL web site and therefore users are requested to include in addition to the original scientific reference an appropriate citation (see web site for details) in any publications resulting from the use of these basis sets. See [3] for the current version of the basis set exchange library and [4] for previous versions. The same applies to the polarization consistent (pc, pcseg, pcSseg, pcJ, pcH, pcX) and IGLO (IGLO-II, IGLO-III) basis sets.
    - property–optimized augmentations: def2-SVPD, def2-TZVPD, def2-TZVPPD, def2-QZVPD,def2-QZVPPD (m).
    - basis sets for Dirac–Fock ECPs, i.e. the dhf basis sets: q.
    - basis sets for relativistic all-electron approaches, i.e. the x2c-XVPall (X=S, TZ, QZ) basis sets: r, t and their extensions for NMR shielding constants s, t. Also partly available in decontractef form (-unc).
    - decontracted basis sets of Dyall and co-workers (dyall-vdz, dyall-vtz, dyall-vqz) for all elements except 3d from
      http://dirac.chem.sdu.dk/basisarchives/dyall/.
  - auxiliary basis sets for RI-DFT: c, d, e
  - auxiliary basis sets for RI-MP2: f, k, h (for Dunning basis sets)

Further references of papers not from the TURBOMOLE group are given in the bibliography. The following publications describe details of the methodology implemented in TURBOMOLE:

Methods

I. Electronic Structure Calculations on Workstation Computers: The Program System `TURBOMOLE`. R. Ahlrichs, M. Bär, M. Häser, H. Horn and C. Kölmel; Chem. Phys. Lett., **162**, 165 (1989).

II. Improvements on the Direct SCF Method. M. Häser and R. Ahlrichs; J. Comput. Chem., **10**, 104 (1989).

III. Semi-direct MP2 Gradient Evaluation on Workstation Computers: The MP-GRAD Program. F. Haase and R. Ahlrichs; J. Comp. Chem., **14**, 907 (1993).

IV. Efficient Molecular Numerical Integration Schemes.
O. Treutler and R. Ahlrichs; J. Chem. Phys., **102**, 346 (1995).

V. Auxiliary Basis Sets to Approximate Coulomb Potentials. K. Eichkorn, O. Treutler, H. Öhm, M. Häser and R. Ahlrichs; Chem. Phys. Lett., **242**, 652 (1995).

VI. Auxiliary basis sets for main row atoms and transition metals and their use to approximate Coulomb potentials. K. Eichkorn, F. Weigend, O. Treutler and R. Ahlrichs; Theor. Chem. Acc., **97**, 119 (1997).

VII. Error-consistent segmented contracted all-electron relativistic basis sets of double- and triple-zeta quality for NMR shielding constants.
Y. J. Franzke, R. Treß, T. M. Pazdera and F. Weigend; Phys. Chem. Chem. Phys., **21**, 16658–16664 (2019).

VIII. Stability Analysis for Solutions of the Closed Shell Kohn–Sham Equation. R. Bauernschmitt and R. Ahlrichs; J. Chem. Phys., **104**, 9047 (1996).

IX. Treatment of Electronic Excitations within the Adiabatic Approximation of Time Dependent Density Functional Theory.
R. Bauernschmitt and R. Ahlrichs; Chem. Phys. Lett., **256**, 454 (1996).

X. Calculation of excitation energies within time-dependent density functional theory using auxiliary basis set expansions. R. Bauernschmitt, M. Häser, O. Treutler and R. Ahlrichs; Chem. Phys. Lett., **264**, 573 (1997).

XI. RI-MP2: first derivatives and global consistency. F. Weigend and M. Häser; Theor. Chem. Acc., **97**, 331 (1997).

XII. RI-MP2: Optimized Auxiliary Basis Sets and Demonstration of Efficiency. F. Weigend, M. Häser, H. Patzelt and R. Ahlrichs; Chem. Phys. Lett., **294**, 143 (1998).

XIII. Direct computation of second-order SCF properties of large molecules on workstation computers with an application to large carbon clusters. M. Häser,

R. Ahlrichs, H. P. Baron, P. Weis and H. Horn; Theoret. Chim. Acta, **83**, 455 (1992).

XIV. Calculation of Magnetic Shielding Constants with meta-GGA Functionals Employing the Multipole-Accelerated Resolution of the Identity: Implementation and Assessment of Accuarcy and Efficiency. K. Reiter, F. Mack and F. Weigend; J.. Chem. Theory Comput., **14**, 191 (2018).

XV. Paramagnetic NMR Shielding Tensors and Ring Currents: Efficient Implementation and Application to Heavy Element Compounds. S. Gillhuber, Y. J. Franzke, and F. Weigend; J. Phys. Chem. A **125**, 9707 (2021).

XVI. Paramagnetic NMR Shielding Tensors Based on Scalar Exact Two-Component and Spin–Orbit Perturbation Theory. F. Bruder, Y. J. Franzke, and F. Weigend; J. Phys. Chem. A **126**, 5050 (2022).

XVII. Impact of the current density on paramagnetic NMR properties. Y. J. Franzke and C. Holzer; J.. Chem. Phys. **157**, 031102 (2022).

XVIII. A direct implementation of the GIAO-MBPT(2) method for calculating NMR chemical shifts. Application to the naphthalenium and anthracenium ions. M. Kollwitz and J. Gauss; Chem. Phys. Lett., **260**, 639 (1996).

XIX. Non-Abelian point group symmetry in direct second-order many-body perturbation theory calculations of NMR chemical shifts. M. Kollwitz, M. Häser and J. Gauss; J. Chem. Phys., **108**, 8295 (1998).

XX. Parallelization of Density Functional and RI-Coulomb Approximation in TURBOMOLE. M. v. Arnim and R. Ahlrichs; J. Comp. Chem., **19**, 1746 (1998).

XXI. Geometry optimization in generalized natural internal Coordinates. M. v. Arnim and R. Ahlrichs; J. Chem. Phys., **111**, 9183 (1999).

XXII. CC2 excitation energy calculations on large molecules using the resolution of the identity approximation. C. Hättig and F. Weigend; J. Chem. Phys., **113**, 5154 (2000).

XXIII. Implementation of RI-CC2 for triplet excitation energies with an application to *trans*-azobenzene. C. Hättig and K. Hald; Phys. Chem. Chem. Phys., **4** 2111 (2002).

XXIV. First-order properties for triplet excited states in the approximated Coupled Cluster model CC2 using an explicitly spin coupled basis. C. Hättig, A. Köhn and K. Hald; J. Chem. Phys., **116**, 5401 (2002) and Vir. J. Nano. Sci. Tech., **5** (2002).

XXV. Transition moments and excited-state first-order properties in the coupled-cluster model CC2 using the resolution-of-the-identity approximation.
C. Hättig and A. Köhn; J. Chem. Phys., **117**, 6939 (2002).

XXVI. An efficient implementation of second analytical derivatives for density functional methods. P. Deglmann, F. Furche and R. Ahlrichs; Chem. Phys. Lett., **362**, 511 (2002).

XXVII. Efficient characterization of stationary points on potential energy surfaces.
P. Deglmann and F. Furche; J. Chem. Phys., **117**, 9535 (2002).

XXVIII. A fully direct RI-HF algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency. F. Weigend; Phys. Chem. Chem. Phys., **4**, 4285 (2002).

XXIX. An improved method for density functional calculations of the frequency-dependent optical rotation.
S. Grimme, F. Furche and R. Ahlrichs; Chem. Phys. Lett., **361**,321 (2002).

XXX. Adiabatic time-dependent density functional methods for excited state properties. F. Furche and R. Ahlrichs; J. Chem. Phys. **117**, 7433 (2002), J. Chem. Phys., **121**, 12772 (2004) (E).

XXXI. Geometry optimizations with the coupled-cluster model CC2 using the resolution-of-the-identity approximation. C. Hättig; J. Chem. Phys., **118**, 7751, (2003).

XXXII. Analytic gradients for excited states in the coupled-cluster model CC2 employing the resolution-of-the-identity approximation. A. Köhn and C. Hättig; J. Chem. Phys., **119**, 5021, (2003).

XXXIII. Fast evaluation of the Coulomb potential for electron densities using multipole accelerated resolution of identity approximation. M. Sierka, A. Hogekamp and R. Ahlrichs;J. Chem. Phys., **118**, 9136, (2003).

XXXIV. Nuclear second analytical derivative calculations using auxiliary basis set expansion. P. Deglmann, K. May, F. Furche and R. Ahlrichs; Chem. Phys. Lett., **384**, 103, (2004).

XXXV. Efficient evaluation of three-center two-electron integrals over Gaussian functions. R. Ahlrichs; Phys. Chem. Chem. Phys., **6**, 5119, (2004).

XXXVI. Analytical time-dependent density functional derivative methods within the RI-J approximation, an approach to excited states of large molecules. D. Rappoport and F. Furche; J. Chem. Phys., **122**, 064105 (2005).

XXXVII. Density functional theory for excited states: equilibrium structure and electronic spectra. F. Furche and D. Rappoport; Ch. III of "Computational Photochemistry", Ed. by M. Olivucci, Vol. 16 of "Computational and Theoretical Chemistry", Elsevier, Amsterdam, 2005.

XXXVIII. Structure optimizations for excited states with correlated second-order methods: CC2, CIS(D_∞), and ADC(2). C. Hättig; Adv. Quant. Chem., **50**, 37-60 (2005).

XXXIX. Distributed memory parallel implementation of energies and gradients for second-order Møller-Plesset perturbation theory with the resolution-of-the-identity approximation. C. Hättig, A. Hellweg and A. Köhn; Phys. Chem. Chem. Phys., **8**, 1159-1169, (2006).

XL. Quintuple-ζ quality coupled-cluster correlation energies with triple-ζ basis sets. D. P. Tew, W. Klopper, C. Neiss and C. Hättig; Phys. Chem. Chem. Phys., **9** 921–1930 (2007).

XLI. Benchmarking the performance of spin-component scaled CC2 in ground and electronically excited states. A. Hellweg, S. A. Grün and C. Hättig; Phys. Chem. Chem. Phys., **10**, 4119-4127 (2008).

XLII. Scaled opposite-spin CC2 for ground and excited states with fourth order scaling computational costs. N. O. C. Winter and C. Hättig; J. Chem. Phys., **134**, 184101 (2011).
and: Quartic scaling analytical gradients of scaled opposite-spin CC2. N. O. C. Winter and C. Hättig; *Chem. Phys.*, **401** (2012) 217.

XLIII. The MP2-F12 Method in the TURBOMOLE Programm Package. R. A. Bachorz, F. A. Bischoff, A. Glöß, C. Hättig, S. Höfener, W. Klopper and D. P. Tew; J. Comput. Chem., **32**, 2492–2513 (2011).

XLIV. Accurate and efficient approximations to explicitly correlated coupled-cluster singles and doubles, CCSD-F12. C. Hättig, D. P. Tew and A, Köhn; J. Chem. Phys., **132**, 231102 (2010).

XLV. Large scale polarizability calculations using the approximate coupled cluster model CC2 and MP2 combined with the resolution-of-the identity approximation. D. H. Friese, N. O. C. Winter, P. Balzerowski, R. Schwan and C. Hättig; J. Chem. Phys., **136**, 174106 (2012).

XLVI. A $\mathcal{O}(\mathcal{N}^3)$-scaling PNO-MP2 method using a hybrid OSV-PNO approach with an iterative direct generation of OSVs. G. Schmitz, B. Helmich and C. Hättig; Mol. Phys., **111**, 2463–2476, (2013).

XLVII. Explicitly correlated PNO-MP2 and PNO-CCSD and its application to the S66 set and large molecular systems. G. Schmitz, C. Hättig and D. P. Tew; Phys. Chem. Chem. Phys., **16**, 22167–22178 (2014).

XLVIII. Density functional theory for molecular and periodic systems using density fitting and continuous fast multipole methods. R. Łazarski, A. M. Burow and M. Sierka; J. Chem. Theory Comput., **11**, 3029–3041 (2015).

XLIX. Low-memory iterative density fitting. L. Grajciar; J. Comput. Chem., **36**, 1521–1535 (2015).

L. Linear scaling hierarchical integration scheme for the exchange-correlation term in molecular and periodic systems. A. M. Burow and M. Sierka; J. Chem. Theory Comput., **7**, 3097–3104 (2011).

LI. Resolution of identity approximation for the Coulomb term in molecular and periodic systems. A. M. Burow, M. Sierka and F. Mohamed; J. Chem. Phys., **131**, 214101 (2009).

LII. Self-consistent treatment of spin-orbit interactions with efficient Hartree-Fock and density functional methods. M. K. Armbruster, F. Weigend, C. van Wüllen and W. Klopper; Phys. Chem. Chem. Phys., **10**, 1748–1756, (2008).

LIII. Seminumerical exchange and two-component local hybrids. P. Plessow and F. Weigend; J. Comput. Chem., **33**, 810–816 (2012).

LIV. Improved SCF treatment of spin-orbit interactions and gradients. A. Baldes and F. Weigend; Mol. Phys., **111**, 2617–2624 (2013).

LV. Relativistic all-electron approaches (BSS, DKH, and X2C). Daoling Peng, Nils Middendorf, Florian Weigend, Markus Reiher; J. Chem. Phys., **138**, 184105 (2013).

LVI. Relativistic all-electron approaches including finite nucleus model and SNSO approach, and geometry gradients. Y. J. Franzke, N. Middendorf and F. Weigend; J. Chem. Phys., **148**, 104110 (2018).

LVII. NMR Shielding Tensors and Chemical Shifts in Scalar-Relativistic Local Exact Two-Component Theory. Y. J. Franzke and F. Weigend; J. Chem. Theory Comput., **15**, 1028–1043 (2019).

LVIII. Hyperfine Coupling Constants in Local Exact Two-Component Theory. Y. J. Franzke and J. M. Yu; J. Chem. Theory Comput., **18**, 323–343 (2022).

LIX. Quasi-Relativistic Calculation of EPR g Tensors with Derivatives of the Decoupling Transformation, Gauge-Including Atomic Orbitals, and Magnetic Balance. Y. J. Franzke and J. M. Yu; J. Chem. Theory Comput., **18**, 2246–2266 (2022).

LX. Exact Two-Component Theory Becoming an Efficient Tool for NMR Shieldings and Shifts with Spin–Orbit Coupling. Y. J. Franzke and C. Holzer; ChemRxiv DOI: 10.26434/chemrxiv-2023-k8jwn.

LXI. Efficient self-consistent implementation of local hybrid functionals. H. Bahmann and M. Kaupp; J. Chem. Theory Comput., **11**, 1540–1548, (2015).

LXII. Implementation of molecular gradients for local hybrid density functionals using seminumerical integration techniques. S. Klawohn, H. Bahmann and M. Kaupp; J. Chem. Theory Comput., **12**, 4254–4262, (2016).

LXIII. Efficient semi-numerical implementation of global and local hybrid functionals for time-dependent density functional theory. T. M. Maier, H. Bahmann and M. Kaupp; J. Chem. Theory Comput., **11**, 4226–4237, (2015).

LXIV. Quasirelativistic two-component core excitations and polarisabilities from a damped-response formulation of the Bethe–Salpeter equation. M. Kehry, Y. J. Franzke, C. Holzer and W. Klopper; Mol. Phys., 118, e1755064 (2020).

LXV. An improved seminumerical Coulomb and exchange algorithm for properties and excited states in modern density functional theory. C. Holzer; J. Chem. Phys., **153**, 184115 (2020).

LXVI. Assessing the accuracy of local hybrid density functional approximations for molecular response properties. C. Holzer, Y. J. Franzke, M. Kehry; J. Chem. Theory. Comput., **17**, 2928–2947 (2021).

LXVII. Development and implementation of excited-state gradients for local hybrid functionals. R. Grotjahn; F. Furche; M. Kaupp, J. Chem. Theory Comput., **15**, 5508–5522, (2019).

LXVIII. A local hybrid exchange functional approximation from first principles. C. Holzer, Y. J. Franzke, J. Chem. Phys., **157**, 034108 (2022).

Basis sets

The following tables can be used to find the proper citations of the standard orbital and auxiliary basis sets in the TURBOMOLE basis set library. Recommendations for applications and a historical overview are provided in the supporting information of [2]. There, the employed ECPs for heavy elements are listed. ECPs can also be obtained from the website of the Dolg group together with a detailed bibliography, please see http://www.tc.uni-koeln.de/PP/clickpse.en.html.

Orbital basis sets, elements H–Kr

	H,He	Li	Be	B–Ne	Na,Mg	Al–Ar	K	Ca	Sc–Zn	Ga–Kr
SVP, SV(P)	x	a	a	a	a	a	a	a	a	a
TZVP	x	b	b	b	b	b	b	b	b	b
TZVPP	x	f	f	f	f	f	f	f	f	f
QZVP, QZVPP	i									
def2-SV(P)	j	j	j	j	j	j	j	j	j	j
def2-SVP	j	j	j	j	j	j	j	j	j	j
def2-TZVP	j	j	j	j	j	j	j	j	j	j
def2-TZVPP	j	j	j	j	j	j	j	j	j	j
def2-XVPD/XVPPD, X=S,T,Q	m									
x2c-XVP (X=S, TZ), PP, -2c	j	r								
x2c-XVP-s (X=S, TZ)	s									
x2c-QZVP, PP, -2c, -s	j	t								

Note: For H–Kr def-SV(P), def-SVP, ... are identical with the basis sets without def prefix. def2-QZVPP and def2-QZVP are identical with QZVPP and QZVP. One-component dhf and def2 type basis sets are identical for the elements up to Kr.
def2-XVPD/XVPPD denotes the property–optimized augmentations def2-SVPD, def2-TZVPD, def2- TZVPPD, def2-QZVPD, def2-QZVPPD.

Orbital basis sets, elements Rb–Rn

	Rb	Sr	Y–Cd	In–Xe	Cs	Ba	La, Hf–Hg	Ce–Lu	Tl–At	Rn
def-SVP, def-SV(P), def-TZVP	d							-	d	j
def-TZVPP	f	d	f			d	f	-	d	j
def2-SV(P)	j	j	j	j	j	j	j	n	j	j
def2-SVP	j	j	j			j	j	n	j	j
def2-TZVP, def2-TZVPP	j							n	j	
def2-QZVP, def2-QZVP	j							n	j	
def2-XVPD/XVPPD, X=S,T,Q	m							-	m	
dhf-XVP (X=S–QZ), PP, -2c	q			j	q			–	q	
x2c-XVP (X=S, TZ), PP, -2c	r									
x2c-XVP-s (X=S, TZ)	s									
x2c-QZVP, PP, 2c-, -s	t									

Auxiliary basis sets for RI-J in HF/DFT (Coulomb fitting)

	H, He	Li–Kr	Rb–La	Ce–Lu	Hf–At	Rn
(def-)SVP,(def-)SV(P)	c	c	d	-	d	e
(def-)TZVP	d	d	d	-	d	e
def2, universal	e			n	e	
x2c-XVP, PP, -2c, -s (X=S,TZ)	r					
x2c-QZVP, PP, -2c, -s	r, t					

Auxiliary basis sets for RI-K in HF/DFT (Exchange fitting)

	H	He	B–F	Ne	Al–Cl	Ar	Ga-Br	Kr-La	Ce–Lu	Hf–Rn
(def-)TZVPP	o	p	o	p	o	p	o	-		
def2, universal	p								n	p
cc-pVXZ (X = T, Q, 5)	o	-	o	-	o	-	o	-		

Auxiliary basis sets for RI-MP2 and RI-CC, elements H–Ar

	H	He	Li	Be	B–F	Ne	Na,Mg	Al–Cl	Ar
SVP,SV(P)	f	k	f	f	f	k	f	f	k
TZVP,TZVPP	f	k	f	f	f	k	f	f	k
QZVP,QZVPP	k								
def2-SV(P)	f	k	l	f	f	k	l	f	k
def2-SVP	f	k	l	f	f	k	l	f	k
def2-TZVP,def2-TZVPP	f	k	f	l	f	k	l	l	k
def2-XVPD/XVPPD, X=S,T,Q	v								
(aug-)cc-pVXZ, X=D–Q	h	h	k	k	h	h	k	h	h
(aug-)cc-pV5Z	k	k	-	-	k	k	-	k	k
cc-pwCVXZ, X=D–5	-	-	-	-	k	k	-	k	k

Note: the auxiliary basis sets for the (aug-)cc-pV(X+d)Z basis sets for Al–Ar are identical with the (aug-)cc-pVXZ auxiliary basis sets.

Auxiliary basis sets for RI-MP2 and RI-CC, elements K–Kr

	K	Ca	Sc–Zn	Ga–Br	Kr
SVP, SV(P)	f	f	f	f	k
TZVP, TZVPP	f	f	f	f	k
QZVP, QZVPP	k				
def2-SV(P)	l	f	f	f	k
def2-SVP	l	f	l	f	k
def2-TZVP, def2-TZVPP	l	f	l	f	k
def2-XVPD/XVPPD, X=S,T,Q	v				
(aug-)cc-pVXZ, X=D–Q	-	-	-	h	h
(aug-)cc-pV5Z	-	-	-	u	u
cc-pCWVXZ, X=D–5	-	-	-	u	u
(aug-)cc-pVXZ-PP, X=D–5	-	-	-	u	u
cc-pwCVXZ-PP, X=D–5	-	-	-	u	u

Auxiliary basis sets for RI-MP2 and RI-CC, elements Rb–Rn

	Rb	Sr	Y–Cd	In–Xe	Cs	Ba	La, Hf–Hg	Ce–Lu	Tl–At	Rn
def-SVP,def-SV(P)	f							-	f	l
def2-SVP,def2-SV(P)	l	f	f	l	l	f	f	w	l	l
def-TZVP,def-TZVPP	f							-	f	l
def2-TZVP,def2-TZVPP	l							w	l	
def2-QZVP,def2-QZVP	l							w	l	
def2-XVPD/XVPPD, X=S,T,Q	v							-	v	
aug-cc-pVXZ-PP, X=D–5	-	-	-	u	-	-	-	-	u	u
cc-pwCVXZ-PP, X=D–5	-	-	-	u	-	-	-	-	u	u

a. Fully Optimized Contracted Gaussian Basis Sets for Atoms Li to Kr. A. Schäfer, H. Horn and R. Ahlrichs; J. Chem. Phys., **97**, 2571 (1992).

b. Fully Optimized Contracted Gaussian Basis Sets of Triple Zeta Valence Quality for Atoms Li to Kr. A. Schäfer, C. Huber and R. Ahlrichs; J. Chem. Phys., **100**, 5829 (1994).

c. Auxiliary Basis Sets to Approximate Coulomb Potentials. K. Eichkorn, O. Treutler, H. Öhm, M. Häser and R. Ahlrichs; Chem. Phys. Lett., **242**, 652 (1995).

d. Auxiliary basis sets for main row atoms and transition metals and their use to approximate Coulomb potentials. K. Eichkorn, F. Weigend, O. Treutler and R. Ahlrichs; Theor. Chem. Acc., **97**, 119 (1997).

e. Accurate Coulomb-fitting basis sets for H to Rn. F. Weigend; Phys. Chem. Chem. Phys., **8**, 1057 (2006).

f. RI-MP2: Optimized Auxiliary Basis Sets and Demonstration of Efficiency. F. Weigend, M. Häser, H. Patzelt and R. Ahlrichs; Chem. Phys. Lett., **294**, 143 (1998).

g. Contracted all-electron Gaussian basis sets for Rb to Xe. R. Ahlrichs and K. May; Phys. Chem. Chem. Phys., **2**, 943 (2000).

h. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. F. Weigend, A. Köhn and C. Hättig; J. Chem. Phys., **116**, 3175 (2002).

i. Gaussian basis sets of quadruple zeta valence quality for atoms H–Kr. F. Weigend, F. Furche and R. Ahlrichs; J. Chem. Phys., **119**, 12753 (2003).

j. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design an assessment of accuracy. F. Weigend and R. Ahlrichs; Phys. Chem. Chem. Phys., **7**, 3297 (2005).

k. Optimization of auxiliary basis sets for RI-MP2 and RI-CC2 calculation: Core-valence and quintuple-ζ basis sets for H to Ar and QZVPP basis sets for Li to Kr. C. Hättig; Phys. Chem. Chem. Phys., **7**, 59 (2005).

l. Optimized accurate auxiliary basis sets for RI-MP2 and RI-CC2 calculations for the atoms Rb to Rn. A. Hellweg, C. Hättig, S. Höfener and W. Klopper; Theor. Chem. Acc., **117**, 587 (2007).

m. Property–optimized Gaussian basis sets for molecular response calculations. D. Rappoport and F. Furche; J. Chem. Phys., **133**, 134105 (2010).

n. Error-Balanced Segmented Contracted Basis Sets of Double-ζ to Quadruple-ζ Valence Quality for the Lanthanides. R. Gulde, P. Pollak and F. Weigend; J. Chem. Theory Comput., **18**, 4062 (2012).

o. A fully direct RI-HF algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency. F. Weigend; Phys. Chem. Chem. Phys., **4**, 4285 (2002).

p. Hartree–Fock Exchange Fitting Basis Sets for H to Rn. F. Weigend; J. Comput. Chem., **29**, 167 (2008).

q. Segmented contracted basis sets for one– and two–component Dirac–Fock effective core potentials. F. Weigend and A. Baldes; J. Chem. Phys. **133**, 174102 (2010).

r. Segmented Contracted Error-Consistent Basis Sets of Double- and Triple-ζ Valence Quality for One- and Two-Component Relativistic All-Electron Calculations. P. Pollak and F. Weigend; J. Chem. Theory Comput., **13**, 3696 (2017).

s. Error-consistent segmented contracted all-electron relativistic basis sets of double- and triple-zeta quality for NMR shielding constants.
Y. J. Franzke, R. Treß, T. M. Pazdera and F. Weigend; Phys. Chem. Chem. Phys., **21**, 16658–16664 (2019).

t. Error-consistent segmented contracted all-electron relativistic basis sets of double- and triple-zeta quality for NMR shielding constants.
Y. J. Franzke, L. Spiske, P. Pollak and F. Weigend; J. Chem. Theory. Comput., accepted (2020), DOI: 10.1021/acs.jctc.0c00546.

u. Auxiliary basis sets for density-fitted correlated wavefunction calculations: Weighted core-valence and ECP basis sets for post-*d* elements. C. Hättig, G. Schmitz and J. Koßmann; Phys. Chem. Chem. Phys., 14, 6549 (2012).

v. Development of new auxiliary basis functions of the Karlsruhe segmented contracted basis sets including diffuse basis functions (def2-SVPD, def2-TZVPPD, and def2-QVPPD) for RI-MP2 and RI-CC calculations. A. Hellweg and D. Rappoport; Phys. Chem. Chem. Phys., 17, 1010 (2015).

w. Optimized auxiliary basis sets for density fitted post-Hartree-Fock calculations of lanthanide containing molecules. J. Chmela and M. E. Harding; Mol. Phys., **116**, 1523 (2018).

x. unpublished. Orbital basis sets given in the supporting information of j.

1.4 Modules and Their Functionality

For references see Bibliography.

`define` interactive input generator which creates the input file `control`. `define` supports all basis sets available from the basis set library, especially the fully atom optimized consistent basis sets of SVP, TZV and QZV quality [5–9] available for the atoms H–Rn. `define` determines the molecular symmetry and internal coordinates which allow efficient geometry optimization. `define` allows to perform a geometry optimization at a force field level to preoptimize the geometry and to calculate a Cartesian Hessian matrix. `define` sets the keywords necessary for single point calculations and geometry optimizations within a variety of methods. There are also many features to manipulate geometries of molecules: just try and see how it works.

`uff` performs a geometry optimization at a force field level. The Universal Force Field (UFF) [10] is implemented. Beyond this it calculates an analytical Hessian (Cartesian) which can be used as a start Hessian for an *ab initio* geometry optimization.

`dscf` for (semi–)direct SCF–HF and DFT calculations (see keywords for functionals supported). `dscf` supports restricted closed-shell (RHF), spin-restricted ROHF as well as UHF runs. `dscf` includes an in-core version for small molecules.

`grad` requires a successful `dscf` run and calculates the gradient of the energy with respect to nuclear coordinates for all cases treated by `dscf`.

`ridft` and `rdgrad` perform (direct) SCF–HF and DFT calculations—as `dscf` and `grad`—within the very efficient (multipole accelerated) RI–J approximation for the interelectronic Coulomb term. These programs also permit to approximate HF exchange within the RI–K approximation or using semi-numerical techniques. The exchange correlation functionals supported are specified in `define`. Relativistic two-component calculations are also available.

`riper` performs DFT calculations for molecules and periodic systems using the RI technique. In addition, a low-memory RI implementation based on preconditioned conjugate gradient algorithm is available for molecular systems. Furthermore, a real time-time dependent DFT implementation based on Magnus expansion for the time evolution operator is available for molecular systems. Both RHF and UHF runs are supported. Currently hybrid functionals are not supported.

`mpgrad` requires a well converged SCF run—by `dscf`, see keywords—and performs closed-shell RHF or UHF calculations yielding single point MP2 energies and, if desired, the corresponding gradient. Note that `mpgrad`

performs conventional, i.e. non-RI MP2 calculations only. For real-life applications it is highly recommended to use RI-MP2 instead (see module `ricc2`).

`ricc2` calculates electronic ground and excitation energies, transition moments and properties of ground and excited states at the MP2, CIS, CIS(D), ADC(2) and CC2 level using either a closed-shell RHF or a UHF SCF reference function. Calculates R12 basis set limit correction for MP2 energies. Employs the RI technique to approximate two-electron integrals. [11–18].

`ccsdf12` calculations of electronic ground state energies beyond MP2/CC2: RI-MP2-F12, MP3, MP3-F12, MP4, MP4(F12*), CCSD, CCSD(F12), CCSD(F12*), CCSD(F12)(T), CCSD(F12*)(T) and electronic excitation energies at the CCSD level. [19–22]

`pnoccsd` calculations of electronic ground state energies with PNO-based methods starting from MP2 and MP2-F12 up to PNO-CCSD(T). [23, 24]

`relax` requires a gradient run—by `grad`, `rdgrad`, `ricc2`, `egrad`, or `mpgrad`—and proposes a new structure based on the gradient and the approximated force constants. The approximated force constants will be updated. This module will not be used by default any more if `jobex` is called.

`statpt` performs structure optimization using the "Trust Radius Image Minimization" algorithm. It can be used to find minima or transition structures (first order saddle points). Transition structure searches usually require initial Hessian matrix calculated analytically or the transition vector from the lowest eigenvalue search.

`frog` executes one molecular dynamics (MD) step. Like `relax`, it follows a gradient run: these gradients are used as classical Newtonian forces to alter the velocities and coordinates of the nuclei.

`aoforce` requires a well converged SCF or DFT run—by `dscf` or `ridft`, see keywords—and performs an analytic calculation of force constants, vibrational frequencies and IR intensities. `aoforce` is also able to calculate only the lowest Hessian eigenvalues with the corresponding eigenvectors which reduces computational cost. The numerical calculation of force constants is also possible (see tool `NumForce` in Section 1.5).

`escf` requires a well converged SCF or DFT run and calculates time dependent and dielectric properties (spin-restricted closed-shell or spin-unrestricted open-shell reference):

- static and frequency-dependent polarizabilities within the SCF approximation

– static and frequency-dependent polarizabilities within the time-dependent Kohn–Sham formalism, including hybrid functionals such as B3-LYP

– electronic excitations within the RHF and UHF CI(S) restricted CI method

– electronic excitations within the so-called SCF-RPA approximation (poles of the frequency dependent polarizability)

– electronic excitations within the time dependent Kohn–Sham formalism (adiabatic approximation). It can be very efficient to use the RI approximation here, provided that the functional is of non-hybrid type: we recommend B-P86 (but slightly better results are obtained for the hybrid functional B3-LYP) [25].

–stability analysis of single-determinant closed-shell wave functions (second derivative of energy with respect to orbital rotations) [26].

–relativistic two-component calculations [27–29]

–Bethe–Salpeter equation (BSE) [29–33] and the GW method [34–37]

`egrad` computes gradients and first-order properties of excited states. Well converged orbitals are required. The following methods are available for spin-restricted closed shell or spin-unrestricted open-shell reference states:

– CI-Singles approximation (TDA)

– Time-dependent Hartree–Fock method (RPA)

– Time-dependent density functional methods

`egrad` can be employed in geometry optimization of excited states (using `jobex`, see Section 5.1), and in finite difference force constant calculations (using `NumForce`). Details see [38].

`rirpa` calculates ground state energies and analytic first-order properties within the random phase approximation (RPA) and its perturbative corrections, see Section 13. A Kramers-restricted two-component formalism is implemented for ground state energies.

`mpshift` requires a converged SCF or DFT run for closed shells. `mpshift` computes NMR chemical shieldings for all atoms of the molecule at the SCF, DFT or MP2 level within the GIAO ansatz and the (CPHF) SCF approximation. From this one gets the NMR chemical shifts by comparison with the shieldings for the standard compound usually employed for this purpose, e.g. TMS for carbon shifts. Note that NMR shielding typically requires more flexible basis sets than necessary for geometries or energies. In molecules with ECP-carrying atoms, chemical shieldings on all the other atoms can be computed with `mpshift` [39, 40] in the way suggested in J. Chem. Phys. 136, 114110 (2012). Alternatively,

scalar-relativistic and spin–orbit all-electron X2C is available to treat heavy elements [41–43]. The (multipole accelerated) RI–J approximation is supported [44]. `mpshift` and `aoforce` can be used to calculate vibrational circular dichroism (VCD) spectra [45]. `mpshift`can also be employed to calculate EPR properties. DFT calculations should not be performed with gridsize m3 or m4.

`freeh` calculates thermodynamic functions from molecular data in a `control` file; an `aoforce` or a `NumForce` run is a necessary prerequisite.

`evib` calculates the matrix elements of the first order derivative of the Kohn–Sham operator with respect to atomic displacements and describes the first order electron-vibration (EV) interaction.

`intense` calculates Raman scattering cross sections from molecular data in a `control` file; an `aoforce` and an `egrad` run are a necessary prerequisite. Please use the `Raman` script to run these three steps in an automated way.

`woelfling` computes a finite number of structures along reaction paths within different interpolation algorithms. It provides an initial path using a modified Linear Synchronous Transit. See Section 5.9 for details. Please use the `woelfling-job` script to run optimizations with it.

`proper` computes a variety of first-order properties and provides several functionalities to analyse wavefunctions such as orbital localization, population analysis, natural transition orbitals, AIM critical points and paths, etc. and can generate output in a variety of plotting formats (see chapter 20).

1.5 Tools

Note: these tools are very helpful and meaningful for many features of `TURBOMOLE`.

This is a brief description of additional `TURBOMOLE` tools. Further information will be available by running the programs with the argument `-help`.

`actual` please use: `actual -help`

`adg` adds data group to control file.
E.g.: 'adg scfinstab ucis' inserts:
$scfinstab ucis

`aoforce2g98` usage: `aoforce2g98 aoforce.out > g98.out`
converts output from the `aoforce` program to Gaussian 98 style, which can be interpreted by some molecular viewer (e.g. jmol) to animate the normal coordinates.

`bend` example: `bend 1 2 3`
displays the bending angle of three atoms specified by their number from the `control` file. Note that unlike in the `TURBOMOLE` definition of internal coordinates the apex atom is the second!

`cbasopt` optimize auxiliary basis sets for RI-MP2 and RI-CC2 calculations. Uses `ricc2` to calculate the error functional and its gradient and `relax` as optimization module. For further details call `cbasopt -h`.

`cc2cosmo` manages macro iterations for RI-MP2, RI-CC2 or RI-ADC(2) calculations in an equilibrated solvent environment described by `cosmo`(see Chapter 19.2).

`cgnce` plots energies as a function of SCF iteration number (gnuplot required).

`cosmoprep` sets up `control` file for a `cosmo` run (see Chapter 19.2).

`cpt` The Color Prediction Tool predicts color for calculated and measured spectra. Simply type `cpt` in the directory of a finished TD-DFT calculation to obtain absorption and emission colors within a linear color approximation. However note that `cpt` is a standalone tool and not strictly bound to `TURBOMOLE`It may be used with data obtained from various theoretical programs as well as experimental data by reading in simple ASCII files. A detailed list of functionality is obtained from `cpt -help`.

`dist` example: `dist 1 2`
calculates atomic distances from `TURBOMOLE` input files; `dist -l 4` gives all interatomic distances to 4 a.u. (5 a.u. is the default).

`DRC` automates dynamic reaction coordinate calculations forward and backward along the imaginary vibrational mode of a transition state structure. A transition state optimization with a subsequent frequency calculation is prerequisite.
For further details call `DRC -h`.

`eiger` displays orbital eigenvalues obtained from data group `$scfmo`. Shows HOMO-LUMO gap, occupation, checks if there are holes in the occupation, and much more.

`evalgrad` reads the gradient file and prints the energies of each cycle versus bond lengths or angles. Five operational modes are possible:

`evalgrad`	prints the energy.
`evalgrad 1`	prints the coordinate of atom 1.
`evalgrad 1 2`	prints the distance between atoms 1 and 2.
`evalgrad 1 2 3`	prints the bending angle as defined in `Bend`.
`evalgrad 1 2 3 4`	prints the torsional angle as defined in `Tors`.

`file2control` This script copies the content of external data groups (file=) into the control file. If `$file2control` is found, the process is reverted.

`finit` initialises the force constant matrix for the next `statpt` or `relax` step.

`FDE` drives the Frozen Density Embedding calculations.

`Fukui` automates the calculations of Fukui functions. The density change is written in dtx files and condensed Fukui functions based on different population analyses are computed.
For further details call `Fukui -h`.

`gallier` converts IR intensities and/or VCD rotational strengths to a spectrum after the corresponding calculation was performed. Intensities can be broadened with Gaussian or Lorentzian functions.

`hcore` prepares the `control` file for a Hamilton core guess.

`jobex` usage: see Section 5.1
is the `TURBOMOLE` driver for all kinds of optimizations.

`jobbsse` usage: see `jobbsse -h`
is the driver for counterpoise corrected calculations.

`kdg` example: `kdg scfdiis`
kills a data group (here `$scfdiis`) in the `control` file.

`lhfprep` prepares for Localized Hartree-Fock calculations by adjusting parameters of the `control` file.

`log2x` converts the file logging an MD trajectory into coordinates in frames appropriate for `jmol` animation program.

`log2egy` extracts the energy data (KE, total energy, PE) from an MD log file.

`log2int` extracts bond lengths or angle from an MD log file.

`log2rog` computes the radius of gyration, geometric radius and diameter from an MD log file.

`mdprep` interactive program to prepare for an MD run, checking in particular the `mdmaster` file (`mdprep` is actually a FORTRAN program).

`MECPprep` prepares the input for minimum-energy crossing point calculations. The subdirectories `state1` and `state2` will be created. Multiplicity and charge for the two states can be set.
For further details call `MECPprep -h`.

`MECPopt` driver for geometry optimizations of minimum-energy crossing points. The electronic structure calculations are carried out in the subdirectories `state1` and `state2` and the optimizer step is performed in the starting directory.
For further details call `MECPopt -h`.

`mp2prep` prepares MP2 calculations interactively by adjusting parameters of the `control` file according to your system resources.

`NumForce` calculates numerically force constants, vibrational frequencies, and IR intensities. (Note that the name of the shell script is `NumForce` with capital `F`.)

`outp` example: `outp 1 2 3 4`
displays the out-of-plan angle between atom1 and the plane that is defined by the last three atoms. atom1 is fixed at atom4.

`panama` converts energies and oscillator strengths to a spectrum broadened by Gaussian functions and/or calculates non-relaxed difference densities of excitations.

`past` translates and rotates coordinates in the principal axis system and prints out the rotational constants.

`raman` calculates vibrational frequencies and Raman intensities. See Section 15.2 for explanation.

`screwer` distorts a molecule along a vibrational mode.

`scanprep` prepares a series of control files with frozen internal coordinates. The data group $constraints (e.g. provided by TmoleX) is evaluated.
For further details call `scanprep -h`.

`redox` automates the calculation of reduction/oxidation potentials functions w.r.t. the standard hydrogen electrode by computing the electron affinities/ionization energies in the gas phase and Gibbs free energy of solvation with DCOSMO-RS. As a side product electron reorganisation energies are printed.
For further details call `redox -h`.

`vibration` distorts a molecule along a vibrational mode or generates a plot of an IR spectrum (gnuplot required)

`sdg` shows data group from `control` file:
for example `sdg energy` shows the list of calculated energies.

`sysname` returns the name of your system, used in almost all `TURBOMOLE` scripts.

`stati` prepares the `control` file for a statistics run.

`t2x` converts `TURBOMOLE` coordinates to xyz format.

`t2aomix` creates an input file for the AOMix program. AOMix a software the analysis of molecular orbitals. For more information
see: (http://www.sg-chem.net/aomix).
Uses `tm2molden` as described below by automatically adding the `$aomix` keyword to the control file.

`tm2molden` is a versatile tool to create

- molden format input file for the Molden program,
- AOMix input files or
- detailed information about the largest AO contributions to the MOs.

Molden is a graphical interface for displaying the molecular density, MOs, normal modes, and reaction paths. For more information about molden see: http://www.cmbi.ru.nl/molden/molden.html.
This format is also often used as input for other program packages or property tools.
NOTE: The default normalization of molecular orbitals of d-type (and beyond) when using `tm2molden` is different to what Molden expects. To generate Molden input files with the Molden-own normalization, please call `tm2molden norm`, the default name of the resulting file will be *molden_std.input* rather than *molden.input*.
If `tm2molden` finds the keyword `$aomix` in the control file, it will write out an AOMix input file, see: http://www.sg-chem.net/aomix
Finally, `tm2molden` can be used to print out the largest contributions of the AO basis functions to the molecular orbitals.
Usage:

```
tm2molden mostat [molist] [above <threshold>]
```

e.g.:

```
tm2molden mostat 230-240,251,255
tm2molden mostat 434-440 above 0.001
tm2molden mostat above 0.02
```

Only contributions which are larger than a certain percentage (default is 1%) are printed, this value can be changed with the `above` option (as absolute value, so 1% is 0.01). Without a list of orbitals (the numbering follows the output of `eiger`) all MOs are printed.

`tors` is a script to query a dihedral angle in a molecular structure:
e.g. `tors 1 2 3 4` gives the torsional angle of atom 4 out of the plane of atoms 1, 2 and 3.

`tbtim` is used to convert timings output files from TURBOBENCH calculations to LaTeXtables (for options please type `TBTIM -help`).

`tblist` is used to produce summaries of timings from TURBOBENCH calculations to LaTeXformat. (for options please type `TBLIST -help`).

`uhfuse` deprecated command, present in the current version for compatibility reasons only.
Transforms the UHF MOs from a given symmetry to another symmetry, which is C_1 by default (just enter `uhfuse`). but can be specified (e.g. as C_{2v}) by entering `uhfuse -s c2v`. Now this functionality

is included in the MO definition menu of `define` program, see Section 4.3.1.

`vcd` calculates VCD rotational strengths. See Section 15.3 for explanation.

`woelfling-job` optimizes a reaction path with `woelfling`.
For further information please type `woelfling-job -h`.

`x2t` converts standard xyz files into `TURBOMOLE` coordinates.

Chapter 2

Installation of TURBOMOLE

2.1 Install TURBOMOLE command line version

Installation requires familiarity with some simple UNIX commands. The TURBOMOLE package is generally shipped as one *tar* file. This has to be uncompressed

```
gunzip turbomole_78.tar.gz
```

and unpacked

```
tar -xvf turbomole_78.tar
```

to produce the whole directory structure.

> **Important Note**: Do **NOT** install or run TURBOMOLE as root or with root permissions! The files would have the wrong user- and group-IDs and the permissions will not be set correctly!

Unpacking TURBOMOLE has to be done just once, preferably by a user who has the same group-ID as all other users of the program. The best place is on a network disk which is available on all machines. If only root is allowed to place files there, please first generate an empty directory as root (e.g. *sudo mkdir /nfs-disk/software/TURBOMOLE-78*), change the owner of the directory (e.g. *sudo chown user:usergroup /nfs-disk/software/TURBOMOLE-78*) and then unpack as non-root user the downloaded file there.

2.1.1 Settings for each user:

The environmental variable $TURBODIR must be set to the directory where TURBOMOLE has been unpacked, for example:

```
TURBODIR=/nfs_disk/software/TURBOMOLE
```

Then, the most convenient way to extend your path to the TURBOMOLE scripts and binaries is to source the file Config_turbo_env:

```
source $TURBODIR/Config_turbo_env
```

If you have a csh or tcsh as default login shell use

```
source $TURBODIR/Config_turbo_env.tcsh
```

instead.

It is recommended to add the two lines given above to your .bashrc (or .profile or wherever you prefer to add your local settings).

Note: If you do not set $TURBODIR first but use the full path to the Config_turbo_env file when sourcing, the path should be detected automatically.

2.1.2 Setting system type and $PATH by hand

This section is only needed if

- the automatic setting as described in the sub-chapter before, using the Config_turbo_env file, does not work. And/or if you:
- want to know what variables and paths have to be set to make TURBOMOLE ready for use in a detailed manner.

First check that the Sysname tool works on your computer:

```
$TURBODIR/scripts/sysname
```

should return the name of your system and this should match a bin/[arch] subdirectory in your TURBOMOLE installation.

If Sysname does not print out a single string matching a directory name in $TURBODIR/bin/, and if one of the existing binary versions does work, you can force sysname to print out whatever is set in the environment variable $TURBOMOLE_SYSNAME:

```
TURBOMOLE_SYSNAME=em64t-unknown-linux-gnu
```

Please make sure *not* to append _mpi or _smp to the string when setting $TURBOMOLE_SYSNAME, even if you intend to run parallel calculations. sysname will append this string automatically to the system name if $PARA_ARCH is set to MPI or SMP (see chapter 3.4.1 how to set up parallel environment).

You can call TURBOMOLE executables and tools easily from anywhere if you add the corresponding directories to your path (kornshell or bash syntax):

```
PATH=$PATH:$TURBODIR/scripts
PATH=$PATH:$TURBODIR/bin/‘sysname‘
```

Note that `sysname` is set in back quotes which tells the shell to substitute the entry by the *output* of `sysname`.

Now the `TURBOMOLE` executables can be called from a directory with the required input files. For example to call `dscf` and to save the output to a file named dscf.out:

```
$TURBODIR/bin/‘sysname‘/dscf > dscf.out
```

or, if the path is OK, simply

```
dscf > dscf.out
```

Executable modules are in the `bin/[arch]` directory (for example, Linux modules are in `bin/em64t-unknown-linux-gnu`). Tools (including `jobex`) are in `scripts` and (auxiliary) basis sets are kept in the directories `basen`, `jbasen`, `jkbasen`, `cbasen`, `xbasen` and `cabasen`. Coordinates for some common chemical fragments are supplied in `structures`. The documentation and a tutorial can be found in the folder `DOC`.

2.1.3 Testing the installation

In addition, some sample calculations are supplied in `Turbotest` so that the modules can be tested. Just run `TTEST` from this directory to run all tests or `TTEST -help` to get help on how this works:

```
cd $TURBODIR/TURBOTEST
TTEST
```

2.2 Installation problems: How to solve

Please check your user limits!

If one or several tests of the test suite fail, it is very likely that your user limits for stack size and/or memory are too small.

`sh/bash/ksh` users: please do a

```
ulimit -a
```

to get your actual limits. The output should look like:

```
core file size (blocks)     0
data seg size (kbytes)      unlimited
```

```
file size (blocks)          unlimited
max locked memory (kbytes)  unlimited
max memory size (kbytes)    unlimited
open files                  1024
pipe size (512 bytes)       8
stack size (kbytes)         unlimited
cpu time (seconds)          unlimited
max user processes          8191
virtual memory (kbytes)     unlimited
```

The most important entries are data size, stack size, max memory size, and virtual memory. Those should be either unlimited or as big as your total RAM.

To set, e.g. the stack size to the maximum allowed size on your system (the so called *hard* limit), do:

```
ulimit -s hard
```

`csh/tcsh` users: please do `limit` instead of ulimit and check the output.

Again, like given above, the limits should be at least as high as your memory available. The syntax for changing the limits to unlimited using csh/tcsh is:

```
limit stacksize hard
```

Note that on some machines the option `hard` leads to an error and is not recognized. In such cases try `ulimit -s unlimited` or set it to a value like `ulimit -s 4019516` (value is in kB, so this example sets it to 4GB).

If you are using a queuing system:

Note that if you are submitting jobs to a queue, the user limits might be different from what you get when you log in on the machines! To check your limits, you have to add `ulimit` or `limit` in the script that is sent to the queue:

```
....
ulimit -a > mylimits.out
jobex -ri -c 200 -statpt > jobex.out
...
```

send it to the queue and check the file *mylimits.out* to find out which limits are set.

Parallel version:

The parallel binaries are being started by the mpirun command which often uses ssh to start a process on a remote node. The limits for the stack size can not be set by the user in such a case, so everything in `$HOME/.profile`, `$HOME/.bashrc`, etc. will not help to get rid of the problem.

To check the limits on a remote node, try (sh/bash/ksh syntax):

```
ssh <hostname> ulimit -a
```

If the ssh command gives a lower stack size than *unlimited* or a large number, you have to change the file

/etc/security/limits.conf

on *all* nodes where the parallel binaries might run, and add there the line (example for 4GB limit)

```
*                soft    stack           4194303
```

Redo *ssh <hostname>ulimit -a* and you should get 4GB stack size limit, as it is set in limits.conf now.

Chapter 3

How to Run TURBOMOLE

All TURBOMOLE modules need the `control` file as input file. The `control` file provides directly or by cross references the information necessary for all kinds of calculations. There are different ways to generate the input file, depending on how you want to use TURBOMOLE.

3.1 Writing simple input files (withdout using define)

For standard non-relativistic DFT (or HF-SCF) ground-state calculations TURBOMOLE requires only a minimal input:

- generate your atomic coordinates by any tool you are familiar with, save it as an `.xyz` file (xyz is a file format for coordinates) and use the TURBOMOLE script `x2t` to convert the `.xyz` file into the TURBOMOLE format:
  `x2t xyzinputfile > coord`
  or use a conversion tool like `babel`:
  `babel -i` *input-type* `-o tmol coord`
  where *input-type* should be babel's abbreviation for the format of your input file.
- generate then the `control` file with, e.g., the following content:

```
$atoms
   basis = def2-SV(P)
$coord file=coord
$dft
   funtional b-p
   grid m3
$end
```

- you can then start either a single-point calculation with

```
dscf > dscf.out
```

or a geometry optimization with

```
jobex
```

- if you want to use the RI-J approximation (`ridft`) add a line with `$rij`. In that case single-point calculations should be started with

```
ridft > ridft.out
```

and geometry optimizations with

```
jobex -ri
```

- for vibrational frequencies and IR intensities invoke after a converged geometry optimization the program `aoforce`:

```
aoforce > aoforce.out
```

- you can optionally include a title line that will be printed in the outputs

```
$title
    DFT/B-P/def2-SV(P) for MyMolecule
```

The DFT and HF-SCF programs `dscf` and `ridft` will automatically detect the molecular point group from the provided coordinates and exploit the point group symmetry. By default the programs will do an Extended Hückel Theory (EHT) guess for the (neutral) molecule to generate start orbitals and determine the orbital occupation from the aufbau principle with EHT orbital energies.

If you want to do a calculation on a charged species or for a different number of unpaired electrons, or, more precisely, a different minor spin quantum number M_S, this can be specified in the `control` file with:

`$eht charge=`n `unpaired=`m

where n and m have to be integer numbers and m has to be positive.

Alternatively, you can use any of the other available options to define the orbitals occupations and generate start orbitals.

The Cartesian positions of atoms can be fixed by adding in the `$coord` data group a `f` behind the atom symbol:

```
$coord
  0.000   0.000  -0.764 o f
  1.500   0.000   0.382 h f
 -1.500   0.000   0.368 h
```

Some basis sets require for heavier atoms the addition of effective core potentials (ECPs). This can be done as follows:

```
$atoms
   basis = dhf-SVP-2c
pb 2-4,7
   ecp   = pb dhf-ecp-2c
```

Some old basis sets from Pople and coworkers (e.g. 6-31G(d)) should be used with 6-component, i.e. cartesian, d-functions, while TURBOMOLE's default is to use 5-component spherical d-functions. The use of 6 component d-functions and the full spherical sets for higher angular momentum functions can be requested by adding the keyword

```
$pople CAO
```

For the additional input needed for post-HF or post-KS calculations please see the following sections and chapters.

3.1.1 Symmetry handling

As mendioned above, the dscf and ridft programs per default automatically detect the molecular point group from the provided coordinates. If symmetry elements are found, the molecular structure will be shifted and rotated into the standard orientation needed for the detected point group (e.g. the main rotation axis aligned with the z axis)

If a calculation should for any reason not use the full molecular point group, the point group actually used in the calculation can be set in the `control` file with:

`$symmetry` *sflies*

where *sflies* should be the Schönflies symbol in lower case letters without using any special symbol to indicate lower indices, i.e. `c2v` for C_{2v}. Note, however, that for these cases the coordinates that you provide have to be for the standard orientation expected by TURBOMOLE.

With `c1` the use of point group symmetry is disabled.

3.1.2 Geometry optimizations

Geometry optimizations on molecular (i.e. non-periodic) systems are by default done in internal redundant coordinates that will be generated automatically by the optimizer `statpt` when it is invoked for the first time. If you want to set up different internal coordinates or if you want to freeze internal coordinates you can do this with the input generator `define` (*vide infra*).

For geometry optimizations in cartesians coordinates you need to disable the use of internal coordinates by setting the data group `$optimize` accordingly (*vide infra*).

3.2 The graphical user interface TmoleX

An easy way to start as a newbie with TURBOMOLE is to use the free graphical user interface TmoleX which is part of every TURBOMOLE distribution. Please install TmoleX

on your local desktop computer and avoid running the GUI on remote machines using remote desktop tools like X11. If you do not have a local version of TmoleX, consider to download and use the free version which is able to generate input, to run TURBOMOLE jobs on external (Linux) boxes and to visualize the results - download is available from here: BIOVIA TURBOMOLE

A detailed tutorial for the usage of TURBOMOLE on the command line can be found in the DOC directory of your TURBOMOLE installation and on the TURBOMOLE website www.turbomole.org

3.3 A 'Quick and Dirty' Tutorial for the define input generator

For the generation of input files for more complex calculations TURBOMOLE offers the interactive input generator `define`, which guides the user through a series of menues to set up the required input without the need to know by hard the names of the keywords and options.

The `define` module (program) generates in a step by step manner and interactively the `control` file: coordinates, atomic attributes (e.g. basis sets), MO start vectors and keywords specific for the desired method of calculation. We recommend generating a set of Cartesian coordinates for the desired molecule using your favourite molecular builder (e.g. `molden`) and converting these coordinates into TURBOMOLE format (see Section 24.2) as input for `define`. Alternatively the graphical user interface TmoleX can be used to import and/or build molecules.

A straightforward way to perform even complex TURBOMOLE calculations from scratch is as follows:

- generate your atomic coordinates by any tool you are familiar with,
- save it as an `.xyz` file which is a standard output format of all programs, or use a conversion tool like `babel`,
- use the TURBOMOLE script `x2t` to convert your `.xyz` file to the TURBOMOLE `coord` file:
  ```
  x2t xyzinputfile > coord
  ```
- **since input files for TURBOMOLE are always called `control`, each input has to be placed in a different directory. Create a new directory and copy the coord file there**,
- call
  `define`
  after specifying the title, you get the coord menu — just enter
  `a coord`
  to read in the coordinates.
  Use `desy` to let `define` determine the point group automatically.

If you want to do geometry optimizations, we recommend to use generalized internal coordinates; `ired` generates them automatically.

- you may then go through the menus without doing anything: just press `<Enter>`, `*` or `q`—whatever ends the menu, or by confirming the proposed decision of `define` again by just pressing `<Enter>`.
  This way you get the necessary specifications for a (SCF-based) run with def-SV(P) as the default basis set (which is qualitatively similar to 6-31G*).
- for more accurate SCF or DFT calculations choose larger basis sets, e.g. TZVP by entering `b all def-TZVP` or `b all def2-TZVP` in the basis set menu.
- ECPs which include (scalar) relativistic corrections are automatically used beyond Kr.
- an initial guess for MOs and occupation numbers is provided by `eht`
- for DFT you have to enter `dft` in the last menu and then enter `on`
- for efficient DFT calculations you best choose the RI approximation by entering `ri` and then `on`. For small molecules it can be beneficial to provide additional memory (with `m` *number*; *number* in MB), but make sure not to use more than 80% of the memory your computer has available (note that the setting is per core for parallel jobs!). Auxiliary basis sets are provided automatically. For medium-sized to larger molecules, additional memory for integral-storage is not helpful (can even slow down the calculation), but activating the multipole accelerated RI-J (`marij`) can speed up the calculation significantly (without introducing additional errors for RI-J).
- B-P86 is the default functional. It has a good and stable performance throughout the periodic system.
- for an HF or DFT run without RI, you simply enter:
  ```
  [nohup] dscf > dscf.out &
  ```
  or, for a RI-DFT run:
  ```
  [nohup] ridft > ridft.out &
  ```
- for a gradient run, you simply enter:
  ```
  [nohup] grad > grad.out &
  ```
  or
  ```
  [nohup] rdgrad > rdgrad.out &
  ```
- for a geometry optimization simply call `jobex`:
  for a standard SCF input:
  ```
  [nohup] jobex &
  ```
  for a standard RI-DFT input:
  ```
  [nohup] jobex -ri &
  ```
- many features, such as NMR chemical shifts or vibrational frequencies at SCF or DFT level, do not require further modifications of the input. Just call e.g. `mpshift` or `aoforce` after the appropriate energy calculation.

- other features, such as post–SCF methods need further action on the input, using either the last menu of define where one can activate all settings needed for DFT, TDDFT, MP2, CC2, etc. calculations (this is the recommended way), or tools like `mp2prep`.

If that was a too quick and dirty chapter, please read the `TURBOMOLE` Tutorial in the DOC directory of your local `TURBOMOLE` installation. It explains step by step the generation of input with `define` and how to run calculations on the command line.

3.3.1 Single Point Calculations: Running TURBOMOLE Modules

All calculations are carried out in a similar way. First you have to run `define` to obtain the `control` file or to add/change the keywords you need for your purpose. This can also be done manually with an editor. Given a bash and a path to `$TURBODIR/bin/`[arch] (see installation, Chapter 2) you call the appropriate module in the following way (e.g. module `dscf`):

```
nohup dscf > dscf.out &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. The output will be written to the file `dscf.out`. Several modules write some additional output to the `control` file. For the required keywords see Section 23. The features of `TURBOMOLE` will be described in the following section.

3.3.2 Energy and Gradient Calculations

Energy calculations may be carried out at different levels of theory.

Hartree–Fock–SCF
: use modules `dscf` and `grad` or `ridft` and `rdgrad` to obtain the energy and gradient. The energy can be calculated after a `define` run without any previous runs. `dscf` and `grad` need no further keywords `ridft` and `rdgrad` only need the keyword `$rij`. The gradient calculation however requires a converged `dscf` or `ridft` run.

Density functional theory
: DFT calculations are carried out in exactly the same way as Hartree–Fock calculations except for the additional keyword `$dft`. For DFT calculations with the fast Coulomb approximation you have to use the modules `ridft` and `rdgrad` instead of `dscf` and `grad`. Be careful: `dscf` and `grad` ignore RI–K flags and will try to do a normal calculation, but they will not ignore RI–J flags (`$rij`) and stop with an error message. To obtain correct derivatives of the DFT energy expression in `grad` or `rdgrad` the program also has to consider derivatives of the quadrature weights—this option can be enabled by adding the keyword `weight derivatives` to the data group `$dft`.

For a semi-direct `dscf` calculation (Hartree–Fock or DFT) you first have to perform a statistics run. If you type

```
stati dscf
nohup dscf > dscf.stat &
```

the disk space requirement (MB) of your current `$thime` and `$thize` combination will be computed and written to the data group `$scfintunit size=`*integer* (see Section 23.2.10). The requirement of other combinations will be computed as well and be written to the output file `dscf.stat`. The size of the integral file can be set by the user to an arbitrary (but reasonable) number. The file will be written until it reaches the given size and `dscf` will continue in direct mode for the remaining integrals. Note that `TURBOMOLE` has no 2GB file size limit.

MP2 and MP2-F12

MP2 calculations need well converged SCF runs (the SCF run has to be done with at least the density convergence `$denconv 1.d-7`, and `$scfconv 7` as described in Section 23). This applies also to the spin-component scaled (SCS and SOS) and explicitly-correlated (F12) variants of MP2. For MP2 and MP2-F12 calculations in the RI approximation use the `ricc2` or `pnoccsd` modules. The module `mpgrad` calculates the conventional (non-RI and non-F12) MP2 energy its gradient (only recommended for test calculations). The input can be prepared with the `mp2`, `cc`, or `pnocc` menu in `define`.

Excited states with CIS, TDHF and TDDFT (`escf`)

Single point excited state energies for CIS, TDHF, and TDDFT methods can be calculated using `escf`. Excited state energies, gradients, and other first order properties are provided by `egrad`. Both modules require well converged ground state orbitals.

Excited states with second-order wavefunction methods (`ricc2`)

The module `ricc2` calculates beside MP2 and CC2 ground state energies also CIS (identical to CCS), CIS(D), CIS(D_∞), ADC(2) or CC2 excitation energies using the resolution-of-the-identity (RI) approximation. Also available are spin-component scaled (SCS and SOS) variants of the second-order methods CIS(D), CIS(D_∞), ADC(2) or CC2. Excited state gradients are available at the CCS, CIS(D_∞), ADC(2), and CC2 levels and the spin-component scaled variants of the latter three methods. In addition, transition moments and first-order properties are available for some of the methods. For more details see Section 10. The input can be prepared using the `cc` menu of `define`.

Coupled-Cluster methods beyond CC2: CCSD(F12*)(T) (`ccsdf12`)

Coupled-Cluster methods beyond CC2 as CCSD and CCSD(T) and Møller-Plesset perturbation theory beyond MP2 and explicitly-correlated F12 variants thereof are since Release V7.0 implemented in the `ccsdf12` program. The F12 variants of these methods have a much faster basis set convergence and are therefore more efficient. We recommend in particular CCSD(F12*)

and CCSD(F12*)(T). Excitation energies are only available for (conventional) CCSD.

3.3.3 Calculation of Molecular Properties

See Section 1.4 for the functionality and Section 23 for the required keywords of the modules `dscf`, `ridft`, `mpshift`, `escf`, and `ricc2`.

3.3.4 Modules and Data Flow

See Figure 3.1.

3.4 Parallel Runs

Some of the `TURBOMOLE` modules are parallelized using the message passing interface (MPI) for distributed and shared memory machines or with OpenMP or multi-threaded techniques for shared memory and multi-core machines.

Generally there are two hardware scenarios which determine the kind of parallelization that is possible to use:

- On a **single node** with several CPUs and/or cores using the same memory (shared memory), the user can run all parallelized modules of `TURBOMOLE` . For some modules, both shared-memory and MPI versions are available, but it is recommended not to use the latter ones for performance reasons.

  How to run the parallel `TURBOMOLE` SMP version on multi-core and/or multi-CPU systems: Please see chapter 3.4.2.

- On a **cluster** a parallel calculation can be performed using several distinct nodes, each one with local memory and disks. This can be done with the MPI version. It is, however, often more efficient to use the SMP version also on a cluster by running each individual job on a single node using all cores.

  How to run the parallel `TURBOMOLE` MPI version on clusters: Please see chapter 3.4.1.

The list of programs parallelized includes presently:

- `ridft` — parallel ground state Hartree-Fock and DFT energies including RI-J and the multipole accelerated RI (MA-RI-J), SMP and MPI
- `rdgrad` — parallel ground state gradients from `ridft` calculations, SMP and MPI
- `dscf` — Hartree-Fock and DFT ground state calculations for all available DFT functionals, without the usage of RI-J approximation, SMP and MPI

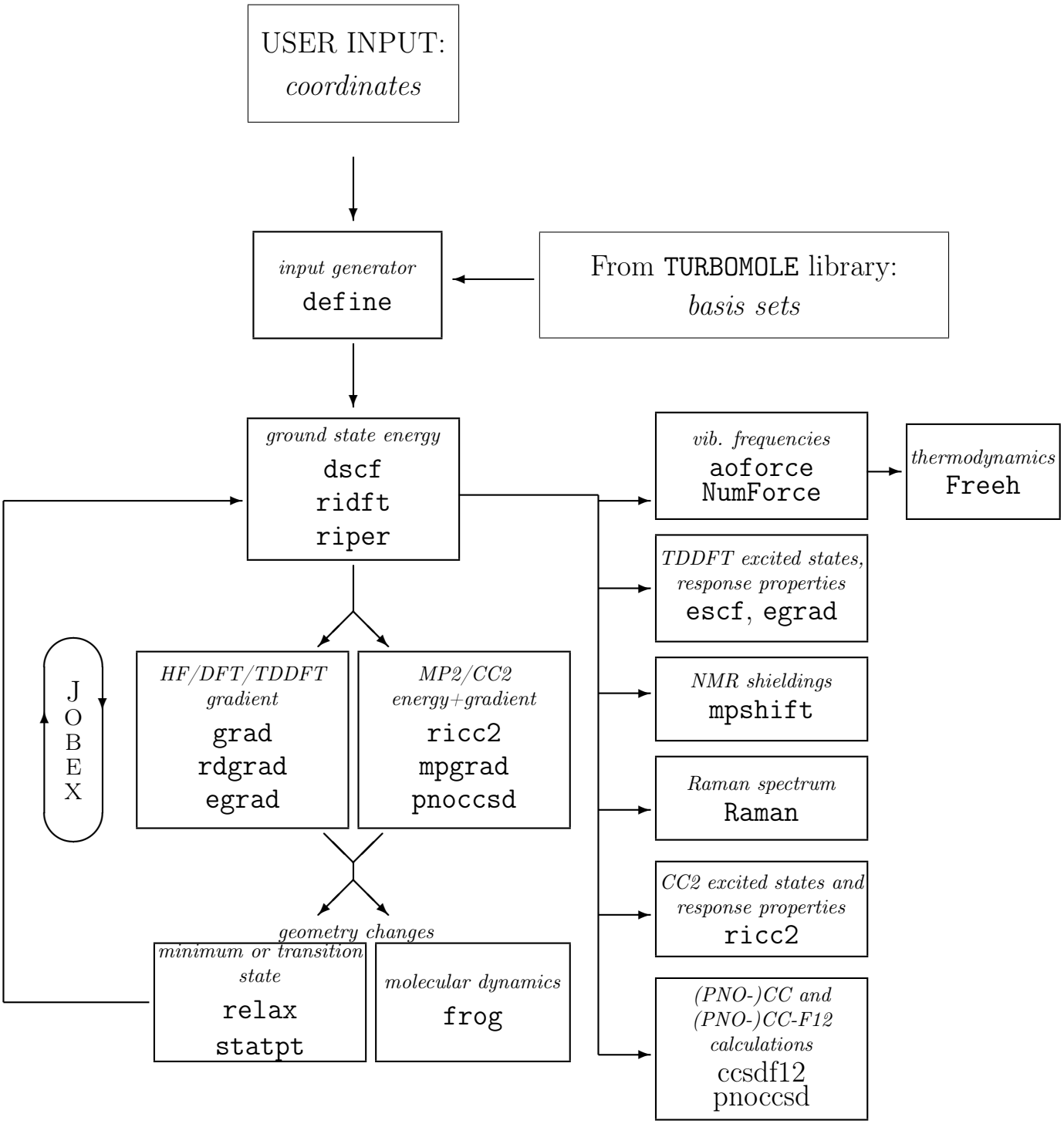

Figure 3.1: The modules of TURBOMOLE and the main data flow between them.

- `riper` — parallel ground state DFT energies for molecular and periodic systems. SMP only.
- `grad` — parallel ground state gradients from `dscf` calculations, SMP and MPI
- `ricc2` — parallel ground and excited state calculations of energies and gradients at MP2 and CC2 level using RI, as well as energy calculations of other wave function models, see chapter 10.7. SMP and MPI
- `ccsdf12` — parallel ground state energies beyond MP2/CC2 and excitation energies beyond CC2. SMP
- `pnoccsd` — parallel ground state energies at the OSV-PNO-MP2 and OSV-PNO-MP2-F12 level, SMP and MPI. For PNO-CCSD and PNO-CCSD(T) only the SMP version is available.
- `mpgrad` — parallel conventional (i.e. non-RI) MP2 energy and gradient calculations. Please note that RI-MP2 is one to two orders of magnitude faster than conventional MP2, so even serial RI-MP2 will be faster than parallel MP2 calculations. SMP and MPI
- `aoforce` — parallel Hartree-Fock and DFT analytic 2nd derivatives for vibrational frequencies, IR spectra, generation of Hessian for transition state searches and check for minimum structures. SMP and MPI
- `escf` — parallel TDDFT, RPA, CIS excited state calculations (UV-Vis and CD spectra, polarizabilities). SMP and MPI
- `egrad` — parallel TDDFT, RPA, CIS excited state analytic gradients, including polarizability derivatives for RAMAN spectra. SMP only.
- `mpshift` — parallel NMR shielding constants and chemical shifts. SMP only.
- `tb` — parallel GFN2-xTB energy and gradient calculations. SMP only.
- `NumForce` — this script can used for a trivial parallelization of the numerical displaced coordinates. SMP and MPI

See also [2]. Additional keywords necessary for parallel runs with the SMP or MPI binaries are not needed. When using the parallel version of `TURBOMOLE`, scripts are replacing the binaries. Those scripts prepare a usual input, run the necessary steps and automatically start the parallel programs. The users just have to set environment variables, see Sec. 3.4.1 below.

To use the OpenMP parallelization, only an environment variable needs to be set. But to use this parallelization efficiently one should consider a few additional points, e.g. memory usage, which are described in Sec. 3.4.2.

3.4.1 Running Parallel Jobs — MPI case

The parallel version of `TURBOMOLE` runs on all supported systems:

- workstation cluster with Ethernet, Infiniband, Myrinet (or other) connection
- SMP systems
- or combinations of SMP and cluster

Setting up the parallel MPI environment

In addition to the installation steps described in Section 2 (see page 42) you just have to set the variable `PARA_ARCH` to `MPI`, i.e. in `sh/bash/ksh` syntax:

```
export PARA_ARCH=MPI
```

This will cause `sysname` to append the string `_mpi` to the system name and the scripts like `jobex` will take the parallel binaries by default. To call the parallel versions of programs like `ridft`, `rdgrad`, `dscf`, `ricc2`, etc. from your command line without explicit path, expand your `$PATH` environment variable to:

```
export PATH=$TURBODIR/bin/‘sysname‘:$PATH
```

The usual binaries are replaced by scripts that prepare the input for a parallel run and start `mpirun` (or `poe` on IBM) automatically. The number of CPUs that shall be used can be chosen by setting the environment variable `PARNODES`:

```
export PARNODES=8
```

The default for `PARNODES` is 2.

Finally the user can set a default scratch directory that must be available on all nodes. Writing scratch files to local directories is highly recommended for I/O intensive modules like `ricc2`, otherwise the scratch files will be written over the network to the same directory where the input is located. The path to the local disk can be set with

```
export TURBOTMPDIR=/scratch/username
```

This setting is automatically recognized by most parallel programs. Note:

- For RI-DFT calculations it is usually neither necessary nor helpful to set TURBOTMPDIR.
- This does not set the path for the integral scratch files for `dscf` (see section below about `twoint` of keyword `$scfintunit`).

- In MPI parallel runs the programs attach to the name given in `$TURBOTMPDIR` node-specific extension (e.g. `/scratch/username-001`) to avoid clashes between processes that access the same file system. The jobs must have the permissions to create these directories. Therefore one must not set `$TURBOTMPDIR` to something like `/scratch` which would result in directory names like `/scratch-001` — which usually can not be created by jobs running under a standard user account.
  So please set the temporary directory for parallel files to a local file system at a position you are allowed to generate directories, like /tmp/myname/tmjobs/mpifiles

MPI versions, distributions and flavours

`TURBOMOLE` is using the MPI version which has been utilized to generate the binaries. To make sure that the parallel version is running, no matter which MPI flavour you have installed on your machines, `TURBOMOLE` does **include** the run-time version of the MPI flavour it needs.

> Please do not try to use `TURBOMOLE` with your local MPI version (OpenMPI, MPICH, ...)! Do not call the parallel MPI binaries directly, just set `$PARA_ARCH` as described above and call the modules the same way you use them in the serial version.

On Linux systems **Intel MPI** `Intel MPI` is used.

BIOVIA and `TURBOMOLE` GmbH ship `TURBOMOLE` with a runtime version Intel MPI. `TURBOMOLE` users do not have to install or license Intel MPI themselves. Parallel binaries will run out of the box on the fastest interconnect that is found - Infiniband, TCP/IP, etc.

Note: most parallel `TURBOMOLE` modules need an extra server running in addition to the clients. This server is included in the parallel binaries and it will be started automatically — but this results in one additional task that usually does not need any CPU time. So if you are setting `PARNODES` to `N`, `N+1` tasks will be started.

If you are using a queuing system or if you give a list of hosts where `TURBOMOLE` jobs shall run on (see below), make sure that the number of supplied nodes match `$PARNODES` — e.g. if you are using 4 CPUs via a queuing system, make sure that `$PARNODES` is set to 4.

Starting parallel jobs

After setting up the parallel environment as described in the previous section, parallel jobs can be started just like the serial ones. If the input is a serial one, it will be prepared automatically for the parallel run.

For the additional mandatory or optional input for parallel runs with the `ricc2` program see Section 10.7.

Running calculations on different nodes

If `TURBOMOLE` is supposed to run on a cluster, we highly recommend the usage of a queuing system like PBS, Univa/SGE GridEngine or LFS. The parallel version of `TURBOMOLE` will automatically recognise that it is started from within one of the queuing systems:

- PBS (Torque/Maui)
- LSF
- SLURM
- SGE or Univa Grid Engine

and the binaries will run on the machines those queuing systems provide.

Note: The default is to use secure shell (ssh) to start the jobs on the nodes. Please make sure that ssh works in a passwordless fashion on the cluster. If ssh is not possible, but *srun* is enabled, set `export MPI_USESRUN=1` to use srun instead of ssh.

Important: Make sure that the input files are located on a network directory like an NFS disk which can be accessed on all nodes that participate at the calculation.

If parallel jobs are started outside a queuing system, or if you have a non-supported or a non-default installation of above mentioned queuing systems, the number of nodes and their names can also be provided by the user. A file that contains a list of machines has to be created, each line containing one machine name:

```
node1
node1
node2
node3
node4
node4
```

And the environment variable `$HOSTS_FILE` has to be set to that file:

```
export HOSTS_FILE=/nfshome/username/hostsfile
```

Note: Do not forget to set `$PARNODES` to the number of lines in `$HOSTS_FILE`, unless you have set in addition OMP_NUM_THREADS (see below).

Note: In general the stack size limit has to be raised to a reasonable amount of the memory (or to unlimited). In the serial version the user can set this by `ulimit -s unlimited` on bash/sh/ksh shells or `limit stacksize unlimited` on

csh/tcsh shells. However, for the parallel version that is not sufficient if several nodes are used, and the `/etc/security/limits.conf` files on all nodes might have to be changed. See chapter 2.2 of this documentation, page 45

OpenMP/MPI hybrid version

Some `TURBOMOLE` modules like `dscf`, `grad`, `aoforce`, `ricc2`, `escf` or `pnoccsd` are parallelized using a hybrid OpenMP/MPI scheme. For those modules it is sufficient to start just one single process per node. In addition, please set

`export OMP_NUM_THREADS=`*<number of cores per node>*

when starting the job. This environment variable will be exported to each node such that the processes started there will open *<number of cores per node>* threads.

Memory for parallel jobs

Since there are several different parallel versions of the individual `TURBOMOLE` modules available, the meaning of the keywords to set memory (`$ricore` and `$maxcor`) can be quite confusing. A lot of problems can be avoided if following points are taken care of:

- for almost all cases increasing `$ricore` will not speed up the calculation but increase memory consumption significantly. It is therefore recommended to set `$ricore` to a small value like 100 or 500. Except:
- usage of RI-JK does benefit from large `$ricore` values. Check if `$rik` is present in your control file — and if yes, try to increase the memory to a value which your machines are capable to handle.
- many post-SCF programs read and use `$maxcor`. There are a couple of options to this keyword like `per_proc` to define the amount of memory per process or `per_node` for memory settings on one node, etc. A detailed description can be found in chapter 23.2.3 on page 414.
- if you worry about speed in RI-DFT calculations, make sure to have `$marij` in your control file

Testing the parallel binaries

The binaries `ridft`, `rdgrad`, `dscf`, `grad`, and `ricc2` can be tested by the usual test suite: go to `$TURBODIR/TURBOTEST` and call `TTEST`

Note: Some of the tests are very small and will only pass properly if 2 CPUs are used at maximum. Therefore `TTEST` will not run any test if `$PARNODES` is set to a higher value than 2.

If you want to run some of the larger tests with more CPUs, you have to edit the `DEFCRIT` file in `TURBOMOLE/TURBOTEST` and change the `$defmaxnodes` option.

Sending additional server task to sleep

Except for the MPI parallel binaries of `ridft` and `rdgrad` all modules start an additional server process which is not participating in the calculation itself, but just responsible for the task distribution. This server task is waiting for the processes to ask for new tasks in a way that the response time is as low as possible. This leads to a noticeable CPU usage.

To send the server task to sleep while waiting for communication, it is possible to set the environment variable `TM_SERVERSLEEP` either to a value in microseconds or to a string like 'on' to use the default of 500 microseconds:

```
export TM_SERVERSLEEP=1000
#or
export TM_SERVERSLEEP=on
```

This is only reasonable if you use relatively few CPUs for a calculation. The larger the number of processes, the more important is a timely reply from the server.

Sample simple PBS start script

```
#!/bin/sh
# Name of your run :
#PBS -N turbomole
#
# Number of nodes to run on:
#PBS -l nodes=4
#
# Export environment:
#PBS -V

# Set your TURBOMOLE pathes:

######## ENTER YOUR TURBOMOLE INSTALLATION PATH HERE ##########
export TURBODIR=/whereis/TURBOMOLE
###############################################################

export PATH=$TURBODIR/scripts:$PATH

## set locale to C
unset LANG
unset LC_CTYPE
```

```
# set stack size limit to unlimited:
ulimit -s unlimited

# Count the number of nodes
PBS_L_NODENUMBER=‘wc -l < $PBS_NODEFILE‘

# Check if this is a parallel job
if [ $PBS_L_NODENUMBER -gt 1 ]; then
##### Parallel job
# Set environment variables for a MPI job
    export PARA_ARCH=MPI
    export PATH="${TURBODIR}/bin/‘sysname‘:${PATH}"
    export PARNODES=‘expr $PBS_L_NODENUMBER‘
else
##### Sequentiel job
# set the PATH for Turbomole calculations
    export PATH="${TURBODIR}/bin/‘sysname‘:${PATH}"
fi

#VERY important is to tell PBS to change directory to where
#     the input files are:

cd $PBS_O_WORKDIR

######## ENTER YOUR JOB HERE ###################################
jobex -ri > jobex.out
################################################################
```

3.4.2 Running Parallel Jobs — SMP case

The SMP version of `TURBOMOLE` currently combines three different parallelization schemes which all use shared memory:

- `dscf`, `grad`, `ridft`, `rdgrad`, `aoforce`, `escf`, `egrad`, `ricc2`, `ccsdf12`, `pnoccsd`, `mpshift`, `evib`, `odft`, `rirpa` and `riper` are partially parallelized with OpenMP for applications on shared-memory, in particular multi-CPU and multi-core, machines.
- `aoforce`, `escf`, `egrad`, `ridft` and `rdgrad` are also parallelized as described in [46]
- `ridft` and `rdgrad` are parallelized with MPI using shared memory on SMP systems. This is also the default version for SMP systems, not just for MPI runs.

Setting up the parallel SMP environment

In addition to the installation steps described in Section 2 (see page 42) you just have to set the variable `PARA_ARCH` to `SMP`, i.e. in `sh/bash/ksh` syntax:

```
export PARA_ARCH=SMP
```

This will cause `sysname` to append the string `_smp` to the system name and the scripts like `jobex` will take the parallel binaries by default. To call the parallel versions of the programs (like `ridft` or `aoforce`) from your command line without explicit path, expand your `$PATH` environment variable to:

```
export PATH=$TURBODIR/bin/‘sysname‘:$PATH
```

The usual binaries are replaced now by scripts that prepare the input for a parallel run and start the job automatically. The number of CPUs that shall be used can be chosen by setting the environment variable `PARNODES`:

```
export PARNODES=8
```

The default for `PARNODES` is 2.

NOTE: Depending on what you are going to run, some care has to be taken that the system settings like memory limits, etc. will not prevent the parallel versions to run. See the following sections.

OpenMP parallelization of almost all time consuming modules

The OpenMP parallelization does not need any special program startup. The binaries can be invoked in exactly the same manner as for sequential (non-parallel) calculations. Just set the environment variable `PARNODES` to the number or threads that should be used by the programs. The scripts will set `OMP_NUM_THREADS` to the same value and start the OpenMP binaries directly. The number of threads is essentially the max. number of CPU cores the program will try to utilize. To exploit e.g. all eight cores of a machine with two quad-core CPUs set

```
export PARNODES=8
```

(for csh and tcsh use `setenv PARNODES=8`).

Presently the OpenMP parallelization of `ricc2` comprises all functionalities apart from the $\mathcal{O}(\mathcal{N}^4)$-scaling LT-SOS-RI functionalities (which are only parallelized with MPI) and expectation values for $\hat{S}^2$ (not parallelized). Note that the memory specified with `$maxcor` is for OpenMP-parallel calculation the maximum amount of memory that will be dynamically allocated by all threads together. To use your computational resources efficiently, it is recommended to set this value to about 75% of the physical memory available for your calculations.

For Localized Hartree-Fock calculations please use the `dscf` program which is parallelized using OpenMP. In this case an almost ideal speedup is obtained because the most expensive part of the calculation is the evaluation of the Fock matrix and of the Slater-potential, and both of them are well parallelized. The calculation of the correction-term of the grid will use a single thread.

The OpenMP parallelization of `riper` covers all contributions to the Kohn-Sham matrix and nuclear gradient. Hence an almost ideal speedup is obtained.

To use the OpenMP version of `ridft` and `rdgrad` instead of the default parallelization on SMP machines, just set

```
export TM_PAR_OMP=on
```

and start the jobs the usual way. Some features like the semi-numerical exchange (keyword `$senex`, see section 23.2.10, two-component difference densities, periodic embedding, COSMO, coulex) are parallelized with OpenMP only. Moreover, OpenMP can have other benefits like the amount of hardware resources used.

Restrictions:

- In the `ricc2` program the parts related to RI-MP2-F12, LT-SOS-RI-MP2 or calculation of expectation values for $\hat{S}^2$ do not (yet) use OpenMP parallelization. If the OpenMP parallelization is switched on (by setting `OMP_NUM_THREADS`) these parts will still be executed sequentially.
- In the `dscf` program the `$incore` option will be ignored if more than one thread is used. Semi-direct `dscf` calculations (i.e. if a size larger than 0 is given two-electron integral scratch file in `$scfintunit`) can not be combined with the OpenMP parallel runs. (The program will then stop with error message in the first Fock matrix construction.)

Multi-thread parallelization of `dscf`, `grad`, `aoforce`, `escf`, `egrad`, `ridft` and `rdgrad`

The parallelization of those modules is described in [46] and is based on `fork()` and Unix sockets. Except setting `PARNODES` which triggers the environment variable `SMPCPUS`, the environment variable

```
export TM_PAR_FORK=on
```

has to be set. Alternatively, the binaries can be called with `-smpcpus <N>` command line option or with the keyword `$smp_cpus` in the control file.

The efficiency of the parallelization is usually similar to the default version, but for `ridft` and `rdgrad` RI-K is not parallelized. If density convergence criteria (`$denconv`) is switched on using `ridft` and if no RI-K is being used, the multi-threaded version should be used.

SMP/MPI version of `ridft` and `rdgrad`

Since `TURBOMOLE` version 7.2 the usage of GlobalArrays has been omitted. Instead, a set of routines which utilize shared memory on a node has been implemented. Both modules, `ridft` and `rdgrad`, start each process as an individual MPI instance. Processes on the same node are then collected to collectively store and use data in a shared memory region. This avoids excessive memory usage and reduces the amount of memory requirements significantly, especially compared to the old MPI implementation (which has been used by default in former `TURBOMOLE` versions). It is nevertheless recommended to

- run jobs in parallel only if the molecules and/or basis set size is large enough — several hundred basis functions for non-hybrid functionals, or few hundred for hybrid functional calculations. The RI approximation in combination with MARI-J is already very fast in the serial version, usage of many cores would introduce a communication overhead which exceeds the computational time on a single core.
- not ask for too much memory. For medium sized to large molecules adding a significant amount of `$ricore` will not speed up the calculation, but eventually lead to overstressed systems (or even memory swapping) or failure of jobs due to too large memory requirements. So please do not set large `$ricore` values in such cases, a few ten or hundred MB are sufficient (even zero works equally well).

SMP version with GPU offloading

Since `TURBOMOLE` version 7.7 some DFT modules are able to offload work to GPUs under Linux. Details and system requirements are given in the README-GPU.txt file of the `TURBOMOLE` installation.

Chapter 4

Preparing your input file with DEFINE

`define` is the general interactive input generator of `TURBOMOLE`. During a session with `define`, you will create the `control` file which controls the actions of all other `TURBOMOLE` programs. Starting with Version 7.5, the syntax of `control` (namely the `$atoms` data group) has changed and older versions of `TURBOMOLE` are no longer able to read the new `control` file. The old syntax can be enforced by starting `define` with the command line option `-old`. New versions of `TURBOMOLE` are able to handle both the old and the new syntax.

During your `define` session you will be guided through four main menus:

1. **The geometry main menu**: This first menu allows you to build your molecule, define internal coordinates for geometry optimizations, determine the point group symmetry of the molecule, adjust internal coordinates to the desired values and related operations. Beyond this one can perform a geometry optimization at a force field level to preoptimize the geometry and calculate a Cartesian analytical Hessian. After leaving this menu, your molecule to be calculated should be fully specified.

2. **The atomic attributes menu**: Here you will have to assign basis sets and/or effective core potentials to all atoms. The SV(P) basis is assigned automatically as default, as well as ECPs (small core) beyond Kr.

3. **The occupation numbers and start vectors menu**: In this menu you should choose `eht` to start from Extended Hückel MO vectors. Then you have to define the number of occupied orbitals in each irreducible representation.

4. **The general menu**: The last menu manages a lot of control parameters for all `TURBOMOLE` programs.

Most of the menu commands are self-explanatory and will only be discussed briefly. Typing `*` (or `q`) terminates the current menu, writes data to `control` and leads to the next while typing `&` goes back to the previous menu.

4.0.3 Universally Available Display Commands in DEFINE

There are some commands which may be used at (almost) every stage of your `define` session. If you build up a complicated molecular geometry, you will find the `dis` command useful. It will bring you to the following little submenu:

```
ANY COMMAND WHICH STARTS WITH THE 3 LETTERS  dis  IS A
DISPLAY COMMAND. AVAILABLE DISPLAY COMMANDS ARE :
disc  <range> : DISPLAY CARTESIAN COORDINATES
dist  <real>  : DISPLAY DISTANCE LIST
disb  <range> : DISPLAY BONDING INFORMATION
disa  <range> : DISPLAY BOND ANGLE INFORMATION
disi  <range> : DISPLAY VALUES OF INTERNAL COORDINATES
disg  <range> : GRAPHICAL DISPLAY OF MOL. GEOMETRY
<range> IS A SET OF ATOMS REFERENCED
<real>  IS AN OPTIONAL DISTANCE THRESHOLD (DEFAULT=5.0)
AS AN EXAMPLE CONSIDER  disc  1,3-6,10,11  WHICH DISPLAYS
THE CARTESIAN COORDINATES OF ATOMS 1,3,4,5,6,10,and 11 .
HIT >return< TO CONTINUE OR ENTER ANY DISPLAY COMMAND
```

Of course, you may enter each of these display commands directly without entering the general command `dis` before. The option `disg` needs special adaption to the computational environment, however, and will normally not be available.

4.0.4 Specifying Atomic Sets

For many commands in `define` you will have to specify a set of atoms on which that command shall act. There are three ways to do that:

- You may enter `all` or `none`, the meaning of which should be clear (entering `none` makes not much sense in most cases, however).
- You may specify a list of atomic indices like `1` or `3,5,6` or `2,4-6,7,8-10` or similar.
- You may also enter atomic identifiers which means strings of at most eight characters: the first two contain the element symbol and the remaining six could be used to distinguish different atoms of the same type. For example, if you have several carbon atoms in your molecule, you could label some `c ring` and others `c chain` to distinguish them. Whenever you want to enter an *atomic identifier*, you have to put it in double quotation marks: `"c ring"`.

You should take into account that `define` also creates, from the atoms you entered, all others according to symmetry. If necessary, you will therefore have to lower the (formal) symmetry before executing a command.

4.0.5 `control` as Input and Output File

`define` may be used to update an existing `control` file, which is helpful if only the basis set has been changed. In this case just keep all data, i.e. reply with `<enter>` on

all questions, and only specify new start MOs. The more general usage is described now.

At the beginning of each `define` session, you will be asked to enter the name of the file to be created. As mentioned earlier, all `TURBOMOLE` programs require their input to be on a file named `control`, but it may be useful at this moment to choose another name for this file (e.g. if you have an old input file `control` and you do not want to overwrite it). Next you will be asked to enter the name of an *old* file which you want to use as input for this session. This prevents you from creating the new input from scratch if you want to make only minor changes to an old `control` file. It is possible to use the same file as input and output file during a `define` session (which means that it will only be modified). This may lead to difficulties, however, because `define` reads from the input file when entering each main menu and writes the corresponding data when leaving this menu. Therefore the input file may be in an ill-defined status for the next main menu (this will be the case, for example, if you add or change atoms in the first menu so that the basis set information is wrong in the second menu). `define` takes care of most—but not all—of these problems.

For these reasons, it is recommended to use a different filename for the input and the output file of the `define` session if you change the molecule to be investigated. In most cases involving only changes in the last three of the four main menus no problem should arise when using the same file as input and output.

4.0.6 Be Prepared

Atomic Coordinates

Molecules and their structures are specified by coordinates of its atoms, within the program invariably by Cartesian coordinates in atomic units (Ångstrøm would also do). In `TURBOMOLE` these coordinates are contained in the file `coord` (see Section 24 "Sample `control` files" for an example).

Recommendation

We strongly recommend to create the `coord` file *before* calling `define`, only for small molecules one should use the interactive input feature of `define`. Set up the molecule by any program you like and write out coordinates in the xyz-format (`XMol` format), which is supported by most programs. Then use the `TURBOMOLE` tool `x2t` to convert it into a `TURBOMOLE` `coord` file (see Section 1.5.

Internal Coordinates

Structure optimizations, see `jobex`, are most efficient if carried out in internal coordinates and `TURBOMOLE` offers the following choices.

`internals` based on bond distances and angles, see Section 4.1.2.

`redundant internals`
: defined as linearly independent combinations of *internals* (see ref. [47]), provided automatically by the command `ired` in the 'geometry main menu' in Section 4.1 below. This works in almost all cases and is efficient. The disadvantage is, that this is a black box procedure, the coordinates employed have no direct meaning and cannot be modified easily by the user.

`cartesians`
: should always work but are inefficient (more cycles needed for convergence). Cartesians are the last resort if other options fail, they are assigned as default if one leaves the main geometry menu and no other internals have been defined.

4.1 The Geometry Main Menu

After some preliminaries providing the title etc. you reach the geometry main menu:

```
SPECIFICATION OF MOLECULAR GEOMETRY ( #ATOMS=0     SYMMETRY=c1  )
YOU MAY USE ONE OF THE FOLLOWING COMMANDS :
sy <group> <eps> : DEFINE MOLECULAR SYMMETRY (default for eps=3d-1)
desy <eps>       : DETERMINE MOLECULAR SYMMETRY AND ADJUST
                     COORDINATES (default for eps=1d-6)
syndi <eps>      : LIKE DESY, BUT FIND ONLY GROUPS WITH NON-
                     DEGENERATE IRREPS (D2h AND SUBGROUPS)
susy             : ADJUST COORDINATES FOR SUBGROUPS
ai               : ADD ATOMIC COORDINATES INTERACTIVELY
a <file>         : ADD ATOMIC COORDINATES FROM FILE <file>
aa <file>        : ADD ATOMIC COORDINATES IN ANGSTROEM UNITS FROM FILE <file>
sub              : SUBSTITUTE AN ATOM BY A GROUP OF ATOMS
i                : INTERNAL COORDINATE MENU
ired             : REDUNDANT INTERNAL COORDINATES
red_info         : DISPLAY REDUNDANT INTERNAL COORDINATES
ff               : UFF-FORCEFIELD CALCULATION
m                : MANIPULATE GEOMETRY
frag             : DEFINE FRAGMENTS FOR BSSE CALCULATION
w <file>         : WRITE MOLECULAR COORDINATES TO FILE <file>
r <file>         : RELOAD ATOMIC AND INTERNAL COORDINATES FROM FILE <file>
name             : CHANGE ATOMIC IDENTIFIERS
del              : DELETE ATOMS
dis              : DISPLAY MOLECULAR GEOMETRY
banal            : CARRY OUT BOND ANALYSIS
*                : TERMINATE MOLECULAR GEOMETRY SPECIFICATION
                     AND WRITE GEOMETRY DATA TO CONTROL FILE

IF YOU APPEND A QUESTION MARK TO ANY COMMAND AN EXPLANATION
OF THAT COMMAND MAY BE GIVEN
```

This menu allows you to build your molecule by defining the Cartesian coordinates

interactively (`ai`) or by reading the coordinates from an external file (`a`, `aa`). The structure can be manipulated by the commands `sub`, `m`, `name` and `del`. The command `sy` allows you to define the molecular symmetry while `desy` and `syndi` try to determine automatically the symmetry group of a given molecule.

There exists a structure library which contains the Cartesian coordinates of selected molecules, e.g. CH_4. These data can be obtained by typing for example `a ! ch4` or `a ! methane`. The data files are to be found in the directory `$TURBODIR/structures`. The library can be extended.

You can perform a geometry optimization at a force field level to preoptimize the geometry. For this purpose the Universal Force Field (UFF) developed from Rappé et al. in 1992 [10] has been implemented in the `uff` module (see also Section 5.4). This can also be used to calculate an analytical approximate Cartesian Hessian. If one does so, the start Hessian for the *ab initio* geometry optimization is this Hessian instead of the diagonal one (`$forceinit on carthess` for `relax` module).

Recommendation

Here is an easy way to get internal coordinates, which should work.

Have `coord` ready before calling `define`. In the main geometry menu proceed as follows to define *redundant internals*:

`a coord`	read `coord`
`desy`	determine symmetry, if you expect a higher symmetry, repeat with increased tolerance `desy 0.1` , you may go up to `desy 1.`.
`ired`	get redundant internals
`*`	quit main geometry menu

To define *internals*:

`a coord`	read `coord`
`desy`	determine symmetry
`i`	go to internal coordinate menu
`iaut`	automatic assignment of bends etc.
`q`	to quit bond analysis
`imet`	to get the metric, unnecessary internals are marked `d` now. If `#ideg = #k` in the head line you are done. Otherwise this did not work.
`<enter>`	go back to main geometry menu
`*`	quit main geometry menu

To define *cartesians*:

`a coord`	read `coord`
`desy`	determine symmetry
`*`	quit main geometry menu

4.1.1 Description of commands

Main Geometry Menu

In the headline of this menu you can see the current number of atoms and molecular symmetry (we use an input for PH_3 as example). The commands in this menu will now be described briefly:

`sy` Definition of the Schönflies symbol of the molecular point group symmetry. If you enter only `sy`, `define` will ask you to enter the symbol, but you may also directly enter `sy c3v`. `define` will symmetrize the geometry according to the new Schönflies symbol and **will create new nuclei if necessary**. You therefore have to **take care** that you enter the correct symbol and **that your molecule is properly oriented**. All `TURBOMOLE` programs require the molecule to be in a standard orientation depending on its point group. For the groups C_n, C_{nv}, C_{nh}, D_n, D_{nh} and D_{nd} the z-axis has to be the main rotational axis, secondary (twofold) rotational axis is always the x-axis, σ_v is always the xz-plane and σ_h the xy-plane. O_h is oriented as D_{4h}. For T_d, the threefold rotational axis points in direction (1,1,1) and the z-axis is one of the twofold axes bisecting one vertex of the tetrahedron.

`desy` `desy` allows you to determine the molecular symmetry automatically. The geometry does not need to be perfectly symmetric for this command to work. If there are small deviations from some point group symmetry (as they occur in experimentally determined structures), `desy` will recognize the higher symmetry and symmetrize the molecule properly. If symmetry is lower than expected, use a larger threshold: `<eps>` up to 1.0 is possible.

`syndi` `syndi` works just like `desy`, but only recognizes point groups with non-reducible irreducible representations (that is D_{2h} and its subgroups). This is designed for some features of `TURBOMOLE` which can only handle such point groups. Note that the orientation of the molecule might be different from what you expect. For example, the xy plane will be a mirror plane of NH_3 (which is assigned C_s symmetry).

`susy` `susy` leads you through the complete subgroup structure if you want to lower symmetry, e.g. to investigate Jahn–Teller distortions. The molecule

is automatically reoriented if necessary.
Example: $T_d \rightarrow D_{2d} \rightarrow C_{2v} \rightarrow C_s$.

`ai` You may enter Cartesian atomic coordinates and atomic symbols interactively. After entering an atomic symbol, you will be asked for Cartesian coordinates for this type of atom until you enter `*`. If you enter `&`, the atom counter will be decremented and you may re-define the last atom (but you surely won't make mistakes, will you?). After entering `*`, `define` asks for the next atom type. Entering `&` here will allow you to re-define the last atom type and `*` to leave this mode and return to the geometry main menu. Enter `q` as atom symbol if you want to use a dummy center without nuclear charge. Symmetry equivalent atoms are created immediately after you entered a set of coordinates.

This is a convenient tool to provide e.g. rings: exploit symmetry group D_{nh} to create an n-membered planar ring by putting an atom on the x-axis.

`a` *file* You may also read atomic coordinates (and possibly internal coordinates) from *file*, where *file* must have the same format as the data group `$coord` in file `control`.

The Cartesian coordinates and the definitions of the internal coordinates are read in free format; you only have to care for the keywords `$coord` and (optionally) `$intdef` and (important!) for the `$end` at the end of the file. The atomic symbol follows the Cartesian coordinates separated by (at least) one blank. For a description of the internal coordinate definitions refer to 4.1.2.

Entering '`!`' as first character of *file* will tell `define` to take *file* from the structure library. (The name following the '`!`' actually does not need to be a filename in this case but rather a search string referenced in the structure library contents file, see Section 4.1).

`aa` *file* same as `a`, but assumes the atomic coordinates to be in Å rather than a.u.

`sub` This command allows you to replace one atom in your molecule by another molecule. For example, if you have methane and you want to create ethane, you could just substitute one hydrogen atom by another methane molecule. The only requirement to be met by the substituted atom is that it must have exactly one bond partner. The substituting molecule must have an atom at the substituting site; in the example above it would not be appropriate to use CH_3 instead of CH_4 for substitution. Upon substitution, two atoms will be deleted and the two ones forming the new bond will be put to a standard distance. `define` will then ask you to specify a dihedral angle between the old and the new unit. It is also possible to use a part of your molecule as substituting unit, e.g. if you have some methyl groups in your molecule, you can create further ones

by substitution. Some attention is required for the specification of this substituting unit, because you have to specify the atom which will be deleted upon bond formation, too. If you enter the filename from which the structure is to be read starting with '`!`', the file will be taken from the structure library (see Section 4.1). Definitions of internal coordinates will be adjusted after substitution, but no new internal coordinates are created.

`i` This command offers a submenu which contains everything related to internal coordinates. It is further described in Section 4.1.2.

`m` This command offers a submenu which allows you to manipulate the molecular geometry, i.e. to move and rotate the molecule or parts of it. It is further described in Section 4.1.3.

`frag` Here, the fragments will be defined as being used by the `jobbsse` script in order to do a calculation using the counter-poise correction scheme. In this menu, up to three monomers can be defined, together with their charges and their symmetry. When assigning atom numbers to fragments, if x is entered instead of a number, the program will request the first and last atoms of a range. This will be useful for very large fragments.

`w` *file* The command `w` writes your molecular geometry and your internal coordinates to *file*. Afterwards you will be back in the geometry main menu. If the filename entered starts with '`!`', the structure will be written to the structure library.

`name` `name` allows you to change atomic identifiers turning, e.g. oxygen atoms into sulfur atoms. After entering the identifier to be changed (remember the double quotation marks : `"c ring"`), you will be asked to enter the new one. You can use question marks for characters not to be changed, e.g. you enter `"??ring"` to change `c chain` to `c ring`. If you do not enter eight characters, your input will be filled up with trailing blanks.

`del` The command `del` allows you to delete one or more atoms. After you entered the atomic list, `define` will show you a list of all atoms concerned and will ask you to confirm deleting these atoms. If any internal coordinate definitions exist, which rely on some of the deleted atoms, these definitions will be deleted, too.

`banal` The command `banal` allows you to perform a bonding analysis, that is, `define` will try to decide which atoms are bonded and which are not (according to a table of standard bond lengths which is included in the code of `define`). You must have performed this command before you can use the display commands `disb` (display bonding information) or `disa` (display bond angle information). The standard bond lengths (and the bonding analysis available from these) are also needed for the

commands `sub` and `iaut` (see internal coordinate menu, Section 4.1.2). If you want to change the standard bond lengths (or define more bond lengths, because not for all possible combinations of elements a standard length is available) you can do that by creating your own file with the non-default values and by specifying its full path name in file `.sys.data`. The file has the following simple format:

```
c - h  2.2
h - h  2.0
. - .  ...
```

The format of the entries is almost arbitrary: the two element symbols have to be separated by a bar, the new bond distance follows in free format (in atomic units). If the file cannot be read properly, a warning message is displayed.

`*` This command leaves this first main menu and writes all data generated so far to file. The default output file is the file you choose in the first question during your `define` session (usually `control`). Now the data groups `$coord` and `$intdef` will be written to file. After leaving this menu, you will enter the atomic attributes menu, which is described in Section 4.2.

4.1.2 Internal Coordinate Menu

```
INTERNAL COORDINATE MENU  ( #ideg=6       #k=2    #f=0    #d=0    #i=0 )

imet <a>  : PROVIDE B-MATRIX FOR ACTIVE INTERNAL COORDINATES
            (CHECK COMPLETENESS AND NUMERICAL QUALITY
            AND CHANGE REDUNDANT INTERNALS TO display)
idef      : SUB-MENU FOR INTERACTIVE DEFINITION OF INTERNAL COORDINATES
ideg <a>  : OUTPUT NUMBER OF TOT. SYMMETRIC INTERNAL DEGREES OF FREEDOM
iaut      : TRY AUTOMATIC DEFINITION OF INTERNAL COORDINATES
iman <a>  : MANIPULATE GEOMETRY BY CHANGING INTERNAL COORDINATE VALUES
imanat <i>: AS iman BUT STARTING AT INTERNAL COORD. NUMBER i
ic <i> <x>: CHANGE STATUS OF INTERNAL COORDINATE <i> TO <x>
            e.g. ic 5 d TO MAKE 5TH COORD. display OR ic k d
irem <i>  : REMOVE INTERNAL COORDINATE <i>,
            e.g. irem d TO REMOVE ALL display COORDS
dis       : ANY DISPLAY COMMAND e.g. disi OR disc
disiat <i>: AS disi BUT STARTING AT INTERNAL COORD. NUMBER i

WHERE  <a>= OPTIONAL ATOMIC SET (DEFAULT=all)
       <i>= INDEX(LIST) OF INTERNAL COORDINATE(S) LIKE 3-6,8 OR <i>=<x>
       <x>= STATUS OF INTERNAL COORDINATE = k, f, d OR i
ADDING A QUESTION MARK TO ANY COMMAND MAY PROVIDE EXPLANATIONS

ENTER COMMAND OR HIT >return< TO GET BACK TO GEOMETRY MAIN MENU
```

The parameters in the headline of this menu have the following meanings:

`#ideg` is the total number of symmetry restricted degrees of freedom.

`#k` is the number of *active* internal coordinates specified up to now. Only these coordinates are optimized during a geometry optimization.

`#f` is the number of *fixed* internal coordinates specified. These coordinates will be included in the **B**-matrix (see command `imet`), but their values will not be changed during geometry optimization.

`#d` is the number of internal coordinates whose values will only be displayed (e.g. by command `disi`), but no gradients will be calculated for these coordinates nor will they be included in the geometry optimization.

`#i` means the number of coordinates which are defined, but will be completely ignored, i.e. they are not even displayed on the screen and will not be used by any program (this is the waste-paper-basket of `define`).

Note that the `#k` plus `#f` must equal the number of degrees of freedom (`#ideg`) of your molecule, if you want to perform a geometry optimization. If you have less coordinates than degrees of freedom, you will have to specify further ones (commands `idef` or `iaut`, see below); if you have more coordinates than degrees of freedom, you will have to throw away some of them (commands `irem` or `imet`, see below).

The commands in this menu allow you to define internal coordinates for your molecule, adjust your geometry to special values of these internal coordinates and to control the numeric reliability of the chosen set of internal coordinates. In detail, the commands act as follows.

Description of commands

`imet` *a* This command computes the so-called **B**-matrix, which is the matrix of the derivatives of the (*active* and *fixed*) internal coordinates with respect to Cartesian coordinates. This matrix is used in program `relax` for the conversion from Cartesian coordinates and gradients to internal ones and vice versa. If this matrix is singular (or even nearly singular) this indicates a linear dependency of your internal coordinate set. As a consequence the geometry update (more exactly the transformation of the updated internal coordinates into Cartesian ones) will fail. This may also happen in the course of a geometry optimization if the coordinates run into linear dependency during their optimization. `imet` checks the **B**-matrix and removes linear dependent internal coordinates from your list (their status is changed from `#k` or `#f` to `#d`). If **B** is only near singular, a warning is issued and the lowest eigenvalue(s) as well as the corresponding eigenvector(s) are displayed. In this case, you should try to find better internal coordinates (although this may not always be possible). After the command `imet`, there may be too few (active plus

fixed) internal coordinates, but certainly not too many (because linear dependencies have been eliminated). Perhaps you will have to add new ones or—better!—try command `iaut` or `ired` in the preceding menu.

Command `imet` should be used always after creating internal coordinates with `iaut` or `idef` (especially after `iaut`, because this command creates usually an overcomplete set of internal coordinates).

`idef` `idef` unfolds a little submenu where you can define internal coordinates manually. The exact procedure of the definition will be described below in a separate section.

`ideg` *a* This command gives you the number of symmetry restricted degrees of freedom (for the atomic set specified by *a*). Without symmetry, this is just $3N - 6$, where N is the number of atoms, but if there is symmetry, some of these degrees of freedom will violate symmetry and therefore are not valid. For geometry optimizations, only the symmetry allowed degrees of freedom are needed, because the symmetry requirements are imposed anyway. In connection with the optional atomic set *a* this command can help you to find out, in which part of a complicated molecule internal coordinates are missing, if you fail to get the full number of `#ideg` (which equals the result of `ideg all`) for the molecule as a whole.

`iaut` `iaut` tries an automatic definition of internal coordinates. This command relies on an recursive procedure which tries to simplify the molecule as far as possible and then starts the definition of internal coordinates. At present not all molecular topologies are supported, therefore it may happen that no internal coordinates can be assigned to your molecule or at least a part of it. However, for all cases in which an automatic assignment of coordinates is possible, `iaut` has up to now proved to provide very good internal coordinates. If `iaut` works for your molecule (and in most non-pathological cases it will) we recommend strongly to use these coordinates, as they may help you to save several cycles in the geometry optimization procedure. After creating internal coordinates with `iaut` you should always use `imet` (see above), because `iaut` may provide an overcomplete set of coordinates. All coordinates which conflict with the molecular symmetry are set to *ignore* by `iaut`.

`iman` *a* `iman` allows you to modify the values of internal coordinates. If you specify a list of atoms *a* only those internal coordinates which refer to only these atoms will be handled. You will get a list of all (active and fixed) internal coordinates and their current values and you will be able to enter a new value for each of them if you like. Default (`<enter>`) keeps the value shown. Be aware that all distances are given in atomic units (1 a.u. = 52.9 pm).

`ic` *i x* This option allows you to change the status of a coordinate, e.g. from *active* to *display* or every other combination. The syntax is `ic 5 d`, if

coordinate no. 5 is to be set to *display*, or `ic k d`, if all active coordinates are to be set to *display*.

`irem` *i* This option allows you to delete definitions of internal coordinates from your list. The indices of the internal coordinates always refer to the full list of coordinates including *display* and *ignore* coordinates. To make sure you delete the right ones, use `disi`. Also the indices will immediately change if you delete coordinates. If you want to delete several coordinates, this is therefore done most easily if you delete them in order of descending indices (because deletion of a coordinate has only an effect on the coordinates with higher indices). After choosing the coordinates to be deleted, a list of all coordinates involved will be displayed and you will be asked to confirm deletion.

The syntax is simply `irem 5` to delete internal coordinate no. 5 or `irem d` to remove all 'display' coordinates.

Hitting `<return>` will bring you back to the geometry main menu.

Interactive Definition of Internal Coordinates

If you choose `idef` in the internal coordinate menu, you will get the following information:

```
ENTER INTERNAL COORDINATE DEFINITION COMMAND
                <x> <type> <indices>
WHERE    <x>       = k    f    d    i
         <type>    = stre invr bend outp tors linc linp
                     comp ring pyrm bipy pris cube octa
THESE COMMANDS WILL BE EXPLAINED IN DETAIL IF YOU ENTER
<x> <type>? FOR SOME CHOICE OF <x> AND <type>, E.G. k stre ?
DEFAULT=GO BACK TO INTERNAL MAIN MENU    DISPLAY=dis
```

The `<x>` means the status (see page 75) of the internal coordinate entered (`k`, `f`, `d`, `i`). The syntax is:

```
k stre 1 2
d tors 3 6 2 7
f bend 3 4 5
i outp 3 4 7 9
```

Note that in the third example atom 5 is the *central atom* of the angle!

Specification of available internal coordinates

The following types of coordinates are available:

`stre` The `stre` (for **stre**tch) describes a distance between two atoms. It needs only two atomic indices to be given, the order of which is arbitrary.

`invr` The `invr` coordinate (for **inv**erse **r**) describes an inverse distance. The declaration is the same as for `stre`, but in some cases (if you are far away from the minimum) the use of `invr` may result in better convergence.

`bend` `bend` describes a bond angle. It requires three atoms to be specified, of which the **third** one is the atom at the apex.

`outp` Out-of-plane angle: `outp` *abcd* is the angle between bond $a-d$ and plane $b-c-d$.

`tors` Dihedral angle: `tors` *abcd* is the angle between the planes $a-b-c$ and $b-c-d$.

`linc` This is a special coordinate type to describe the bending of a near-linear system. `linc` *abcd* describes the collinear bending of $a-b-c$ (where the angle is defined as for `bend`: the apex atom appears last) **in** the plane of $b-c-d$ (see also below, command `linp`). The system $b-c-d$ has to be non-linear, of course.

`linp` This coordinate is similar to `linc`, but describes the bending of $a-b-c$ *perpendicular* to the plane $b-c-d$. These two types of coordinates are in most cases sufficient to describe the bending of near-linear systems. An example may help you to understand these two coordinate types. Consider ketene, H_2CCO, which contains a linear system of three atoms. Without symmetry, this molecule has 9 degrees of freedom. You could choose the four bond lengths, two CCH angles and the out-of-plane angle of the C–C bond out of the CHH–plane. But then two degrees of freedom still remain, which cannot be specified using these *normal* coordinate types. You can fix these by using `linc` and `linp`. The two coordinates `linc 1 3 2 4` and `linp 1 3 2 4` (where 1=oxygen, 2=carbon, 3=carbon, 4=hydrogen) would solve the problem.

`comp` The type `comp` describes a **comp**ound coordinate, i.e. a linear combination of (primitive) internal coordinates. This is often used to prevent strong coupling between (primitive) internal coordinates and to achieve better convergence of the geometry optimization. The use of linear combinations rather than primitive coordinates is especially recommended for rings and cages (see ref. [48]). Command `iaut` uses linear combinations in most cases.

After you entered `k comp` n where n is the number of primitive internal coordinates to be combined, you will be asked to enter the type of the coordinate (`stre`, `bend`, ...). Then you will have to enter the weight (the coefficient of this primitive coordinate in the linear combination) and the atomic indices which define each coordinate. The definition of the primitive coordinates is the same as described above for the corresponding coordinate types. It is not possible to combine internal coordinates of different types.

`ring` This type helps you to define special ring coordinates. You only have to enter `k ring` n where `n` is the ring size. Then you will be asked for the atomic indices of all atoms which constitute the ring and which must be entered in the same order as they appear in the ring. The maximum number of atoms in the ring is 69 (but in most cases the ring size will be limited by the maximum number of atoms which is allowed for `define`).

Hitting `<return>` will bring you back to the internal coordinate menu where you can see the new number of internal coordinates in the headline.

4.1.3 Manipulating the Geometry

Note that the molecular geometry can be modified with the `iman` command of the internal coordinate menu described earlier, if internal coordinates has been defined. Another option is to select `m` in the geometry main menu which provides the following submenu:

```
CARTESIAN COORDINATE MANIPULATION MENU :
move   : TRANSLATE AND/OR ROTATE PART OF THE MOLECULE
align  : ROTATE MOLECULE TO ALIGN TWO AXES
inert  : MOVE MOLECULE SO THAT COORDINATE AXES BECOME
         PRINCIPAL AXES OF INERTIA
mback  : RESTORE PREVIOUS MOLECULAR GEOMETRY
dis    : DISPLAY MOLECULAR GEOMETRY
YOU MAY APPEND A QUESTION MARK TO ANY OF THESE COMMANDS
FOR FURTHER EXPLANATIONS.
HIT >return< OR USE ANY GEOMETRY COMMAND NOT IN THIS LIST
TO TERMINATE THIS MENU.
UPON TERMINATION THE MOLECULAR SYMMETRY WILL BE ENFORCED
ACCORDING TO SYMMETRY GROUP c3v .
```

The meaning of the commands `inert` and `mback` should be clear; the commands `move` and `align` allow you to manipulate the geometry of your molecule. After entering `move` or `align`, you will be asked to specify a set of atoms on which the command shall act. You can use this to manipulate only a part of your molecule, e.g. if you are building a structure from subunits and you want to adjust their relative arrangement. As long as you stay in this menu, the molecular symmetry needs not be correct (so that you can try different movements and/or rotations), but as soon as you leave it, the geometry will be symmetrized according to the present Schönflies symbol. In `move`, after you specified the atomic set to be considered, you get the following information:

```
INPUT DIRECTION OF MOVEMENT OR LOCATION OF ROTATION AXIS
EITHER AS A COORDINATE TRIPLE SEPARATED BY BLANKS,
OR AS TWO ATOMIC INDICES SEPARATED BY KOMMA, OR x OR y OR z
OR ENTER ANY DISPLAY COMMAND FIRST OR & TO GO BACK
```

You can thus specify the direction of movement (or the rotational axis) in the form `0. 0. 1.` or simply `z` (which both describes the z-axis) or `1.3256 -3.333 0.2218` for an arbitrary axis. If you want to specify an axis which is related to your molecule,

you may also enter two atomic indices which define it. After having specified the axis, you have to enter the distance of movement and the angle of rotation. If you want to perform a simple rotation, enter 0 for the distance of movement and if you want to simply move your structure, enter 0 for the rotational angle.

In the `align` submenu, you are asked for two directions ("current orientation" and "new direction") and you can use the same syntax as described above. The molecule will be rotated such that the "current orientation" axis points into the new direction. For example, first entering `1,2` and then `z` rotates (a part of) the molecule such that the bond between atom 1 and atom 2 is parallel to the z axis.

You can leave this menu and return to the geometry main menu by hitting `<return>` or by entering any command of the geometry main menu.

4.2 The Atomic Attributes Menu

After you specified the molecular geometry and symmetry and wrote this data to file, you will encounter the atomic attributes menu, which is the second of the four main menus. You will enter this menu, if all necessary data cannot be read from your input file or if you do not use an input file. This menu deals with the specification of basis sets and other data related to the atom type:

```
ATOMIC ATTRIBUTE DEFINITION MENU  ( #atoms=5     #bas=5     #ecp=0     )

b    : ASSIGN ATOMIC BASIS SETS
bb   : b RESTRICTED TO BASIS SET LIBRARY
bl   : LIST ATOMIC BASIS SETS ASSIGNED
bm   : MODIFY DEFINITION OF ATOMIC BASIS SET
bp   : SWITCH BETWEEN 5d/7f AND 6d/10f
lib  : SELECT BASIS SET LIBRARY
ecp  : ASSIGN EFFECTIVE CORE POTENTIALS
ecpb : ecp RESTRICTED TO BASIS SET LIBRARY
ecpi : GENERAL INFORMATION ABOUT EFFECTIVE CORE POTENTIALS
ecpl : LIST EFFECTIVE CORE POTENTIALS ASSIGNED
ecprm: REMOVE EFFECTIVE CORE POTENTIAL(S)
c    : ASSIGN NUCLEAR CHARGES (IF DIFFERENT FROM DEFAULTS)
cem  : ASSIGN NUCLEAR CHARGES FOR EMBEDDING
m    : ASSIGN ATOMIC MASSES (IF DIFFERENT FROM DEFAULTS)
iso  : ASSIGN ISOTOPE FOR NUCLEAR COUPLING CALCULATION
dis  : DISPLAY MOLECULAR GEOMETRY
dat  : DISPLAY ATOMIC ATTRIBUTES YET ESTABLISHED
h    : EXPLANATION OF ATTRIBUTE DEFINITION SYNTAX
*    : TERMINATE THIS SECTION AND WRITE DATA OR DATA REFERENCES TO control
GOBACK=& (TO GEOMETRY MENU !)
```

The headline gives you the number of atoms, the number of atoms to which basis sets have already been assigned and the number of atoms to which effective core potentials have already been assigned. Most of the commands in this menu deal with the specification of basis sets and pseudopotentials.

Basis sets available

The following basis sets are available on `$TURBODIR/basen/`, which you may inspect to see which other basis sets are supported automatically. The corresponding publications can be found here 1.3.

`SV(P)` or `def-SV(P)`	for routine SCF or DFT. Quality is about 6–31G*.
`TZVP` or `def-TZVP`	for accurate SCF or DFT. Quality is slightly better than 6–311G**.
`TZVPP` or `def-TZVPP`	for MP2 or close to basis set limit SCF or DFT. Comparable to 6–311G(2df).
`QZVP and QZVPP`	for highly correlated treatments; quadruple zeta + 3d2f1g or 4d2f1g (beyond Ne), 3p2d1f for H.

These basis sets are available for atoms H–Kr, and the split-valence (SV) and valence-triple-ζ (TZV) basis sets types with ECPs also for Rb–Rn, except lanthanides.

For calculations with the programs `ricc2`, `ccsdf12`, and `pnoccsd` optimized auxiliary basis sets are available for the basis sets SV(P), SVP, TZVP, TZVPP, and QZVPP.

NEW: New sets of basis functions, partly identical with those mention above, denoted def2-XYZ are available for atoms H–Rn [9]. The def2 basis sets for 5p and 6p block elements are designed for small core ECPs (ECP-28, ECP-46 and ECP-60). For each family, SV, TZV, and QZV, we offer two sets of polarisation functions leading to:

`def2-SV(P)` and `def2-SVP`

`def2-TZVP` and `def2-TZVPP`

`def2-QZVP` and `def2-QZVPP`

We strongly recommended the new def2-basis, since they have been shown to provide consistent accuracy across the periodic table.

Recommendation

Use the same basis set type for all atoms; use ECPs beyond Kr since this accounts for scalar relativistic effects.

New basis sets (def2-XYZ): MP2 implies RI-MP2 and RICC2

exploratory	MP2: SVP
almost quantitative	DFT: SV(P), HF: SVP, MP2: TZVPP; properties (HF and DFT): TZVPP
quantitative	DFT: TZVP, HF: TZVPP, MP2: QZVPP

basis set limit DFT: QZVP, HF: QZVP

If you want a better basis than SV(P), assigned automatically, use `b all def2-TZVP` (or another basis). The assignment can be checked by `bl`.

Diffuse functions should only be added if really necessary. E.g. for small anions or treatment of excited states use: `TZVP` instead of `SVP` + *diffuse*. This is more accurate and usually faster. Only for excited states of small molecules or excited states with (a partial) Rydberg character add additional diffuse functions (e.g. by using the aug-cc-pVTZ basis) as well as the keyword `diffuse`, for more information, see page 429 in the keyword section.

[**Old basis sets** (def-XYZ): For standard correlated calculations (MP2, RI-MP2, RI-CC2) use the doubly-polarized TZVPP (or def-TZVPP) basis.]

Correlation-Consistent (Dunning) Basis Sets

Dunning basis sets like `cc-pVDZ`, `cc-pVTZ`, `cc-pVQZ` are also supported, e.g. by `b all cc-pVTZ`. But these basis sets employ generalized contractions for which `TURBOMOLE` is not optimized. This has in particular strong effects on the performance of all programs which use 4-index electron repulsion integrals, for RI-MP2 and RI-CC2 this is partially compensated by the RI-approximation.

The following correlation consistent basis sets are available in the `TURBOMOLE` basis set library:

cc-pVXZ standard valence X-tuple zeta basis sets (X = D, T, Q, 5, 6); available for H, He, Li–Ne, Na–Ar, K, Ca, Ga–Kr.
(cc-pV6Z only for H, He, B–Ne, Al–Ar; for Al–Ar also the recommended newer cc-pV(X+d)Z sets are available)

cc-pwCVXZ-PP weighted core-valence x-tuple zeta basis sets (X= D, T, Q, 5) are available for post-*d* main group elements Ga–Kr, In–Xe, and Tl–Rn.
(also pure valence basis sets cc-pVXZ-PP are available for these elements, but it is not recommended to use them)

cc-pwCVXZ weighted core-valence X-tuple zeta basis sets (X = D, T, Q, 5); available for B–Ne, Al–Ar, and Ga–Kr
(for Al–Ar also the recommended combination of the cc-pV(X+d)Z sets with the core valence functions (wC), i.e. the cc-pwCV(X+d)Z basis set are available)

aug- diffuse functions for combination with the basis sets cc-pVXZ, cc-pV(X+d)Z, cc-pwCVXZ, cc-pV(X+d)Z, cc-pVXZ-PP or cc-pwCVXZ-PP; available for H, He, B–Ne, Al–Ar with X = D–6 and Ga–Kr, In–Xe, and Tl–Rn with X = D–5.

cc-pVXZ-F12 with X = D, T, Q for use with the explicitly-correlated F12 variants of wavefunction methods (MP2-F12, CCSD(F12*), etc.)

For calculations with the programs that employ the RI approximation with a correlated wavefunction optimized auxiliary basis sets are available for most of the correlation consistent basis set series.

4.2.1 Description of the commands

`b` With `b` you can specify basis sets for all atoms in your molecule. After entering `b` you will be asked to specify the atoms to which you want to assign basis sets. You can do this in the usual ways (refer to Section 4.0.4), including `all` and `none`. Then you will be asked to enter the *nickname* of the basis set to be assigned. There are two principal ways to do this:

1) If you are in the *append* mode, the nickname you entered will be appended to the atomic symbol of the element under consideration. This is especially useful if you want to assign basis sets to different atoms with one command. For example, if you want to assign basis sets to hydrogen and oxygen atoms and you enter only `DZ`, the basis sets `h DZ` and `o DZ` will be read from the basis set library.
2) If you are in the *non-append* mode, no atomic symbol will be inserted in front of the nickname entered. Therefore you have to enter the *full* basis set nickname, e.g. `h DZ`. This mode is advantageous if you want to assign basis sets to dummy centers (i.e. points without nuclear charge but with basis functions, e.g. for counterpoise calculations) or if you want to use the basis set nickname `none` (which means no basis functions at this atom).

You can switch between the two modes with '`+`' (switches to append mode) and '`-`' (switches to non-append mode).

Once you have specified your basis set nickname, `define` will look in the standard input file (normally `control`) for this basis set. If it can not be found there, you can switch to the standard basis set library (if you did not use a standard input file, the standard library will be searched immediately). If the basis set cannot be found there, you are asked either to enter a new standard library (which will be standard only until you leave this menu) or another input file, where the basis set can be found. If you do not know the exact nickname of your basis set, you may abbreviate it by '`?`', so you could enter `h DZ?` to obtain basis sets like `h DZ`, `h DZP`, `h DZ` ***special***, etc. `define` will give you a list of all basis sets whose nicknames match your search string and allows you to choose among them. You may also use the command `list` to obtain a list of all basis sets cataloged.

`bb` `bb` does essentially the same as `b` but does not search your default input file for basis sets. Instead it will look in the basis set library immediately.

`bl` `bl` gives you a list of all basis sets assigned so far.

`bm` This command is used to modify basis sets which are already assigned. The corresponding submenu is self-explanatory, but we recommend to change directly the file `basis`.

`bp` The `TURBOMOLE` programs normally work with basis sets of $5d$-functions (which means they delete the s-component of the full $6d$-set). `bp` allows to switch between the proper $5d/7f$-set and the Cartesian $6d/10f$-set.

`ecp` This command allows you to specify effective core potentials for some atoms. The assignment works exactly like the specification of basis sets (see above).

`ecpb` This one does the same as command `ecp`, but restricted to the basis set library (the input file will not be used).

`ecpi` `ecpi` gives you some general information about what type of pseudopotentials is supported. For more information we refer to [49] and references therein.

`ecpl` `ecpl` gives you a list of all pseudopotentials assigned so far.

`ecprm` `ecprm` allows to remove a pseudopotential assignment from the list. This command is useful if you want to perform an all electron calculation after an ECP treatment.

`c` Command `c` assigns a special nuclear charge to an atom. This is useful to define dummy centers for counterpoise calculations where you set the nuclear charge to zero.

`m` This command allows you to assign non-default atomic masses to an atom. Use this if you want to analyze isotopic shifts of calculated harmonic frequencies. The standard masses are those of the natural isotope mix.

`iso` This command allows you to assign non-default gyromagnetic ratios to an atom. For practically all isotopes, it is sufficient to enter the mass number (sum of protons and neutrons) to obtain the gyromagnetic ratio.

`dat` `dat` gives you a list of all data already specified.

`*` This is again the usual command to leave a menu and write all data to file `control` (or any other output file). It is not possible to leave this menu unless basis sets have been specified for all atoms in your molecule. If you do not want to use a basis set for one or more atoms, use the basis set nickname `none`. On leaving this menu, the data groups `$atoms` and `$basis` will be written to the output file.

After you finished this menu, you will enter the third main menu of `define` which deals with start vectors and occupation numbers.

4.3 Generating MO Start Vectors

4.3.1 The MO Start Vectors Menu

This menu serves to define the occupation numbers, and to generate the start vectors for HF-SCF and DFT calculations. They may be constructed from earlier SCF calculations (perhaps employing another basis set, type `use`), by Hamilton core guess (`hcore`), or by an extended Hückel calculation which can be performed automatically (`eht`). An occupation of the start orbitals will be proposed and can be modified if desired.

```
OCCUPATION NUMBER & MOLECULAR ORBITAL DEFINITION MENU

 CHOOSE COMMAND
 infsao     : OUTPUT SAO INFORMATION
 atb        : Switch for writing MOs in ASCII or binary format
 eht        : PROVIDE MOS && OCCUPATION NUMBERS FROM EXTENDED HUECKEL GUESS
 use <file> : SUPPLY MO INFORMATION USING DATA FROM <file>
 man        : MANUAL SPECIFICATION OF OCCUPATION NUMBERS
 hcore      : HAMILTON CORE GUESS FOR MOS
 flip       : FLIP SPIN OF A SELECTED ATOM
 &          : MOVE BACK TO THE ATOMIC ATTRIBUTES MENU
 THE COMMANDS  use  OR  eht  OR  *  OR q(uit) TERMINATE THIS MENU !!!
 FOR EXPLANATIONS APPEND A QUESTION MARK (?) TO ANY COMMAND
```

Recommendation

You will normally only need to enter `eht`. For the EHT-guess, `define` will ask for some specifications, and you should always choose the default, i.e. just `<enter>`. Of importance is only the molecular charge, specified as `0` (neutral, default), `1` or `-1` etc.

Based on the EHT orbital energies `define` proposes an occupation. If you accept you are done, if not you get the "occupation number assignment menu" explained in 4.3.2.

Description of Commands

`infsao` Command `infsao` provides information about the symmetry adapted basis which is used for the SCF-calculation. To exploit the molecular symmetry as efficiently as possible, `TURBOMOLE` programs do not use the simple basis which you specified during your `define` session. Instead it builds linear combinations of equal basis functions on different—but symmetry equivalent—atoms. This basis is then called the SAO-basis (**S**ymmetry **A**dapted **O**rbital). It has the useful property that each basis function transformed to this basis transforms belongs to one irreducible representation of the molecular point group (that is, the basis reflects the full molecular symmetry as specified by the Schönflies symbol). `infsao`

gives you a listing of all symmetry adapted basis functions and their constituents either on file or on the screen. This may help you if you want to have a closer look at the SCF vectors, because the vector which is output by program `dscf` is written in terms of these SAOs.

`atb` Molecular orbitals can be written either in ASCII or in binary format. This command switches from one option to the other, and it is highly recommended to read which setting is currently active. ASCII format is portable and allows the usage of `TURBOMOLE`input files on different systems with incompatible binary format. Binary format is faster and smaller files will be written. The external program `atbandbta` can be used to transform existing `mos`, `alpha`, and `beta` files from ASCII to binary format and vice versa.

`eht` `eht` performs an extended Hückel calculation for your molecule. The orbital energies available from this calculation are then used to provide occupation numbers for your calculation and the Hückel MOs will be projected onto the space that is spanned by your basis set. This start-vectors are not as good as the ones which may be obtained by projection of an old SCF vector, but they are still better than the core Hamiltonian guess that is used if no start vectors are available. When using this command, you will be asked if you want to accept the standard Hückel parameters and to enter the molecular charge. Afterwards you will normally get a list of the few highest occupied and lowest unoccupied MOs, their energies and their default occupation. If you don't want to accept the default occupation you will enter the occupation number assignment menu, which is described in Section 4.3.2. Note that the occupation based on the Hückel calculation may be unreliable if the difference of the energies of the HOMO and the LUMO is less than 0.05 a.u. (you will get a warning). You will also have to enter this menu for all open-shell cases other than doublets.

`use` *file* With command `use` you are able to use information about occupied MOs and start vectors from a former calculation on the same molecule. *file* should be the path and name of the `control` file of this former calculation, of which all data groups related to occupation numbers and vectors will be read. As the new generated data will overwrite the existing data if both resist in the same directory, it is best and in some cases necessary to have the data of the former calculation in another directory than the one you started the `define` session in. Then just type `use <path>/control` to construct a new SCF vector from the data of the old calculation, without changing the old data. The data groups `$closed shells` and `$open shells` will be taken for your new calculation and the SCF vector from the old calculation will be projected onto the space which is spanned by your present basis set. These start vectors are usually better than the ones you could obtain by an extended Hückel calculation.

`man` `man` allows you to declare occupation numbers or change a previous declaration manually. After selecting this command, you will get a short information about the current occupation numbers:

```
------------------------------------------------------------
actual closed shell orbital selection         range
------------------------------------------------------------
 a1                                           #   1-  18
 a2                                           #   1-   1
 e                                            #   1-  13
------------------------------------------------------------

any further closed-shell orbitals to declare ? DEFAULT(y)
```

If you answer this question with `y`, you enter the orbital specification menu which will be described in Section 4.3.3.

The same procedure applies to the open-shell occupation numbers after you finished the closed-shell occupations.

`hcore` `hcore` tells programs `dscf` and `ridft` to run without a start vector (it writes the data group `$scfmo none` to file `control`). `dscf` or `ridft` will then start from the core Hamiltonian start vector, which is the vector obtained by diagonalizing the one-electron Hamiltonian. Note that you still have to specify the occupation numbers. This concerns only the first SCF run, however, as for the following calculations the converged vector of the previous iteration will be taken. A SCF calculation with a core Hamiltonian start vector typically will take $2-3$ iterations more than a calculation with an extended Hückel start vector (a calculation with the converged SCF vector of a previous calculation will need even less iterations, depending on how large the difference in the geometry between the two calculations is).

`flip` flipping of spins at a selected atom. Requirements: converged UHF molecular orbitals and no symmetry (C_1). `define`will localize the orbitals, assign them to the atoms and give the user the possibility to choose atoms at which alpha-orbitals are moved to beta orbitals, or vice versa. This is useful for spin-broken start orbitals, but not for spatial symmetry breaking.

`*` This command (as well as `use` and `eht`) terminates this menu, but without providing a start vector. If the keyword `$scfmo` exists in your input file, it will be kept unchanged (i.e. the old vector will be taken), otherwise `$scfmo none` will be inserted into your output file, which forces a calculation without start vector to be performed. When you leave this menu, the data groups `$closed shells`, `$open shells` (optionally) and `$scfmo` will be written to file. You will then reach the last of the four main menus (the General Menu) which is described in Section 4.4.

4.3.2 Assignment of Occupation Numbers

If an automatic assignment of occupation numbers is not possible or you do not except the occupation numbers generated by the EHT, you enter the following menu:

```
OCCUPATION NUMBER ASSIGNMENT MENU ( #e=60 #c=0 #o=0)

 s          : CHOOSE UHF SINGLET OCCUPATION
 t          : CHOOSE UHF TRIPLET OCCUPATION
 u <int>    : CHOOSE UHF WITH <int> UNPAIRED ELECTRONS
 l <list>   : PRINT MO'S FROM EHT IN <list>, (DEFAULT=ALL)
 p <index>  : PRINT MO-COEFFICIENTS OF SHELL <index>
 c <list>   : CHOOSE SHELLS IN <list> TO BECOME CLOSED SHELLS
 o <index>  : CHOOSE SHELL <index> TO BECOME AN RHF OPEN SHELL
 a <list>   : CHOOSE SHELLS IN <list> TO BECOME UHF ALPHA SHELLS
 b <list>   : CHOOSE SHELLS IN <list> TO BECOME UHF BETA SHELLS
 v <list>   : CHOOSE SHELLS IN <list> TO BECOME EMPTY SHELLS
 &          : REPEAT THE EXTENDED HUECKEL CALCULATION
 *          : SAVE OCCUPATION NUMBERS & GO TO NEXT ITEM
 dis        : GEOMETRY DISPLAY COMMANDS
 e          : CALCULATE EHT-ENERGY
 f          : FURTHER ADVICE
 <int>      = INTEGER
 <index>    = INDEX OF MO-SHELL ACCORDING TO COMMAND s
 <list>     = LIST OF MO-SHELL INDICES (LIKE  1-5,7-8,11)
```

Recommendation

Enter `l` to get a list of EHT MO energies. Then make up your mind on what to do: closed shell, RHF open shell (not allowed for DFT) or UHF. Look at the examples below.

RHF `c 1-41,43,45` to define these levels to be doubly occupied.

UHF `a 1-5 alpha` levels to be occupied, `b 1-3,5 beta` levels to be occupied. Or simply, `s`, `t`, or `u 1` to get singlet, triplet or doublet occupation pattern.

ROHF `c 1-41,43,45` levels to be doubly occupied; `o 42` level 42 should be partially occupied. You will then be asked to specify the occupation. If there are more open shells you have to repeat, since only a single open shell can be specified at a time. Watch the headline of the menu, which tells you the number of electrons assigned to MOs.

Description of Commands

`s` *list* This command gives you a listing of all MOs and their energies as obtained from the extended Hückel calculation. For NH_3 in C_{3v} and `TZVP` you get, e.g.:

```
ORBITAL SYMMETRY  ENERGY      SHELL    CUMULATED  CL.SHL OCC.  OP.SHL OCC.
(SHELL)   TYPE              DEGENERACY SHELL DEG. PER ORBITAL  PER ORBITAL
    1     1a1      -15.63244      2           2      0.0000       0.0000
    2     2a1       -0.99808      2           4      0.0000       0.0000
    3     1e        -0.64406      4           8      0.0000       0.0000
    4     3a1       -0.57085      2          10      0.0000       0.0000
    5     2e         0.30375      4          14      0.0000       0.0000
    6     4a1        0.87046      2          16      0.0000       0.0000
TO CONTINUE, ENTER <return>
```

`p` *index* This allows you to get the linear combination of basis functions which form the MO-index. Note that this refers *not* to the basis set you specified, but to the extended Hückel basis. *index* must be a single index, not an index list.

`c` *list* This command allows you to specify closed shells. Their occupation will be 2 per MO, the total occupation the shell degeneracy which you can obtain by using command `s`. *list* is a list of shell indices like `1-13` or `1,3-5,7`.

`o` *index* This command allows you to specify open shells. *index* must be a single shell index, not an index list. You will then be asked for the number of electrons **per MO** which shall be contained in this shell. For example, for a fluorine atom you should choose `o n` (where `n` is the index of the *p*-shell) and an occupation of 5/3 per MO. You may enter the occupation numbers as simple integers or as integer fractions, e.g. `1` for the *s*-occupation in sodium, `5/3` for the *p*-occupation in fluorine.

`v` *list* With this command you can remove an orbital occupation, if you specified a wrong one. *list* is again a list of shell indices in usual syntax.

`&` This command has a different meaning in this menu than in the rest of `define`. Here it will repeat the extended Hückel calculation (perhaps you want to change some Hückel parameters for the next one).

`*` `*` will *not* bring you back to the occupation numbers menu, but will terminate the whole occupation number and start vector section and will bring you to the last main menu, which is described in Section 4.4. If you want to leave this menu without assigning all electrons in your molecule to shells, `define` will issue a warning and suggest to continue defining occupation numbers. You can ignore this warning, if you do not want to assign all electrons.

`e` Calculates and displays the extended Hückel total energy of your molecule.

`f` `f` will give you some information about the commands in this menu.

You may overwrite occupation numbers once given by just redefining the corresponding shell. For example, if you choose shells 1–10 as closed shells and afterwards shell no. 9 as open shell (with any occupation number), the open shell will be correctly assigned.

4.3.3 Orbital Specification Menu

`define` provides the possibility to assign the occupation numbers of the MOs manually, if you like. To do that, use the command `man` in the occupation number main menu and you will arrive at the following submenu:

```
------------ ORBITAL SPECIFICATION MENU --------------

<label> <list>  : select orbitals within <list>
-<label> <list> : skip orbitals within <list>
&               : ignore input for last label
clear           : clear all assignments
p(rint)         : print actual orbital selection
for help, type ? or help // for quit, type * or q(uit)
```

Depending on whether you are in the closed- or in the open-shell section, the commands of this menu refer only to the corresponding type of orbitals. The commands of this menu do not need much explanation. `<label>` is the irrep label of one irreducible representation of the molecular point group (e.g. `a1`, `b2`, `t1g`, ...). `<list>` is a list of orbital indices within this *irrep* (e.g. `1,2,4` or `1-8,10,11`). `p` or `print` will give you the same listing of the orbital occupations as you saw before entering this menu. After you leave this submenu, you will be back in the occupation numbers main menu.

4.3.4 Roothaan Parameters

In open-shell calculations within the restricted Hartree–Fock ansatz (ROHF), the coupling between the closed and the open shells must be specified using two parameters `a` and `b`, which depend on the type of the open shell, the number of electrons in it (the electron configuration), but also on the state to be calculated. For example, there are three states arising from the s^2p^2 configuration of an atom (^{3}P, ^{1}D, ^{1}S) which have different values of a and b. For the definition of these parameters and their use refer to Roothaan's original paper [50]. For simple cases, `define` sets these parameters automatically. If not, you have to enter them yourself. In this case, you will get the following message:

```
ROOTHAAN PARAMETERS a AND b COULD NOT BE PROVIDED ...
TYPE IN ROOTHAAN a AND b AS INTEGER FRACTIONS
OR ENTER val FOR AN AVERAGE OF STATES CALCULATION
OR ENTER  &  TO REPEAT OCCUPATION NUMBER ASSIGNMENT
```

Note that *not* all open shell systems can be handled in this way. It is possible to specify a and b for atomic calculations with s^n, p^n, d^1, and d^9 configurations and for calculations on linear molecules with π^n and δ^n configurations. Furthermore, it is possible to do calculations on systems with half-filled shells (where a=1, b=2). In the literature you may find tabulated values for individual states arising from d^n configurations, but these are **not** correct. Instead, these are parameters for an average of all states arising from these configurations. You can obtain these values if you enter `val` on the above question. For a detailed description see Section 6.3.

4.3.5 Start-MOs for broken symmetry treatments ("flip")

Broken-symmetry treatments suggested by e.g. Noodleman or Ruiz are a popular tool for the calculation of spin coupling parameters in the framework of DFT. As an example one might consider two coupled Cu^{II} centers, e.g. for a (hypothetical) arrangement like this:

```
$coord
 0.0  2.7 0.0 cu
 0.0 -2.7 0.0 cu
 0.0 -6.1 0.0 f
 0.0  6.1 0.0 f
 2.4  0.0 0.0 f
-2.4  0.0 0.0 f
$end
```

The high-spin case, a doublet with an excess alpha electron at each Cu atom, "aa" in an obvious notation, preserves D_{2h} symmetry, while the low spin state "ba" does not. For a broken-symmetry treatment, it is advisable to calculate the high-spin state first, and then broken-symmetry state(s); from the energy difference(s) one may calculate approximate values for the spin-spin coupling parameters as described by e.g. the above authors. Access to broken-symmetry states usually is possible by the choice of appropriate start-MOs, followed by an SCF-procedure. Start MOs may be obtained by first applying a localization procedure to the MOs of the high-spin state and then by "moving" localized alpha orbitals to the beta subset.

The preparation of broken-symmetry start-MOs can be done with `define` (semi-)automatically. Prerequisite is a converged wave function for the high-spin state in C_1-symmetry, that fulfills the *aufbau* principle.

If in this case one enters `flip` in the orbital definition menu, `define` selects the occupied valence orbitals of the system (by an orbital energy criterion, which one can usually accept, unless the system is highly charged and the orbital energies are shifted). Next a Boys orbital localization procedure is carried out, which - depending on the size of the problem - may take some time. Then the user is asked:

```
ENTER INDICES OF ATOMS OR ELEMENT TO BE MANIPULATED (example:  1,2-3 or
"mn")
```

In case of our above example one may enter `"cu"`, which immediately leads to the following output (a def-SV(P) basis and the B-P functional were used for the high-spin state):

```
 RELEVANT LMOS FOR ATOM   1 cu
 ALPHA:
index occupation "energy"     s    p    d    f   (dxx  dyy  dzz  dxy  dxz  dyz)
  15    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  18    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  20    1.000     -0.335     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  22    1.000     -0.333     0.01 0.00 0.99       0.13 0.03 0.32 0.00 0.00 0.51
```

```
  23    1.000     -0.333     0.01 0.00 0.99       0.14 0.03 0.34 0.00 0.00 0.49

 BETA:
  39    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  41    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  43    1.000     -0.321     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  46    1.001     -0.318     0.05 0.00 0.95       0.00 0.43 0.51 0.00 0.00 0.00

 RELEVANT LMOS FOR ATOM   2 cu
 ALPHA:
index occupation "energy"     s    p    d    f   (dxx  dyy  dzz  dxy  dxz  dyz)
  16    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  17    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  19    1.000     -0.335     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  21    1.000     -0.333     0.01 0.00 0.99       0.13 0.03 0.32 0.00 0.00 0.51
  24    1.000     -0.333     0.01 0.00 0.99       0.14 0.03 0.34 0.00 0.00 0.49

 BETA:
  40    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  42    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  44    1.000     -0.321     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  45    1.001     -0.318     0.05 0.00 0.95       0.00 0.43 0.51 0.00 0.00 0.00

 a2b    : FLIPPING ALPHA TO BETA (default)
 b2a    : FLIPPING BETA TO ALPHA
 r      : repeat atom choice
```

As evident from the second column, for each Cu atom five localized alpha and four localized beta orbitals were generated which are of d-type (the sixth column labelled "d" shows values close to 1, the other columns such close to 0). The six columns at the right show the individual contributions of the six Cartesian d-functions.

What has to be done to generate start MOs for the "ba"-case? Obviously one of the five localized alpha spin orbitals from the first Cu atom (atom label `1 cu`) has to become a beta spin orbital. These five orbitals have the indices `15, 18, 20, 22, 23`. In order to avoid linear dependencies, it is advisable to take the orbital that has no beta-analogue. This can be found by comparing the contributions of the six d-functions. In the present example this is the case for the localized alpha orbitals `15` and `18`: in contrast to all localized beta orbitals they show significant contributions from d_{xy}. One thus enters

```
a2b
15
```

and after confirming the replacement of original MOs with the generated start-MOs one is finally asked

```
 It is advisable to modify damping and orbital shift in the following way:
```

```
 $scfdamp   start=5.000  step=0.050  min=0.500
 $scforbitalshift  automatic=1.0
 $scfiterlimit 999

 Do you want to replace the corresponding entries in the control-file? (y)
```

which should be confirmed, as otherwise the prepared spin state might be destroyed during the SCF iterations.

From this input one may start the SCF(HF/DFT)-procedure. For recommended choices of DFT functionals and formulae to calculate the coupling parameters from these energy differences please consult the papers of the above-mentioned authors. For reasons of economy, a pre-optimization by a pure (non-hybrid) DFT-functional is reasonable.

Important: For the converged wave function one should check, whether the resulting state is really the desired one. This can quite reliably be done by a Mulliken population analysis. For this purpose, add `$pop` to the control file, type `ridft -proper` or `dscf -proper`, respectively, and check the signs of the calculated numbers of unpaired electrons in the output.

4.4 The General Options Menu

After you specified all data concerning the molecule you want to examine, you are on your way to the last of the four main menus. Before reaching it, you will perhaps get a message like the following:

```
DO YOU WANT TO DELETE DATA GROUPS LIKE
    $energy
    $grad
    $hessian
    $hessian (projected)
    $last energy change
    $maximum norm of internal gradient
    $dipgrad
    $vibrational normal modes
    $vibrational spectrum
    $cartesianforce interspace
LEFT OVER FROM PREVIOUS CALCULATIONS ? DEFAULT(n)
```

`define` has scanned your input file for this session and found some data groups which might have become obsolete. If they are still acceptable depends on the changes you made during your present `define` session. They are obviously incorrect if you changed the molecule under consideration; but any change in the basis sets or the occupation numbers will make them dangerous, too, because you might not know some day if they really refer to the basis set which is defined in this `control` file. As a rough guide, delete them whenever you have made changes in one of the first three main menus during your `define` session.

After that you will reach the last main menu of `define` which helps you to control the actions of all TURBOMOLE programs. The meanings of the various options are explained in more detail in the description of the individual programs, therefore only a short explanation will be given here.

Now have a look at the menu:

```
GENERAL MENU : SELECT YOUR TOPIC
scf    : SELECT NON-DEFAULT SCF PARAMETER
mp2    : OPTIONS AND DATA GROUPS FOR rimp2 and mpgrad
pnocc  : OPTIONS AND DATA GROUPS FOR pnoccsd
cc     : OPTIONS AND DATA GROUPS FOR ricc2
ex     : EXCITED STATE AND RESPONSE OPTIONS
prop   : SELECT TOOLS FOR SCF-ORBITAL ANALYSIS
drv    : SELECT NON-DEFAULT INPUT PARAMETER FOR EVALUATION
         OF ANALYTICAL ENERGY DERIVATIVES
         (GRADIENTS, FORCE CONSTANTS)
rex    : SELECT OPTIONS FOR GEOMETRY UPDATES USING RELAX
stp    : SELECT NON-DEFAULT STRUCTURE OPTIMIZATION PARAMETER
e      : DEFINE EXTERNAL ELECTROSTATIC FIELD
dft    : DFT Parameters
ri     : RI Parameters
rijk   : RI-JK-HF Parameters
rirpa  : RIRPA Parameters
gw     : OPTIONS AND DATA GROUPS FOR GW (escf)
senex  : seminumeric exchange parameters
hybno  : hybrid Noga/Diag parameters
dsp    : DFT dispersion correction
nmr    : NMR shift parameters
ncoup  : NMR coupling parameters
epr    : EPR parameters
trunc  : USE TRUNCATED AUXBASIS DURING ITERATIONS
marij  : MULTIPOLE ACCELERATED RI-J
dis    : DISPLAY MOLECULAR GEOMETRY
list   : LIST OF CONTROL FILE
&      : GO BACK TO OCCUPATION/ORBITAL ASSIGNMENT MENU
* or q : END OF DEFINE SESSION
```

This menu serves very different purposes. The next subsection deals with commands required to activate and/or specify specific methods of calculation. The subsequent subsection describes commands used to select non-default options. Standard SCF calculations do not require special action, just leave the menu. The final subsection describes the settings for property calculations.

4.4.1 Important commands

DFT calculations

Command `dft` leads you to the menu:

```
STATUS OF DFT_OPTIONS:
 DFT is NOT used
   functional b-p
     gridsize m3

 ENTER DFT-OPTION TO BE MODIFIED

 func: TO CHANGE TYPE OF FUNCTIONAL
 grid: TO CHANGE GRIDSIZE
  on:   TO SWITCH ON  DFT
 Just <ENTER>, q or '*' terminate this menu.
```

To activate DFT input `on` and then specify the grid for the quadrature of exchange-correlation terms. `TURBOMOLE` offers grids `1` (coarse) to `7` (finest), and the multiple grids `m3` to `m5` [7]. The latter employ a coarser grid during SCF iterations, and grid `3` to grid `5` in the final SCF iteration and the gradient evaluation. Default is grid `m3`, for clusters with more than 50 atoms use `m4`. Note that larger grids are required for relativistic all-electron calculaations. These are employed by appending `a` to the gridsize, e.g., `4a`. For hybrid density functionals, the multiple grids do not result in a notable reduction of the computational costs and the standard grids are recommended. These are also recommended for response properties and parallel calculations, especially for magnetic properties such as NMR shifts and coupling constants. See also Sec. 8.

The functionals supported within `define` are obtained with the command `func`; the first part of the output looks like

```
 SURVEY OF AVAILABLE EXCHANGE-CORRELATION ENERGY FUNCTIONALS

 FUNCTIONAL      | TYPE | EXCHANGE       | CORRELATION    | REFERENCES
 ------------------------------------------------------------------------
 s-vwn           | LDA  | S              | VWN(V)         | 1-3
 s-vwn_Gaussian  | LDA  | S              | VWN(III)       | 1-3
 pwlda           | LDA  | S              | PW             | 1,2,4
 b-lyp           | GGA  | S+B88          | LYP            | 1,2,6
 b-vwn           | GGA  | S+B88          | VWN(V)         | 1-3,5
 b-p             | GGA  | S+B88          | VWN(V)+P86     | 1-3,5,7
 pbe             | GGA  | S+PBE(X)       | PW+PBE(C)      | 1,2,4,8
 tpss            | MGGA | S+TPSS(X)      | PW+TPSS(C)     | 1,2,4,14
 bh-lyp          | HYB  | 0.5(S+B88)     | LYP            | 1,2,5,6,9
                 |      | +0.5HF         |                |
```

This list is far from being complete. Especially with the inclusion of the XCFun and LibXC libraries, uncounted combinations of exchange and correlation functionals

are available. To get an overview of further functionals and how to use arbitrary combinations and exact exchange portions, please read section 6.2.

Default is `b-p`, i.e. B-P86, which is probably best for the whole of Chemistry [51]. For main group compounds we recommend `b3-lyp`; note that GAUSSIAN uses partly different implementations [51].

The programs `dscf` and `grad` are used to carry out conventional DFT treatments, i.e. J and K are evaluated without approximations.

RI-J calculations

For non-hybrid functionals we strongly recommend the RI-J procedure, which speeds up calculations by a factor 10 at least (as compared to conventional treatments) without sacrificing accuracy. Command `ri` gives:

```
STATUS OF RI-OPTIONS:
    RI IS NOT USED
    Memory for RI:           200 MB
    Filename for auxbasis: auxbasis

 ENTER RI-OPTION TO BE MODIFIED
      m: CHANGE MEMORY FOR RI
      f: CHANGE FILENAME
   jbas: ASSIGN AUXILIARY RI-J BASIS SETS
     on: TO SWITCH ON  RI
 Use <ENTER>, q, end, or * to leave this menu
```

Activate RI-J with `on`, and choose with `m` the memory you can dedicate to store three-center integrals (Keyword: `$ricore`), default is 200 MB. The *more* memory, the *faster* the calculation.

A rough guide: put `$ricore` to about 2/3 of the memory of the computer. Use OS specific commands (`top` on most UNIX systems), during an `ridft` run to find the actual memory usage and then adjust `$ricore`, the keyword in `control` specifying memory.

If the option `jbas` is selected, `define` enters a submenu which allows the assignment of auxiliary basis sets (for an explanation of the menu items see Section 4.2). Where available, the program will select by default the auxiliary basis sets optimized for the orbital basis used. Please note that treatment of systems with diffuse wavefunctions may also require an extension of the auxiliary basis. For this cases enlarge the sets of s- and p-functions with diffuse functions.

The RI-J option is only supported by programs `ridft` and `rdgrad`, if you use `jobex` to optimize molecular geometry, put: `nohup jobex -ri ...`

MARI-J option

RI-J calculations can be done even more efficiently with the **M**ultipole **A**ccelerated **RI**-J (MARI-J) option, especially for larger molecules where almost linear scaling

is achieved [52].

```
Parameters:
 1) precision parameter:               1.00E-06
 2) maximum multipole l-moment:              10
 3) maximum number of bins:                   8
 4) minimum separation of bins:            0.00
 5) maximum allowed extension:            20.00
 6) threshold for multipole neglect:   1.00E-18

Enter the number to change a value or <return> to accept all.
```

Just rely on the defaults.

Multiple auxiliary basis sets

With the command `trunc` you can switch on this option. Effect: a reduced auxiliary (or fitting) basis to represent the electron density is employed during SCF iterations, the final SCF iteration and the gradient are computed with the full auxiliary basis.

```
 truncated RI ALREADY SWITCHED ON
 DO YOU WANT TO SWITCH OFF truncation ? (default=no)
```

Note: `trunc` is presently not compatible with marij!

RI in SCF calculations

Considerable savings in CPU times are achieved with the RI technique for both Coulomb J and exchange K terms in SCF calculations, the RI-JK method [53], provided large basis sets are employed, e.g. `TZVPP`, `cc-pVTZ`, or `cc-pVQZ`. With `rijk` you get:

```
STATUS OF RI-OPTIONS:
    RI IS NOT USED
    Memory for RI:           200 MB
    Filename for auxbasis: auxbasis

 ENTER RI-OPTION TO BE MODIFIED
      m: CHANGE MEMORY FOR RI
      f: CHANGE FILENAME
  jkbas: ASSIGN AUXILIARY RI-JK BASIS SETS
     on: TO SWITCH ON  RI
 Use <ENTER>, q, end, or * to leave this menu
```

For an explanation of the menu items see Section 4.4.1. RI-JK calculations can be carried out with the program `ridft`.

Optimization to minima and transition structures using STATPT

Structure optimizations can be carried out by the program `statpt`. For minimizations no additional keywords are required. The default values are assumed, which work in most of the cases. Structure optimization is performed in internal coordinates if they have been set. Otherwise, Cartesian coordinates are used. One can switch the optimization in internal coordinates on or off, respectively in internal redundant or Cartesian coordinates. For transition structure optimizations the index of transition vector has to be set to an integer value > 0 (0 means structure minimization). The value of the index specifies transition vector to follow during the saddle point search. Note, that Hessian eigenpairs are stored in ascending order of the eigenvalues, i.e. the eigenpair with the smallest eigenvector has the index 1.

The command `stp` gives:

```
-------------------------------------------------------------------------
CONVERGENCE CRITERIA:

thre  1.000000E-06     thre : threshold for ENERGY CHANGE
thrd  1.000000E-03     thrd : threshold for MAX. DISPL. ELEMENT
thrg  1.000000E-03     thrg : threshold for MAX. GRAD. ELEMENT
rmsd  5.000000E-04     rmsd : threshold for RMS OF DISPL.
rmsg  5.000000E-04     rmsg : threshold for RMS OF GRAD.

defl : set default values.

-------------------------------------------------------------------------

OPTIMIZATION refers to

int   off          int: INTERNAL coordinates
rdn   off          rdn: REDUNDANT INTERNAL coordinates
crt   on           crt: CARTESIAN coordinates
NOTE : options int and crt exclude each other

ENTER STATPT-OPTIONS TO BE MODIFIED

itvc    0              itvc : change INDEX OF TRANSITION VECTOR
updte   bfgs           updte: change method of HESSIAN UPDATE
hsfrq   0              hsfrq: frequency of HESSIAN CALCULATION
kptm    0              kptm : FREEZING transition vector INDEX
hdiag 5.000000E-01     hdiag: change DIAGONAL HESSIAN ELEMENTS
rmax  3.000000E-01     rmax : change MAX. TRUST RADIUS
rmin  1.000000E-04     rmin : change MIN. TRUST RADIUS
trad  3.000000E-01     trad : change TRUST RADIUS
-------------------------------------------------------------------------
Just <ENTER>, q or '*' terminate this menu.
```

Excited states, frequency-dependent properties, and stability analysis

Excited state calculations with RPA or CIS (based on HF-SCF) and TDDFT procedures as well as stability analyses (SCF or DFT) are carried out by the program `escf`.

You will need a well converged HF-SCF or DFT calculation that were converged to at least `$scfconv=7`, see Section 4.4.2.

Details of calculations are specified with the command `ex`:

```
MAIN MENU FOR RESPONSE CALCULATIONS

OPTION | STATUS | DESCRIPTION
------------------------------------------------------------------
rpas   | off    | RPA SINGLET EXCITATIONS (TDHF OR TDDFT)
ciss   | off    | TDA SINGLET EXCITATIONS (CI SINGLES)
rpat   | off    | RPA TRIPLET EXCITATIONS (TDHF OR TDDFT)
cist   | off    | TDA TRIPLET EXCITATIONS (CI SINGLES)
polly  | off    | STATIC POLARIZABILITY
dynpol | off    | DYNAMIC POLARIZABILITY
single | off    | SINGLET STABILITY ANALYSIS
triple | off    | TRIPLET STABILITY ANALYSIS
nonrel | off    | NON-REAL STABILITY ANALYSIS
bse    | off    | BETHE-SALPETER EX.
cbse   | off    | corr-aug. BETHE-SALPETER EX.

ENTER <OPTION> TO SWITCH ON/OFF OPTION, * OR q TO QUIT
```

If you have selected an option, e.g. `rpas`, and quit this menu, you will get another menu:

```
SELECT IRREP AND NUMBER OF STATES
ENTER ? FOR HELP, * OR Q TO QUIT, & TO GO BACK
```

This should be self-evident. The input of the excitations is used to estimate the memory requirements. The *GW* method is available in a separate gw menu:

```
NO GW OPTIONS CURRENTLY SET
GW                     (gw on/off) : off
RI-GW                (rigw on/off) : off
RPA                   (rpa on/off) : off

INPUT MENU FOR GW CALCULATIONS WITH escf:
gw      : TURN STANDARD GW USING SPECTRAL FUNCTIONS (N^6) ON/OFF
rigw    : TURN RI-GW USING ANALYTIC CONTINUATION (N^4) ON/OFF
rpa     : TURN RPA MODULE (N^6) ON/OFF
Just <ENTER>, q or '*' terminate this menu.
```

Menu nmr, ncoup, and epr

This menu sets the flags for NMR shielding and coupling constant calculations as well as EPR properties. The nmr menu allows to switch between two different CPHF solvers for `mpshift`and to set the corresponding keywords for both, for details see chapter 17.

```
ENTER SCF-OPTION TO BE MODIFIED
Old CPHF solver is selected (default up to V7.4)
Convergence is checked for the heaviest atom by default

STATUS OF NMR OPTIONS
 The old CPHF solver (up to V7.4) is used
 A threshold of 0.1000D-01 ppm is used
 Maximum number of iterations is set to    30
 NMR shieldings are calculated for all atoms

ENTER OPTION TO BE MODIFIED
 oldcphf    :  to switch on or off old CPHF solver (up to V7.4)
 csconv     :  to select convergence criterion for old solver
 shiftconv  :  to select convergence criterion for default solver
 csmaxiter  :  to select maximum number of iterations
 nucsel     :  to calculate certain nuclei only
 gimic      :  prepare input for GIMIC calculation
Just <ENTER>, q or * to terminate this menu.
```

The ncoup menu is used to control the input for a NMR coupling constant calculation with `escf`.

```
    STATUS OF OPTIONS FOR COUPLING CONSTANTS
The following terms will be calculated: fc sd pso dso fcsdcross
The threshold for the final output is      0.1000
Wave function convergence criteria is set to     9
Response convergence criteria is set to     6
Maximum number of iterations is set to    25
Response equations will be solved for all atoms
Right-hand side integrals will be calculated for all atoms

ENTER OPTION TO BE MODIFIED
 fc/sd/pso/dso on/only/off: set terms to be calculated
 fcsdcross/all on/off
 fromfile/simple
 thr      : threshold for final output
 scfconv  : convergence criterion for wave function (recommended: >= 7)
 rpaconv  : convergence criterion for response equations (recommended: >= 6)
 limit    : maximum number of iterations in response equations ($escfiterlimit)
 nucsel   : nuclei to solve response equations for
 nucsel2  : nuclei to calculate right-hand side integrals for
Just <ENTER>, q or * to terminate this menu
```

The settings are discussed in detail in chapter 8.

The EPR menu is used to control the input for an EPR with `mpshift`.

```
STATUS OF EPR OPTIONS
 We will perform an EPR calculation
 SNSO will be included for spin-orbit coupling
 SNSO parameter     3
 HFC and EFG are calculated for all atoms
 CPHF/CPKS convergence criterion     7
 Maximum number of iterations is set to    30

 ENTER OPTION TO BE MODIFIED
   epr        :  to turn on/off the EPR calculation
   snso       :  set options for spin-orbit treatment
   nucsel     :  to calculate certain nuclei only
   shiftconv  :  to select convergence criterion for CPHF/CPKS
   csmaxiter  :  to select maximum number of iterations
   current    :  current density and magnetic XC kernel for MGGAs/LHFs
Just <ENTER>, q or * to terminate this menu
```

Please see chapter 18 for details.

MP2, MP2-F12, RI-MP2, and PNO-MP2

We recommend to use MP2 for standard applications together with the RI technique using the `ricc2` program. The `mpgrad` program is supplied for special benchmark application where the RI approximation needs to be avoided. The `pnoccsd` program is meant only for very large ($>$ 100 atoms and $>$ 3000 basis functions) MP2 and MP2-F12 single point energy calculations. This is more efficient and supports the frozen core option in the gradient calculation.

The entry `mp2` leads to a submenu which allows to set some keywords for MP2 and RI-MP2 calculations, e.g. defining frozen orbitals, maximum memory usage, or assign auxiliary basis sets for RI-MP2 calculations, etc. If you want to use `ricc2`, you have to use the entry `cc` and the submenu `ricc2` in order to assign MP2 as wavefunction model. For the `pnoccsd` program you have to use the entry `pnocc` and the submenu `pnoccsd` to assign the wavefunction model.

Conventional MP2 calculations with `mpgrad` require a number of additional settings for which it is recommended to invoke the interactive tool `mp2prep`. For geometry optimizations with `jobex` use `nohup jobex -level mp2 -ri ...`

CC2 and CCSD calculations

The entry `cc` leads to a submenu which allows to set a number of keywords essential for calculations with the programs `ricc2` and `ccsdf12`. In particular it allows the assignment of the wavefunction method(s), selection of auxiliary basis sets, the specification of frozen orbitals, and the definition of a scratch directory and of the maximum core memory usage.

The `ricc2` program must be used for excitation energies and response properties with second-order methods (MP2, CIS(D), ADC(2), CC2, etc. and their spin-scaled variants), while the `ccsdf12` program must be used for third- and higher-order methods (MP3, CCSD, CCSD(T), etc.).

2nd analytical derivatives for SCF and DFT

The program `aoforce` computes force constants and IR and Raman Spectra on SCF and DFT level. Analytical second derivative calculations can directly be started from converged SCF or DFT calculations. Note, that the basis is restricted to *d*-functions, and ROHF as well as broken occupation numbers are not allowed. For better efficiency, in case of larger systems, use the keyword `$maxcor` as described in Chapter 15 to reduce computational cost. RI will be used if the RI option for DFT has been specified.

4.4.2 Special adjustments

Adjustments described by the following menus are often better done directly in the `control` file; have a look at the keywords in Chapter 23. For common calculations just start with the defaults, and change keywords directly in `control` if you encounter problems with your calculation.

SCF options

```
ENTER SCF-OPTION TO BE MODIFIED

conv   : ACCURACY OF SCF-ENERGY           $scfconv
thi    : INTEGRAL STORAGE CRITERIA        $thize  $thime
ints   : INTEGRAL STORAGE ALLOCATION      $scfintunit
iter   : MAXIMUM NUMBER OF ITERATIONS     $scfiterlimit
diis   : DIIS CONVERGENCE ACCELERATION    $scfdiis
damp   : OPTIONS FOR DAMPING              $scfdamp
shift  : SHIFTING OF ORBITALS             $scforbitalshift
order  : ORDERING OF ORBITALS             $scforbitalorder
fermi  : THERMAL SMEARING OF OCC. NUMBERS $fermi
pdiag  : PREDIAGONALIZATION               $prediag
soghf  : SPIN ORBIT GENERALIZED SCF       $soghf
x2c    : EXACT 2C HAMILTONIAN             $rx2c
rlocal : DLU SCHEME FOR X2C               $rlocal
finnuc : FINITE NUCLEUS MODEL             $finnuc
rsym   : RELATIVISTIC SYMMETRY            $rsym
snso   : SCREENED NUCLEAR SPIN ORBIT      $snso
```

By the command `$fermi` you can switch on *smearing* of occupation numbers, and thus automatically optimize occupations and spin. The command `$soghf` guides the user for the different settings of a two-component calculation. Relativistic effects are included by ECPs or relativistic all-electron methods, see section 6.4.

Menu drv

The most important of the derivative menus is the first one which tells the programs which derivatives to calculate. This is only necessary for special purposes and you should better not change default options.

```
------------------------------------------------------------------------
 derivative data groups '$drvopt, $drvtol'
------------------------------------------------------------------------
 option | status  | description :
------------------------------------------------------------------------
 crt    |    T    | CARTESIAN 1st derivatives
 sec    |    T    | CARTESIAN 2nd derivatives
 bas    |    F    | energy derivatives with respect to
        |         |  BASIS SET exponents/scaling factors/
        |         |  contraction coefficients
 glb    |    F    | energy derivative with respect to
        |         |  a GLOBAL scaling factor
 dip    |    T    | cartesian 1st derivatives of DIPOLE MOMENT
 pol    |    T    | nuclear contribution to POLARIZABILITY
 fa     |    F    | SPECTROSCOPIC ANALYSIS only
 tol     0.100D-06  derivative integral cutoff
------------------------------------------------------------------------
 use <opt> for enabling, -<opt> for disabling of logical switches
 <&> will bring you back to GENERAL MENU without more changes
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

The handling of these options is very simple. With the exception of `tol`, all are logical switches which are either true (or on, active) or false (or off, inactive). You can switch between the two states if you enter, for example, `crt` (to switch calculation of Cartesian first derivatives on) or `-crt` (to switch it off). The options `crt`, `sec` and `bas` should provide no problems. `glb` refers to a global scaling factor for all basis set exponents. Imagine that you would like to replace your basis set, which contains basis functions

$$\chi_\mu = (x - x_0)^l (y - y_0)^m (z - z_0)^n \exp\left[-\eta_\mu (r - r_0)^2\right]$$

by another basis set which contains basis functions

$$\chi_\mu = (x - x_0)^l (y - y_0)^m (z - z_0)^n \exp\left[-\alpha\eta_\mu (r - r_0)^2\right]$$

where α is the same for all primitive basis functions χ_μ. With command `glb` you are able to calculate analytical derivatives of the total energy with respect to α and can thus easily determine the optimum α.

`dip` enables you to calculate the first derivatives of the electric dipole moment with respect to nuclear displacements which gives you infrared intensities. `pol` allows you to calculate the contribution of the nuclear rearrangement on the electric polarizability. `fa` finally performs only a frequency analysis which means that `aoforce` will read the force constant matrix (`$hessian` or `$hessian (projected)`), diagonalize it and give you the frequencies and normal modes. `tol` is not a logical switch as the

other options in this menu, but a cutoff threshold for the derivative integrals, i.e. integrals below this threshold will be neglected in the derivative calculations.

Entering * will bring you to the second derivative submenu.

Debug Options for the Derivative Programs

The following menu deals only with some debug options for `grad`. Use them with caution, each of them can produce lots of useless output:

```
------------------------------------------------------------------------
 derivative debug options '$drvdebug'
------------------------------------------------------------------------
 option  |status| description :
------------------------------------------------------------------------
 disp1e  |   F   | display 1e contributions to desired derivatives
 only1e  |   F   | calculate 1e contributions to desired derivatives only
 debug1e |   F   | display 1e shell contributions to desired derivatives
         |       |  (WARNING : this produces large outputs!)
 debug2e |   F   | display 2e shell contributions to desired derivatives
         |       |  (WARNING : this produces VERY large  outputs!)
 debugvib|   F   | debug switch for vibrational analysis (force only)
 notrans |   F   | disable transfer relations (gradient only!)
 novirial|   F   | disable virial scaling invariance in basis set
         |       |  optimizations (gradient only)
------------------------------------------------------------------------
 use <opt> for enabling, -<opt> for disabling option <opt>
 <&> will bring you back to GENERAL MENU without more changes
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

As there is no need to use these options normally and the menu text is self-explaining, no further description will be given. Note that all options are logical switches and may be enabled and disabled the same way as shown for the last menu. Entering * will bring you to the last derivative submenu.

4.4.3 Relax Options

Program `relax` has a huge variety of options to control its actions which in program `define` are grouped together in eight consecutive menus. These are only briefly described in the following sections; for a more detailed discussion of the underlying algorithms refer to the documentation of program `relax` (see Section 5.3). Only experts should try to change default settings.

Optimization Methods

The first of the `relax` subgenus deals with the type of optimization to be performed:

```
------------------------------------------------------------------------
 optimization options for RELAX
------------------------------------------------------------------------
 option | status | description : optimization refers to
------------------------------------------------------------------------
 int    |    F   | INTERNAL coordinates
 crt    |    F   | CARTESIAN coordinates
 bas    |    F   | BASIS SET exponents/scale factors
 glb    |    F   | GLOBAL scaling factor
------------------------------------------------------------------------
 use <opt> for enabling, -<opt> for disabling option <opt>
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

You can choose between a geometry optimization in the space of internal coordinates (in this case you will need definitions of internal coordinates, of course) or in the space of Cartesian coordinates (these possibilities are mutually exclusive, of course). Furthermore optimizations of basis set parameters (exponents, contraction coefficients and scaling factors) or of a global scaling factor is possible (these options are also exclusive, but can be performed simultaneous to a geometry optimization). For the geometry optimization you should normally use internal coordinates as they provide better convergence characteristics in most cases.

Coordinate Updates

The next submenu deals with the way `relax` updates the old coordinates. You may choose a maximum change for the coordinates or you can allow coordinate updates by means of extrapolation:

```
------------------------------------------------------------------------
 coordinate update options for RELAX
------------------------------------------------------------------------
 dqmax <real> : coordinates are allowed to change by at most
                <real> (DEFAULT : 0.3000    ) a.u.
 polish       : perform an interpolation or extrapolation of
                coordinates (DEFAULT :y)
 -polish      : disable inter/extrapolation
------------------------------------------------------------------------
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

These options result in better convergence of your optimization in most cases.

Interconversion Between Internal and Cartesian Coordinates

The interconversion between internal and Cartesian coordinates is not possible directly (in this direction). Instead it is performed iteratively. The following options control this conversion:

```
------------------------------------------------------------------
interconversion options for RELAX
------------------------------------------------------------------
option    | description
------------------------------------------------------------------
on        | switch on interconversion (DEFAULT: off)
qconv <r> | set convergence threshold for interconversion
          | of coordinates to <r>. DEFAULT : <r> =  .1000E-09
iter <i>  | allow at most <i> iterations for interconversion
          | of coordinates. DEFAULT : <i> =   25
crtint    | transform cartesian into internal coordinates (DEFAULT=n)
intcrt    | transform internal into cartesian coordinates (DEFAULT=n)
grdint    | transform cartesian into internal gradients (DEFAULT=n)
hssint    | transform cartesian into internal hessian (DEFAULT=n)
------------------------------------------------------------------
use -<opt> for disabling any interconversion option
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

The options `qconv` and `iter` are used in each normal `relax` run to determine the characteristics of the back-transformation of coordinates into the internal space. With the other options and *interconversion* switched on, you can force `relax` to perform only the specified coordinate transformation and write the transformed coordinates to file `control`. To achieve this, enter `on` to switch to the transformation-only mode, and one of the last four options, e.g. `crtint`, to specify the desired transformation.

Updating the Hessian

`relax` provides a variety of methods to generate an updated Hessian every cycle. This includes the well known methods such as BFGS, DFP, or MS update methods as well as some less common procedures:

```
------------------------------------------------------------
OPTIONS FOR UPDATING THE HESSIAN
------------------------------------------------------------
 option  | status | description
------------------------------------------------------------
none     |    F   | NO UPDATE (STEEPEST DESCENT)
bfgs     |    F   | BROYDEN-FLETCHER-GOLDFARB-SHANNO UPDATE
dfp      |    F   | DAVIDON-FLETCHER-POWELL UPDATE
bfgs-dfp |    F   | COMBINED (BFGS+DFP) UPDATE
ms       |    F   | MURTAGH-SARGENT UPDATE
schlegel |    F   | SCHLEGEL UPDATE
diagup   |    F   | DIAGONAL UPDATE (AHLRICHS/EHRIG)
multidim |    F   | RANK > 2 BFGS-TYPE UPDATE
ahlrichs |    T   | MACRO : AHLRICHS UPDATE (DEFAULT)
------------------------------------------------------------
USE <opt> FOR ENABLING OPTION <opt> AND THUS DISABLING
ALL OTHER OPTIONS.
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

We recommend to use the default method `ahlrichs` which provides excellent con-

vergency in most cases.

General Boundary Conditions for Update

The force constant matrix will only be updated if least `mingeo` cycles exist. The maximum number of cycles used for the update is specified by the parameter `maxgeo`. Normally the default values provided by `define` need not be changed.

```
DEFINE BOUNDARY CONDITIONS FOR UPDATE
---------------------------------------------------------------
mingeo <i> | START UPDATE IF THERE ARE AT LEAST <i> CYCLES
           | DEFAULT : min   3
maxgeo <i> | USE LAST <i> CYCLES FOR UPDATE, DEFAULT : max   4
---------------------------------------------------------------
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

Special Boundary Conditions for Ahlrichs and Pulay Updates

For the default update method `ahlrichs` some additional control parameters are available which can be defined in this menu:

```
DEFINE BOUNDARY CONDITIONS FOR AHLRICHS OR PULAY UPDATE
---------------------------------------------------------------
option    | description
---------------------------------------------------------------
modus <i> | DEFINE MODUS FOR GDIIS PROCEDURE : MINIMIZE
          |  <dq|dq> IF  <i> = 0
          |  <g|dq>  IF  <i> = 1
          |  <g|g>   IF  <i> = 2
          |  <dE>    IF  <i> = 3
          | DEFAULT : <i> = 1
fail <r>  | IGNORE GDIIS IF  <g|dq> /| <g|dq> | IS
          | LARGER THAN -<r>. DEFAULT : <r> = 0.1
---------------------------------------------------------------
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

For detailed description consult Section 5.3.

```
------------------------------------------------------------
OPTIONS FOR MANIPULATING THE HESSIAN
------------------------------------------------------------
 option      | description
------------------------------------------------------------
diagonal     | RESTRICT UPDATE TO DIAGONAL-ELEMENTS IF
             | METHOD IS BFGS,DFP OR MS. DEFAULT=n
offreset     | DISCARD OFF-DIAGONAL ELEMENTS. DEFAULT=n
offdamp <r>  | DAMP OFF-DIAGONAL ELEMENTS BY 1/(1+<r>) DEFAULT= 1.000
damp <real>  | DAMP UPDATE BY 1/(1+<real>), DEFAULT= .0000E+00
scale <real> | SCALE INPUT HESSIAN BY <real>, DEFAULT= 1.000
allow <real> | SCALE INPUT HESSIAN BY <real>/|DE| IF |DE|,
             | THE OBSERVED ABSOLUTE CHANGE IN ENERGY, IS
             | OBEYING THE CONDITION |DE| > <real> > 0.
             | DEFAULT : NO SCALING
min <real>   | DO NOT ALLOW EIGENVALUES OF HESSIAN TO DROP
             | BELOW <real>. DEFAULT= .1000E-02
reset <real> | USE <real> AS A RESET VALUE FOR TOO SMALL
             | EIGENVALUES (CP. min). DEFAULT= .1000E-02
max <real>   | DO NOT ALLOW EIGENVALUES OF HESSIAN TO BECOME
             | LARGER THAN <real>. DEFAULT= 1000.
------------------------------------------------------------
WITH THE EXCEPTION OF min,reset AND max, ALL OPTIONS MAY BE
DISABLED BY ENTERING -<opt>
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

Initialization of the Hessian

Finally there are some options to control the choice of the initial Hessian during your geometry optimization:

```
------------------------------------------------------------
FORCE CONSTANTS INITIALIZATION OPTIONS FOR RELAX
------------------------------------------------------------
OPTION    | DESCRIPTION
------------------------------------------------------------
off       | switch off initialization (DEFAULT: on)
cart      | use analytical cartesian hessian provided by a
          | 2nd derivatives calculation. DEFAULT(n)
diag      | use diagonal matrix with diagonal elements set
          | individually within data groups $intdef or
          | $basis or $global. DEFAULT(n)
unit <r>  | use multiple of the unit matrix ( H = <r>*E ).
          | DEFAULT(n) - DEFAULT <r> =  1.000
------------------------------------------------------------
NOTE THAT THESE OPTIONS ARE MUTUALLY EXCLUSIVE
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

Option `off` will be used if you have already a good Hessian from a previous calculation which may be used. `cart` describes an even better state where you have a Hessian from a calculation of the second derivatives available (`aoforce`). The other

two options describe real procedures for initialization of the Hessian. Default values: stretches (0.5), angles (0.2).

4.4.4 Definition of External Electrostatic Fields

This submenu allows you to calculate first and second numerical derivatives of the energy with respect to an external electric field. The first three options should be clear; `1st` and `2nd` are logical switches which are turned on and off the usual way (`1st` or `-1st`) and `delta` is the increment for the numerical differentiation, that is, the finite value of the external field, which replaces the (ideally) differential field:

```
------------------------------------------------------------------------
electrostatic field definition menu
------------------------------------------------------------------------
option       | status | description
------------------------------------------------------------------------
1st          |    F   | numerical 1st derivative dE/dField
2nd          |    F   | numerical 2nd derivative d2E/dField2
delta <real> |        | increment for numerical differentiation
             |        | DEFAULT =    .5000E-02
geofield     |    F   | geometry optimization with external field
man          |    F   | explicit definition of electrostatic field(s)
------------------------------------------------------------------------
```

`geofield` gives the possibility to perform a whole geometry optimization under the influence of a finite external field and thus to obtain the (distorted) minimum geometry in this field. To do this, an external electrostatic field must be defined explicitly which can be done using command `man`. Note that `geofield` must also be switched on if any properties are to be evaluated in the presence of an electric field. The most prominent example is the calculation of hyperpolarizabilies.

Take Care, due to some inconsistencies in `define` it is *always* necessary to switch on the field calculations manually. Therefore edit the `control` file after having finished your `define` session and enter `on` after the entries of `fields` and `geofield`.

4.4.5 Properties

Most properties can be calculated with the *-proper* option in the corresponding modules like `dscf` or `ridft` directly. Previously, the program `moloch` was used for this purpose and `define` is still able to use the old menu for `moloch` (see below). For a detailed list of the keywords, please see Sec. 23.2.28. If you enter `prop` in the general menu, `define` first will check whether the data group `$properties` does already exist in your `control` file or in a file referenced therein. If this is not the case you will be asked to specify the file on which `$properties` shall be written:

```
CURRENT STATUS OF PROPERTY KEYWORDS:

NO KEYWORDS FOR CALCULATION OF PROPERTIES ARE SET


SELECT ONE OF THE FOLLOWING OPTIONS:

 mvd   : Calculation of p^4 and Darwin term
 pop   : population analyses (-> submenu)
 loc   : calculation of localized orbitals
 esp   : ESP-fit
 plt   : Calculation of densities, MOs, etc. on a grid; e.g.
         for further use in visualization programs (keqword: $pointval)
 old   : old menu for moloch
 *     : leave this submenu
```

All keywords can be disabled by using the same option again and then selecting to deactivate the input (*).

Option `mvd`

The option `mvd` directly activates the calculation of the relativistic mass-velocity and the Darwin term based on perturbation theory.

Option `esp`

The option `esp` directly activates the calculation of the electrostatic potential (ESP) fit.

Option `pop`

Activating `pop` drives the calculation of various population analyses.

```
YOU MAY CHOOSE BETWEEN:

 mul:   Mulliken PA
 low:   Loewdin  PA
 nbo:   natural  PA
 pab:   PA basen on occupation numbers
 wbi:   Wiberg bond indices
 all:   all of the above PAs
 *  :   continue without activating any PA
```

After selecting one of the analyses, all atoms and bonds or a subset can be considered. It is recommended to run the PA for the full molecule as the evaluation is not time-determing. Note that Wiberg bond indices are always calculated for all bonds.

Option `loc`

Activating `loc` allows to setup the input for constructing localized molecular orbitals according to the Foster–Boys scheme.

```
CHOOSE ORBITALS FOR LOCALIZATION (default=all)

 m <list> : MOs from list are included
 ge <thr> : above list is generated automatically.
            It will contain MOs with energies
            e(HOMO)-thr < e <= e(HOMO)
 gn <int> : above list is generated automatically.
            It will contain the <int> highest
            occupied MOs
 lm <no> : list highest <no> occupied MOs
               (default: no=200)
 *       : activate localization and continue

 NOTE: for further options see manual
```

For other localization schemes such as the Pipek-Mezey method or intrinsic bond orbitals (IBOs) and the required keywords, please see Sec. 23.2.28.

Option `plt`

Using the option `plt` inserts the necessary keywords to calculate various quantities such as MOs, LMOs or the electron localization function on a grid. The default format is plt, which can be opened by TmoleX or gOpenMol. Other formats can be selected in `define`. A post-processing script to convert plt to other frequently used formats in quantum chemistry is available at the homepage of the `TURBOMOLE` project, please see https://www.turbomole.org/turbomole/utilities/.

```
YOU MAY CHOOSE BETWEEN:

 m <list> : plot for MOs from <list>
 ge <thr> : above list is generated automatically.
            It will contain MOs with energies
            e(HOMO)-thr < e <= e(HOMO)
 gn <int> : above list is generated automatically.
            It will contain the <int> highest
            occupied MOs
 lm <no>,<nv> : list highest <no> occupied and lowest <nv>
               virtual MOs (default: no=20, nv=1)
 e        : plot electron localization function (ELF)
 f <fmt>  : change format; default is plt; further options:
                        map, xyz, plv (see also manual)
 d        : plot densities (default)
 *        : activate $pointval and continue

 NOTE: for further options see manual
```

Option `old`

The program `moloch` was previously used for the calculation of properties and analysis of the wave function. The subsequent description for an older version may not work in all cases—sorry for that.

If you enter `prop` in the general menu and use the option `old`, `define` first will check whether the data group `$properties` does already exist in your `control` file or in a file referenced therein. If this is not the case you will be asked to specify the file on which `$properties` shall be written:

```
data group $properties has not yet been specified
FOR INITIALIZING <moloch> KEYWORDS ENTER
  [return] : WRITE TO CONTROL FILE control (DEFAULT),   OR
  filename : WRITE TO ANOTHER FILE
```

Afterwards you will get the following submenu which allows you to control all possible actions of program `moloch`:

```
switch on one or more of the following options <i>
<i> = 1,..., 9
for switching off option <i>, specify -<i>
( 1) trace                                    off
( 2) moments                                  off
( 3) potential                                off
( 4) cowan-griffin                            off
( 5) localization                             off
( 6) population analyses                      off
( 7) plot                                     off
( 8) firstorder                               off
selecting an already active option indicates that
suboptions shall be modified
* or q(uit) = quit | for help, type help <integer>
```

All options in this menu are selected by entering their number as indicated in the first column. For example, to switch on option `trace` enter `1`. The flag `off` will then change to `active`. To switch off an option enter its negative number, e.g. `-1` for `trace`. Most of the options require additional input and will therefore lead you to further submenus. These are briefly described below.

Option `trace` `trace` will calculate the trace of density times overlap matrix:

$$N = \mathrm{tr}\{\mathbf{DS}\}$$

If the orbitals are orthonormal, N should yield the total number of electrons in your molecule. If this is not true, your MO-vector will most probably be erroneous. For example, the vector might belong to another geometry or basis set. As this is a very sensitive test for errors like these and the calculation requires almost no time, you should always switch on this option.

Option `moments` This option leads you to the following submenu:

```
add/change options for data group $moments
option            | status | description
------------------|--------|-------------------------------
point <x> <y> <z> |    T   | reference point = (x,y,z)
atom <i>          |    F   | reference point = atom no. <i>
0th               |    T   | compute 0th moment
1st               |    F   | compute 1st moment
2nd               |    F   | compute 2nd moment
3rd               |    F   | compute 3rd moment
------------------|--------|-------------------------------
-<moment>   : skip computation of <moment>
* or q(uit) : terminate input
```

This menu serves to specify the electrostatic moments to be calculated (`0th`=charge, `1st`=dipole moment, `2nd`=quadrupole moment, `3rd`=octuple moment). The reference point is the origin of the coordinate system used in the calculation. The value of any calculated moment will be independent of this reference point, if all lower moments are zero. The default for the reference point is the origin, i.e. the coordinate system used for the calculation of the moments will be the same as the one in which the atomic coordinates are specified. The reference point may be changed by typing `point` with the three new coordinates appended. Alternatively you may choose the coordinates of one of the atoms as reference point by entering `atom` and the atom index.

Option `potential` This option collects all possible quantities related to the electrostatic field created by the molecular charge distribution. This includes the following suboptions:

```
list of suboptions :
pot          - electrostatic potential
fld          - electrostatic field
fldgrd       - electrostatic field gradient
shld         - diamagnetic shielding
file         - file reference
*            - quit
```

The meaning of the four suboptions `pot`, `fld`, `fldgrd` and `shld` will probably present no problems to you. For each of them, however, you will have to specify at which point(s) this property should be calculated. This is accomplished by one or more data groups `$points` in file `control`. After you chose one or more of the above options, you will therefore reach the next submenu which deals with the specification of these data groups:

```
there are     1 data groups $points
manipulate data group(s) $points
a             - add another data group
m <integer>   - modify <integer>th data group
m all         - modify all data groups
d <integer>   - delete <integer>th data group
d all         - delete all data groups
off <integer> - switch off <integer>th data group
off all       - switch off all data groups
on <integer>  - switch on <integer>th data group
on all        - switch on all data groups
s             - scan through data groups
*             - quit
```

The first line informs you how many of these data groups already exist in your `control` file. Each of these data groups may consist of several points at which the properties will be calculated. You may now create new data groups, delete old ones or simply switch on or off individual data groups (without deleting them from `control`). The number of different data groups `$points` as well as the number of points in each of them are not limited. However, if you use many points, you should consider specifying them in a separate file. This is most easily done using option `file` in the `potential` menu. This option will create a file for your data groups `$points` and will write a reference of this file to file `control`.

Option `cowan-griffin` This option activates the computation of the first order relativistic correction to the energy as given by the expectation value of the Cowan–Griffin operator.

Option `localization` Specifying option `localization` will switch on a Boys localization of molecular orbitals. `define` by default chooses a set of MOs to be localized according to a certain threshold for the orbital energy. Information about these are displayed like this:

```
BOYS localization will be performed with respect to x y z
number of sweeps =      10000
subset of molecular orbitals to be localized :
---> all occupied molecular orbitals
     with orbital energy above -2.00000     Hartree
------------------------------------------------------------------
 shells to be localized
------------------------------------------------------------------
a1       4-5                                          #   1-   5
e        2                                            #   1-   2
------------------------------------------------------------------
you are employing default options for localization
do you want to modify them ? DEFAULT(n)
```

If you want to change the MO selection or other options for the localization enter `y` at this point (By default or when typing `n` you will reach the `moloch` options menu again). You will then be asked whether to change the MO selection method. If you want this, you will enter a little submenu where you can choose one of three possible

selection procedures:

`all` selects all occupied orbitals

`thr` selects all occupied orbitals with orbital energy larger than a certain threshold

`man` enables you to select the MOs manually later in this section

If the selection method `thr` is specified you then will be asked for the threshold to be applied for the selection. Afterwards you have the possibility to change some other topics concerning the localization:

- specify other localization directions
- switch on utilization of localized orbitals for population analysis and/or preparation of plot data within the same `moloch` run
- set the maximum number of sweeps in the localization procedure
- specify a file where localized orbitals shall be written to

Option `population analyses` When activating this option you first have to specify whether the population analysis (PA) should be performed in the CAO (default) or AO basis. Afterwards `define` will ask you whether you want to perform a Mulliken population analysis. In this case, the following submenu will be displayed:

```
add or delete one or more special options for a
mulliken population analysis
 option | status | description
 -------|--------|---------------------------------------
 spdf   |    F   | compute MO contributions to atomic
        |        | brutto populations
 molap  |    F   | compute MO contributions to atomic
        |        | overlap populations
 netto  |    F   | compute atomic netto populations
 irpspd |    F   | compute IRREP contributions to atomic
        |        | brutto populations
 irpmol |    F   | compute IRREP contributions to atomic
        |        | overlap populations
 mommul |    F   | print electrostatic moments resulting
        |        | from atomic charges
 -------|--------|---------------------------------------
 -<option>   : switch off <option>
 * or q(uit) : leave this menu
```

Here you can activate several optional quantities to be computed along with the Mulliken PA. To switch on one or more of these options you must enter the corresponding option keywords, e.g. `spdf netto` for computation of atomic net populations and MO contributions to atomic gross populations. The status flags for these tasks will then

change from `F` (false) to `T` (true). To switch off any option you simply have to enter the corresponding keyword preceded by a '`-`', e.g. `-netto` for disabling calculation of atomic net populations.

After having left the Mulliken PA section you will be asked whether a population analysis based on occupation numbers (a modified Roby–Davidson PA) should be performed by `moloch`. When typing `y` you will see the following submenu, where you can switch on several special options for the PA in the same manner as described above.

```
add or delete one or more special options for a
population analysis based on occupation numbers
 option  | status | description
 --------|--------|---------------------------------------
 momao   |    F   | compute MO contributions to modified
         |        | atomic orbital (MAO) occupation numbers
 maodump |    F   | dump all MAOs onto standard output
 maofile |    F   | write MAOs onto a separate file
 select  |    F   | write only those MAOs which have been
         |        | employed in the population analysis
 all     |    F   | write all MAOs
 --------|--------|---------------------------------------
 note that the options select and all are complementary
 -<option>   : switch off <option>
 * or q(uit) : leave this menu
```

Afterwards you have the possibility to change the criterion to be applied for the selection of modified atomic orbitals (MAOs) within the following little submenu:

```
global criterion for selection of Modified Atomic Orbitals (MAOs) :
-------------------------------------------------------------------
 MAOs are employed if 'atomic' density eigenvalues
 exceed a threshold of  .1000
-------------------------------------------------------------------
 specify the appropriate option if you want to use another
 global criterion for selecting MAOs
 option  | status | description
 --------|--------|--------------------------------------
 eig <r> |    T   | select by eigenvalues of the
         |        | 'atomic' density matrices
 occ <r> |    F   | select by occupation numbers
 --------|--------|--------------------------------------
 <r> is the selection threshold (DEFAULT= .1000    )
 * or q(uit) : leave this menu
```

The criterion applied by default is the so-called *atomic density eigenvalue* with a threshold of 0.1. You can switch the criterion to *occupation numbers* by entering `occ`. If you also want to change the threshold, you just have to append its new value to the selection keyword, e.g. `occ .2`. Finally you can select or disable various options in connection with the computation of shared electron numbers (SEN) within the following menu:

```
actual settings for data group $shared electron numbers
 2-center shared electron numbers will be computed;
 values are printed if absolute value exceeds    .0100
 3-center shared electron numbers will be computed;
 values are printed if absolute value exceeds    .0100
 4-center shared electron numbers will be computed;
 values are printed if absolute value exceeds    .0100
add or delete one or more options for the
computation of Shared Electron Numbers (SEN)
 option  | status | description
 --------|--------|---------------------------------------
 2c <r>  |    T   | compute 2-center SEN and print if
         |        | |SEN| > <r> (DEFAULT =  .1000E-01)
 3c <r>  |    T   | compute 3-center SEN and print if
         |        | |SEN| > <r> (DEFAULT =  .1000E-01)
 4c <r>  |    T   | compute 4-center SEN and print if
         |        | |SEN| > <r> (DEFAULT =  .1000E-01)
 --------|--------|---------------------------------------
 nosym   |    F   | switch off use of symmetry
 orbs    |    F   | compute orbital contributions to SEN
 irreps  |    F   | compute irrep contributions to SEN
 --------|--------|---------------------------------------
 -<option>   : switch off <option>
 * or q(uit) : leave this menu
```

The procedure for changing the options is the same as described above. By default calculation of 2-, 3- and 4-center SENs will be enabled with thresholds of 0.01 each.

Option `plot` This option allows you to prepare the data needed for contour plots of orbital amplitudes or total electron densities. We do not recommend to prepare plotting data this way; an easier method—with an easier syntax—is to generate these data directly by the programs, where densities (also MP2 or excited ones) and Molecular orbitals are calculated. This is described in Chapter 20. If you nevertheless want to prepare the input for plotting data as needed by `moloch` using `define`, on activating `plot` you get the following menu:

```
there are     1 data groups $grid
manipulate data group(s) $grid
 a             - add another data group
 m <integer>   - modify <integer>th data group
 m all         - modify all data groups
 d <integer>   - delete <integer>th data group
 d all         - delete all data groups
 off <integer> - switch off <integer>th data group
 off all       - switch off all data groups
 on <integer>  - switch on <integer>th data group
 on all        - switch on all data groups
 s             - scan through data groups
 *             - quit
```

The commands in this menu serve for the manipulation of data groups `$grid` in an

analogous way as described for `$points` in the *potential* section above. `$grid` data groups contain the input information necessary to create the plot data by `moloch` (one data group for each plot). If you want to add a new data group you will enter this submenu:

```
specify the input orbital / input density :
mo <label>    - use occupied molecular orbital <label>
mo density    - use one electron density built from the
                occupied molecular orbitals
lmo <i>       - use localized molecular orbital no. <lmo>
mao <i> <k>   - use modified atomic orbital no. <i>
                centered on atom no. <k>
help          - explanation of the syntax for <label>
*             - quit
```

Here you may specify the orbital to be plotted. To plot the amplitude of the fifth orbital in irrep `a1`, e.g., you would enter `mo 5a1`. Equivalently you can use localized orbitals from a Boys localization procedure or modified atomic orbitals as obtained in a Roby–Davidson–Ahlrichs–Heinzmann population analysis. In the latter cases you will not have to enter an irrep label, as these orbitals are necessarily in C_1 symmetry. Instead you will have to enter the index of the orbital to be plotted (and for option `mao` the index of the atom at which it is situated). In all cases you will additionally have to specify the plane in which the amplitudes or densities will be monitored. To do this, you have to declare two vectors which span that plane and the origin of this new coordinate system relative to the one in which the atomic coordinates are given. Furthermore, you will have to create a grid of points on this plane. The orbital amplitude or electron density will then be calculated for every point in this grid. The grid is created by telling `define` the range to be included along both vectors spanning the plane (where the unit in each direction is the length of the corresponding basis vector) and the number of points to be calculated in this range. It is advantageous to use a wide grid while you test the ranges or planes which give the best results and then to switch to a finer grid for the final calculation. Finally input (MO vector) and output (plot data) files can be specified.

In case you do not want to add a new data group as described above but to change an existing one, you will be asked which one of the specifications you want to modify.

Chapter 5

Calculation of Molecular Structure and *Ab Initio* Molecular Dynamics

5.1 Structure Optimizations using the JOBEX Script

In its normal mode of operation, the shell script `jobex` controls and executes automatic optimizations of molecular geometry parameters. It will cycle through the direct SCF, gradient and force relaxation programs and stop if either the maximum number of cycles is reached or the convergence criteria (change in the total energy, maximum norm of the gradient) are fulfilled. By default, the executable programs are taken from the load modules library within the `TURBOMOLE` directory.

5.1.1 Options

Given a shell the usage is:

```
nohup jobex &
```

This command invokes structure optimization using the default program `statpt`. Structure optimizations using program `relax` can be performed using `-relax` flag:

```
nohup jobex  -relax &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. `jobex` accepts the following arguments controlling the level of calculation, convergence criteria and many more (for example `nohup jobex -gcart 4 &`):

`-energy` *integer*	converge total energy up to $10^{(-<\text{integer}>)}$ Hartree (default: 6)

`-gcart` *integer*	converge maximum norm of Cartesian gradient up to $10^{(-<\text{integer}>)}$ atomic units (default: 3)
`-c` *integer*	perform up to *integer* cycles (default: 100)
`-dscf`	begin with a direct SCF step
`-grad`	begin with a gradient step
`-statpt`	begin with a force relaxation step
`-relax`	use the `relax` program for force relaxation
`-trans`	perform transition state search
`-level` *level*	define the optimization level, *level*=`scf`, `mp2`, `cc2`, `uff`, `rirpa` or `xtb` (default is `scf`).
`-ri`	use RI modules `ridft` and `rdgrad` (fast Coulomb approximation) instead of `dscf` and `grad` as well as `rimp2` instead of `mpgrad`; obligatory option if `-level rirpa`
`-rijk`	in connection with '-level cc2', the RI-JK versions of HF and CPHF are switched on
`-ex`	perform excited state geometry optimization using `egrad`
`-l <path>`	employ programs from directory `<path>`
`-ls <path>`	load scripts from directory `<path>`
`-md`	a molecular dynamics (MD) run (using `frog` instead of `relax`)
`-mdfile` *file*	commands for MD run are contained in this file (default: `mdmaster`).
`-mdscript` *file*	option to execute a shell script before the `frog` step
`-keep`	keep program output from all optimization steps
`-help`	shows a short description of the commands above

5.1.2 Output

There will be an output written to file `job.start` which informs you about the current options. The convergence is signalled by the file `converged`; otherwise, you should find the file `not.converged` within your working directory. If `jobex` finds a file named `stop` or `STOP` in the working directory, `jobex` will stop after the present step has terminated. You can create `stop` by the command `touch stop`.

The output of the last complete cycle is written to file `job.last`, while the output of the running cycle is collected within the file `job.<cycle>`, where `<cycle>` is the index of the cycle. The convergence criteria and their current values are written out at the bottom of the `job.last` file.

5.2 Program STATPT

5.2.1 General Information

Stationary points are places on the potential energy surface (PES) with a zero gradient, i.e. zero first derivatives of the energy with respect to atomic coordinates. Two types of stationary points are of special importance to chemists. These are minima (reactants, products, intermediates) and first-order saddle points (transition states).

The two types of stationary points can be characterized by the curvature of the PES at these points. At a minimum the Hessian matrix (second derivatives of energy with respect to atomic coordinates) is positive definite, that is the curvature is positive in all directions. If there is one, and only one, negative curvature, the stationary point is a transition state (TS). Because vibrational frequencies are basically the square roots of the curvatures, a minimum has all real frequencies, and a saddle point has one imaginary vibrational "frequency".

Structure optimizations are most effectively done by so-called quasi-Newton–Raphson methods. They require the exact gradient vector and an approximation to the Hessian matrix. The rate of convergence of the structure optimization depends on anharmonicity of the PES and of the quality of the approximation to the Hessian matrix.

The optimization procedure implemented in `statpt` belongs to the family of quasi-Newton–Raphsod methods [54]. It is based on the restricted second-order method, which employs Hessian shift parameter in order to control the step length and direction. This shift parameter is determined by the requirement that the step size should be equal to the actual value of the trust radius, `tradius`, and ensures that the shifted Hessian has the correct eigenvalue structure, all positive for a minimum search, and one negative eigenvalue for a TS search. For TS optimization there is another way of describing the same algorithm, namely as a minimization on the "image" potential. The latter is known as TRIM (Trust Radius Image Minimization) [55].

For TS optimizations the TRIM method implemented in `statpt` tries to maximize the energy along one of the Hessian eigenvectors, while minimizing it in all other directions. Thus, one "follows" one particular eigenvector, hereafter called the "transition" vector. After computing the Hessian for your guess structure you have to identify which vector to follow. For a good TS guess this is the eigenvector with negative eigenvalue, or imaginary frequency. A good comparison of different TS optimization methods is given in [56].

Structure optimizations using `statpt` are controlled by the keyword `$statpt` to be present in the `control` file. It can be set either manually or by using the `stp` menu of `define`. The type of stationary point optimization depends on the value of `itrvec` specified as an option within `$statpt`. By default `itrvec` is set to 0, which implies a structure minimization. A value `itrvec` > 0 implies a transition state optimization using the eigenvalue-following TRIM algorithm, where the index of the transition vector is specified by `itrvec`. Note, that `statpt` orders eigenvalues (and eigenvectors) of the Hessian in ascending order, shifting six (or five in the case of

linear molecules) zero translation and rotation eigenvalues to the end.

Note: this order differs from that used for vibrational frequencies in the `control` file, where rotational and translational eigenvalues are not shifted.

By default a structure optimization is converged when all of the following criteria are met:

- the energy change between two optimization cycles drops below the value given by `threchange` (default: 10^{-6} a.u.),
- the maximum displacement element drops below the value given by `thrmax\-displ` (default: 10^{-3} a.u.),
- the maximum gradient element drops below the value given by `thrmaxgrad` (default: 10^{-3} a.u.),
- the root mean square of the displacement elements drops below the value given by `thrrmsdispl` (default: $5 \cdot 10^{-4}$ a.u.),
- the root mean square of the gradient elements drops below the value given by `thrrmsgrad` (defaul:t $5 \cdot 10^{-4}$ a.u.).

The default values for the convergence criteria can be changed using the `stp` menu of `define`. The necessary keywords are described in Section 23.2.26 below.

For structure optimization of minima with `statpt` as relaxation program just use:

```
jobex  &
```

TS optimizations are performed by the `jobex` invocation:

```
jobex  -trans &
```

5.2.2 Hessian matrix

The choice of the initial Hessian matrix has a great effect on the convergence of the structure optimization. At present, there are three choices for the Hessian matrix in `statpt`. For minimization, a diagonal matrix or approximate Hessian matrix from a force field calculation using `uff`(see Section 5.4) can be used. For transition state optimizations you have to provide either the "exact" Hessian or results from the lowest eigenvalue search (LES, see Section 15). Note also that you can **calculate the Hessian with a smaller basis set and/or at a lower wavefunction level, and use it for higher level structure optimization**. Usually, a Hessian matrix calculated in a minimal basis using RI-DFT is good enough for all methods implemented in `TURBOMOLE`.

`statpt` automatically takes the best choice of the Hessian from the `control` file. For minimizations it first looks for the exact Hessian and then for the UFF Hessian. If none of them is found it takes the scaled unit matrix. For transition state optimization the exact Hessian has a higher priority than the results of LES.

The results of LES can be used to obtain an initial Hessian matrix for transition state optimizations involving large molecules, where calculation of the full Hessian is too expensive. Note, that LES calculations for `statpt`, in addition to the `$les` keyword require the following keywords to be added *manually* in the `control` file:

```
$h0hessian
$nomw
```

The default Hessian update for minimization is `bfgs`, which is likely to remain positive definite. The `powell` update is the default for transition state optimizations, since the Hessian can develop a negative curvature as the search progresses.

5.2.3 Finding Minima

Simply specify the `$statpt` keyword in the `control` file and run `jobex` as explained above. You can very often speedup the optimization by calculating the initial Hessian matrix using `uff`.

5.2.4 Finding transition states

Locating minima on a PES is straightforward. In contrast, transition state optimization requires much more input. The diagonal guess Hessian will almost never work, so you must provide a computed one. The Hessian should be computed at your best guess as to what the TS should be.

The real trick here is to find a good guess for the transition state structure. The closer you are, the better. It is often difficult to guess these structures. One way to obtain a good guess is to built an approximate TS and to perform a constrained minimization by freezing internal coordinates that change most during the reaction. Alternatively, you can generate several structures intermediate to reactants and products, and compute the energy at each point. The maximum energy structure is usually a good guess for the true TS.

After obtaining a reasonable initial guess for the TS structure you have to perform a vibrational analysis (or LES calculation for a large molecule) and to identify the index of the transition vector to follow during the optimization. Ideally, this is a vector with a negative eigenvalue, or "imaginary" frequency. The best way to find the right vector is to use some graphical interface to visualize vibrations. For a reasonable guess structure there should be one vibration that resembles the reaction under study. Remember that `statpt` uses a different ordering of eigenvalues as compared to the `aoforce` output—six (five) zero eigenvalues are shifted to the end.

There is an important thing to remember at this point. Even such sophisticated optimization methods like TRIM will not replace your own chemical intuition about where transition states may be located. If you need to restart your run, do so with the coordinates which have the smallest RMS gradient. Note that the energy does not have necessarily to decrease in a transition state search (as opposed to

minimizations). It is sometimes necessary to do restart several times (including a recomputation of the Hessian) before the saddle point can be located.

Assuming you do find the TS, it is always a good idea to recompute the Hessian at this structure. It is fairly common, especially when using symmetry, that at your "TS" there is a second imaginary frequency. This means that you have not found the correct TS. The proper procedure is to distort the structure along the "extra" imaginary normal mode using the tool `screwer` (see Section 1.5). Very often such a distortion requires also lowering the point group symmetry. The distortion must be large enough, otherwise the next run will come back to the invalid structure.

5.3 Program Relax

5.3.1 Purpose

`relax` drives and controls a non-linear optimization procedure to locate the minimum (or a stationary point) of a function $f(x)$. In `TURBOMOLE` f is always the electronic energy, and the coordinates x will be referred to as *general coordinates*. They include

- cartesian atomic coordinates
- internal atomic coordinates
- exponents, contraction coefficients and scaling factors of basis functions
- a global scaling factor (a common scaling factor for all basis set exponents)

The optimization employs an iterative procedure based on gradients ∇f of the current and, if available, previous iterations. Various procedures can be applied: steepest descent, Pulay's DIIS, quasi–Newton, conjugate gradients, as well as combinations of them. `relax` carries out:

- update of general coordinates
- update of approximate Hessians if needed
- conversion of coordinates (internal $\longleftrightarrow$ Cartesian)

The mode of operation is chosen by the keywords `$optimize` and `$interconversion` and the corresponding options, which will be described in the following sections.

5.3.2 Optimization of General Coordinates

After gradients G^k have been calculated for coordinates q^k in optimization cycle k, new coordinates (or basis set exponents) q^{k+1} can be obtained from the quasi–Newton update:

$$q^{k+1} = q^k - F^k G^k$$

where F^k is the inverse of an approximate force constant matrix H^k. This method would immediately converge to the equilibrium geometry if F^k would be the inverse of the exact force constant matrix and the force field would be quadratic. In real applications usually none of these requirements is fulfilled. Often only a crude approximation to the force constant matrix H^k is known. Sometimes a unit matrix is employed (which means coordinate update along the negative gradient with all coordinates treated on an equal footing).

The optimization of nuclear coordinates in the space of internal coordinates is the default task performed by `relax` and does not need to be enabled. Any other optimization task requires explicit specifications in data group `$optimize`, which takes several possible options:

`$optimize` *options*

`internal on/off`	Structure optimization in internal coordinates.
`redundant on/off`	Structure optimization in redundant coordinates.
`cartesian on/off`	Structure optimization in Cartesian coordinates.
`basis on/off`	Optimization of basis set exponents, contraction coefficients, scaling factors.
`global on/off`	Optimization of global scaling factor for all basis set exponents.

Note: All options except `internal` are switched off by default, unless they have been activated explicitly by specifying `on`.

Some of the options may be used simultaneously, e.g.

- `internal`, `basis`
- `internal`, `global`
- `cartesian`, `basis`

Other options have to be used exclusively, e.g.

- `internal`, `cartesian`
- `basis`, `global`

The update of the coordinates may be controlled by special options provided in data group `$coordinateupdate` which takes as options:

`dqmax=`*real*	Maximum total coordinate change (default: 0.3).
`interpolate on/off`	Calculate coordinate update by inter/extrapolation using coordinates and gradients of the last two optimization cycles (default: `interpolate on`) if possible.

`statistics` *integer*/`off` Display optimization statistics for the *integer* previous optimization cycles. Without *integer* all available information will be displayed. `off` suppresses optimization statistics.

The following data blocks are used by program `relax`:

1. Input data from gradient programs `grad`, `rdgrad`, `egrad`, `rimp2`, `mpgrad`, etc.:

   `$grad` Cartesian atomic coordinates and their gradients.

   `$egrad` exponents and scale factors and their gradients.

   `$globgrad` global scale factor and its gradient.

2. Input data from force constant program `aoforce`:

   `$grad` Cartesian atomic coordinates and their gradients.

   `$globgrad` global scale factor and its gradient.

   `$hessian` the force constant matrix in the space of Cartesian coordinates.

3. Output data from program `relax`:

   `$coord` cartesian atomic coordinates.

   `$basis` exponents and scale factors.

   `$global` global scale factor.

For structure optimizations the use of (redundant) internal coordinates is recommended, see Section 4.0.6. Normally internal coordinates are not used for input or output by the electronic structure programs (`dscf`, `mpgrad`, etc.). Instead the coordinates, gradients, etc. are automatically converted to internal coordinates by `relax` on input and the updated positions of the nuclei are written in Cartesian coordinates to the data group `$coord`. Details are explained in the following sections.

5.3.3 Force Constant Update Algorithms

In a Newton-type geometry update procedure often only a crude approximation to the force constant matrix H^k is available. What can be done then is to update $F^k = (H^k)^{-1}$ in each iteration using information about previous coordinates and gradients. This constitutes the quasi–Newton or variable metric methods of which there are a few variants:

1. Murtagh/Sargent (MS):

$$F^k = F^{k-1} + \frac{Z^{k-1}(Z^{k-1})^\dagger}{(Z^{k-1})^\dagger dG^{k-1}}$$

2. Broyden/Fletcher/Goldfarb/Shanno (BFGS):

$$F^k = F^{k-1} + \frac{S(dq^{k-1})^\dagger dq^{k-1} - dq^{k-1}(dG^{k-1})^\dagger F^{k-1} - F^{k-1}dG^{k-1}(dq^{k-1})^\dagger}{S1}$$

3. Davidon/Fletcher/Powell (DFP):

$$F^k = F^{k-1} + \frac{(dq^{k-1})^\dagger dq^{k-1}}{S1} - \frac{F^{k-1}dG^{k-1}(dG^{k-1})^\dagger F^{k-1}}{(S-1)S1}$$

4. combined method (BFGS/DFP): If $S1 < (S-1)S1$ and $S1 > 0$ perform DFP update, otherwise BFGS.

The meaning of the symbols above is as follows:

$F^k = (H^k)^{-1}$ approximate inverse force constant matrix in the k-th iteration.s

q^k general coordinates in the k-th iteration.

G^k gradients in the k-th iteration.

$dq^{k-1} = q^k - q^{k-1}$

$dg^{k-1} = g^k - g^{k-1}$

$Z^{k-1} = dq^{k-1} - F^{k-1}dG^{k-1}$

$S1 = (dq^{k-1})^\dagger dg^{k-1}$

$S = 1 + ((dg^{k-1})^\dagger F^{k-1}dG^{k-1})/(S1)$

An alternative is to use update algorithms for the hessian H^k itself:

Ehrig, Ahlrichs : *Diagonal* update for the hessian by means of a least squares fit

$$H^k_{ii} = \sqrt{H^{k-1}_{ii}(h_i + d_i)}$$

with the new estimate h for the diagonal elements obtained by

$$h_i = \frac{\sum_k dG^k_i dq^k_i}{\sum_k (dq^k_i)^2}$$

and the error d obtained by the regression

$$d_i = \frac{\sqrt{\frac{\sum_k (dg^k_i)^2}{\sum_k (dq^k_i)^2} - h_i^2}}{k-2}\,.$$

Another alternative is to use DIIS-like methods: structure optimization by direct inversion in the iterative subspace. (See ref. [57] for the description of the algorithm). The DIIS procedure can often be applied with good success, using static or updated force constant matrices.

Any of the algorithms mentioned above may be chosen. Recommended is the macro option `ahlrichs`, which leads to the following actions (n is the maximum number of structures to be included for the update, default is $n = 4$):

ncycles $< n$: geometry update by inter/extrapolation using the last 2 geometries.

ncycles $\geq n$: diagonal update for the hessian as described above; DIIS–like update for the geometry.

$||G|| <$ thr: BFGS-type update of the hessian and quasi–Newton update of (generalized) coordinates.

References for the algorithms mentioned above: [54, 57–61]

5.3.4 Definition of Internal Coordinates

If structure optimizations are to be performed in the space of internal coordinates (`$optimize internal`, is the default setting), appropriate internal coordinate definitions have to be provided on data block `$intdef`. The types available and their definitions are described in Section 4.1.2. For recommendations about the choice of internal coordinates consult ref. [48]. Nevertheless the structure of `$intdef` will shortly be described. The syntax is (in free format):

```
 1   k   1.00000000   bend  1  2  3    val=1.9500  fdiag=.6666
```

The first items have been explained in Chapter 4.

Two additional items `val=`*real*, `fdiag=`*real* may be supplied for special purposes:

`val=` serves for the input of values for internal coordinates for the interconversion internal → cartesian coordinates; it will be read in by `relax` if the flag for interconversion of coordinates has been activated (`$interconversion on`), or by the interactive input program `define` within the geometry specification menu.

`fdiag=` serves for the input of (diagonal) force constants for the individual internal coordinates to initialize `$forceapprox`.

5.3.5 Structure Optimizations Using Internal Coordinates

This is the default task of `relax` (`$optimize internal on` does not need to be specified!) You need as input the data groups :

`$grad` Cartesian coordinates and gradients as provided and accumulated in subsequent optimization cycles by the programs `grad`, or `rdgrad` etc.

`$intdef` definitions of internal coordinates.

`$redundant` definitions of redundant coordinates.

Output will be the updated coordinates on `$coord` and the updated force constant matrix on `$forceapprox`. If any non-default force constant update option has been

chosen, `relax` increments its counting variables `<numgeo>`, `<numpul>` within command keyword `$forceupdate`. If the approximate force constant has been initialized (`$forceinit on`) `relax` switches the initialization flag to `$forceinit off`. Refer also to the general documentation of `TURBOMOLE`. It is recommended to check correctness of your definition of internal coordinates:

1. Calculate their values for your Cartesian start coordinates using the `relax` program (see Section 5.3.11) or within a `define` session.
2. Have a look at the eigenvectors of the $\mathbf{BmB}^{\dagger}$-matrix. Set some '?' behind keyword `$intdef`, if there are any eigenvalues close to zero ($< 10^{-2}$ is to be considered bad for small molecules, but there is no general rule) check those internal coordinates for consistency which contribute to the corresponding eigenvector(s)!

5.3.6 Structure Optimization in Cartesian Coordinates

For this task you have to specify:

```
$optimize
  cartesian on
  internal  off
```

These lines switch on the non-default optimization in Cartesian coordinates and switch off the optimization in internal coordinates (this has to be done explicitly!). As input data groups you need only `$grad` as provided by on of the gradient programs. For the first coordinate update an approximate force constant matrix is needed in data group `$forceapprox`. Output will be the updated coordinates on `$coord`, and the updated force constant matrix on `$forceapprox`.

The coordinates for any single atom can be fixed by placing an 'f' in the third to eighth column of the chemical symbol/flag group. As an example, the following coordinates specify acetone with a fixed carbonyl group:

```
$coord
    2.02693271108611      2.03672551266230      0.00000000000000      c
    1.08247228252865     -0.68857387733323      0.00000000000000      c f
    2.53154870318830     -2.48171472134488      0.00000000000000      o     f
   -1.78063790034738     -1.04586399389434      0.00000000000000      c
   -2.64348282517094     -0.13141435997713      1.68855816889786      h
   -2.23779643042546     -3.09026673535431      0.00000000000000      h
   -2.64348282517094     -0.13141435997713     -1.68855816889786      h
    1.31008893646566      3.07002878668872      1.68840815751978      h
    1.31008893646566      3.07002878668872     -1.68840815751978      h
    4.12184425921830      2.06288409251899      0.00000000000000      h
$end
```

5.3.7 Optimization of Basis Sets (SCF only)

For this task you have to specify:

```
$optimize
  basis     on
  internal  off
```

This example would perform only a basis set optimization without accompanying geometry optimization. It is possible, of course, to optimize both simultaneously: Just leave out the last line of the example (`internal off`). Input data groups are:

`$egrad` Basis set exponents, contraction coefficients, scaling factors and their respective gradients as provided and accumulated in subsequent optimization cycles by one of the programs `grad` or `mpgrad`, if `$drvopt basis on` has been set.

`$basis` Description of basis sets used, see Section 4.2.

Output will be the updated basis on `$basis`, and the updated force constant matrix on `$forceapprox`.

For an example, see Section 24.5.

5.3.8 Simultaneous Optimization of Basis Set and Structure

The optimization of geometry and basis set may be performed simultaneously and requires the specification of:

```
$optimize
  internal  on   (or: cartesian on)
  basis     on
```

and needs as input data groups `$grad` and `$egrad`. Output will be on `$coord`, `$basis`, also on `$forceapprox` (updated).

5.3.9 Optimization of Structure and a Global Scaling Factor

Optimization of a global scaling factor is usually *not* performed in geometry optimizations. It is a special feature for special applications by even more special users. As reference see [62].

To optimize the structure and a global scaling factor specify:

```
$optimize
  internal  on   (or: cartesian on)
  global    on
```

You need as input data groups `$grad` and `$globgrad`, the latter contains the global scaling factors and their gradients accumulated in all optimization cycles. Output will be on `$coord`, `$global`, also on `$forceapprox` (updated). Note that for optimization of a global scaling factor a larger initial force constant element is recommended (about 10.0).

5.3.10 Conversion from Internal to Cartesian Coordinates

Due to translational and rotational degrees of freedom and the non-linear dependence of internal coordinates upon Cartesian coordinates, there is no unique set of Cartesian coordinates for a given set of internal coordinates. Therefore an iterative procedure is employed to calculate the next local solution for a given Cartesian start coordinates. This task may be performed using the `relax` program, but it is much easier done within a `define` session.

5.3.11 Conversion of Cartesian Coordinates, Gradients and Force Constants to Internals

To perform this tasks, you have to activate the interconversion mode by

```
$interconversion on
   cartesian --> internal   coordinate gradient hessian
```

Note that any combination of the three options showed is allowed! The default value is `coordinate`, the two other have to be switched on explicitly if desired.

You need as input data groups:

`intdef`	Definitions of (redundant) internal coordinates
`coord`	Cartesian coordinates (for option '`coordinate`')
`grad`	Cartesian coordinates and gradients as provided and accumulated in subsequent optimization cycles by the various gradient programs (for `coordinate` and `gradient`)
`hessian`	Analytical force constant matrix (as provided by the force constant program `aoforce`) (only if option `hessian` is specified). The data group `$hessian (projected)` may be used alternatively for this purpose.

All output will be written to the screen except for option `hessian` (output to data group `$forceapprox`)

5.3.12 The m-Matrix

The m-matrix serves to fix position and orientation of your molecule during geometry optimizations. It cannot be used to fix internal coordinates! The m-matrix is a

diagonal matrix of dimension $3n^2$ (where n is the number of atoms). Normally m will be initialized as a unit matrix by `relax`. As an example consider you want to restrict an atom to the xy-plane. You then set the m(z)–matrix element for this atom to zero. You can use at most six zero m-matrix diagonals (for linear molecules only five)—corresponding to translational and rotational degrees of freedom. Note that the condition of the $\mathbf{BmB}^{\dagger}$-matrix can get worse if positional restrictions are applied to the m-matrix. m-matrix elements violating the molecular point group symmetry will be reset to one. Non-default settings for m-matrix diagonals of selected atoms have to be specified within data group `$m-matrix` as:

```
$m-matrix
     1   0.0  0.0  0.0
    10   1.0  0.0  0.0
    11   1.0  1.0  0.0
```

5.3.13 Initialization of Force Constant Matrices

The most simple initial hessian is a unit matrix. However, better choices are preferable. For structure optimizations using internal coordinates you may use structural information to set up a diagonal force constant matrix with elements chosen in accord to the softness or stiffness of the individual modes. For detailed information refer to ref. [60]. For optimization of basis set parameters less information is available. When neither data block `$forceapprox` is available nor `$forceinit on` is set, the force constant matrix will be initialized as a unit matrix. Specifying the force constant initialization key `$forceinit on diag=...` will lead to:

`diag=`*real* Initialization with *real* as diagonal elements.

`diag=default` Initial force constant diagonals will be assigned the following default values:

internal coordinates	:	stretches	0.50
		angles	0.20
scaling factors	:	s,p	1.50
		d	3.00
exponents	:	uncontracted	0.15
		contracted	10.00
contraction coefficients	:		100.00
global scaling factor	:		15.00
cartesian force constants	:		0.50

`diag=individual` Initial force constant diagonals will be taken from
`$intdef fdiag=...` or
`$global fdiag=...`
Similar initialization modes are NOT supported for geometry optimization in Cartesian space and for the optimization of basis set parameters!

`carthess` Data group `$hessian` (projected) is used.

5.3.14 Look at Results

The `energy` file includes the total energy of all cycles of a structure optimization completed so far. To get a display of energies and gradients use the UNIX command `grep cycle gradient` which yields, e. g. H_2O.

```
cycle =      1    SCF energy =    -76.3432480651   |dE/dxyz| =  0.124274
cycle =      2    SCF energy =    -76.3575482860   |dE/dxyz| =  0.082663
cycle =      3    SCF energy =    -76.3626983371   |dE/dxyz| =  0.033998
cycle =      4    SCF energy =    -76.3633251080   |dE/dxyz| =  0.016404
cycle =      5    SCF energy =    -76.3634291559   |dE/dxyz| =  0.010640
cycle =      6    SCF energy =    -76.3634910117   |dE/dxyz| =  0.000730
```

This should be self-evident. To see the current—or, if the optimization is converged, the final—atomic distances use the tool `dist`. Bond angles, torsional angles etc. are obtained with the tools `bend`, `tors`, `outp`, etc. In the file `gradient` are the collected Cartesian coordinates and corresponding gradients of all cycles. The values of the general coordinates and corresponding gradients are an output of `relax` written to `job.<cycle>` of `job.last` within `jobex`. To look at this search for '`Optimization statistics`' in `job.last` or `job.<cycle>`.

5.4 Force Field Calculations

5.4.1 Purpose

`uff` preoptimizes a structure and calculates an analytical Hessian which can be used as a start Hessian in a geometry optimization. This will accelerate the convergence of an optimizations. For optimizations in Cartesian space this will be faster by a factor of two for any molecule.

5.4.2 How to Perform a UFF Calculation

You have to generate Cartesian coordinates (file `coord`), nothing else. You can start an single-point calculation calculation by typing

```
uff
```

To start an `uff` geometry optimization, one has to change the number of cycles (parameter `maxcycle`) in the block `$uff` in the file `control`. The ouput is the optimized structure (file `coord`), the analytical gradient (file `uffgradient`) and the analytical cartesian hessian (file `uffhessian0-0`). Furthermore the `control` file will be modified:

```
$forceinit on
  carthess
$uffhessian file=uffhesian0-0
```

These commands have the effect to initialize the force constant matrix for a geometry optimization with the hessian one.

In some cases `uff` cannot recognize the connectivity, then one can specify the connectivity in the file `ufftopology`. The program will calculate the bond, angle, torsion, inversion and non-bonded terms (force field terms) based on the connectivity specified in the topology file.

5.4.3 The UFF implementation

The `uff` implementation follows the paper by Rappé [10]. The energy expression in `uff` is as follows:

$$
\begin{aligned}
\mathrm{E}_{UFF} = & \sum^{N_B} \frac{1}{2} \cdot K_{IJ} \cdot (r - r_{IJ})^2 \\
& + \sum^{N_A} \begin{cases} \frac{K_{IJK}}{4} \left(1 - \cos(2\theta)\right) : \text{linear case} \\ \frac{K_{IJK}}{9} \left(1 - \cos(3\theta)\right) : \text{trigonal planar case} \\ \frac{K_{IJK}}{16} \left(1 - \cos(4\theta)\right) : \text{quadratic planar case} \\ \frac{K_{IJK}}{16} \left(1 - \cos(4\theta)\right) : \text{octahedral case} \\ K_{IJK} \cdot \left(C_0^A + C_1^A \cos\theta + C_2^A \cos(2\theta)\right) : \text{general case} \end{cases} \\
& + \sum^{N_T} \frac{1}{2} \cdot V_\phi \cdot \left(1 - \cos\left(n\phi_0\right) \cos(n\phi)\right) \\
& + \sum^{N_I} V_\omega \cdot \left(C_0^I + C_1^I \cos\omega + C_2^I \cos 2\omega\right) \\
& + \sum^{N_{nb}} D_{IJ} \cdot \left(-2 \left(\frac{x_{IJ}}{x}\right)^6 + \left(\frac{x_{IJ}}{x}\right)^{12}\right) \\
& + \sum^{N_{nb}} \frac{q_I \cdot q_J}{\epsilon \cdot x}
\end{aligned}
\tag{5.1}
$$

The Fourier coefficients C_0^A, C_1^A, C_2^A of the general angle terms are evaluated as a function of the *natural* angle θ_0:

$$C_2^A = \frac{1}{4 \sin^2 \theta_0} \tag{5.2}$$

$$C_1^A = -4 \cdot C_2^A \cos\theta_0 \tag{5.3}$$

$$C_0^A = C_2^A \left(2 \cos^2 \theta_0 + 1\right) \tag{5.4}$$

The expressions in the energy term are:

$N_B, N_A, N_T, N_I, N_{nb}$	the numbers of the bond-, angle-, torsion-, inversion- and the non bonded-terms.

K_{IJ}, K_{IJK}	force constants of the bond- and angle-terms.
r, r_{IJ}	bond distance and *natural* bond distance of the two atoms I and J.
θ, θ_0	angle and *natural* angle for three atoms $I-J-K$.
C_0^A, C_1^A, C_2^A	Fourier coefficients of the general angle terms.
ϕ, ϕ_0	torsion angle and *natural* torsion angle of the atoms $I-J-K-L$.
V_ϕ	height of the torsion barrier.
n	periodicity of the torsion potential.
ω	inversion- or out-of-plane-angle at atom I.
V_ω	height of the inversion barrier.
C_0^I, C_1^I, C_2^I	Fourier coefficients of the inversions terms.
x, x_{IJ}	distance and *natural* distance of two non bonded atoms I and J.
D_{IJ}	depth of the Lennard–Jones potential.
q_I, ϵ	partial charge of atoms I and dielectric constant.

One major difference in this implementation concerns the atom types. The atom types in Rappé's paper have an underscore "_". In the present implementation an sp^3 C atom has the name "C 3" instead of "C_3". Particularly the bond terms are described with the harmonic potential and the non-bonded van der Waals terms with the Lennard–Jones potential. The partial charges needed for electrostatic non-bonded terms are calculated with the Charge Equilibration Modell (QEq) from Rappé [63]. There is no cutoff for the non-bonded terms.

The relaxation procedure distinguishes between molecules wih more than 90 atoms and molecules with less atoms. For *small* molecules it consists of a Newton step followed by a linesearch step. For *big* molecules a quasi-Newton relaxation is done. The BFGS update of the force-constant matric is done [58, 64–66]. Pulay's DIIS procedure is implemented for *big* molecule to accelerate the optimization [57, 67].

The coordinates for any single atom can be fixed by placing an 'f' in the third to eighth column of the chemical symbol/flag group. As an example, the following coordinates specify acetone with a fixed carbonyl group:

```
$coord
    2.02693271108611      2.03672551266230      0.00000000000000      c
    1.08247228252865     -0.68857387733323      0.00000000000000      c f
    2.53154870318830     -2.48171472134488      0.00000000000000      o     f
```

```
   -1.78063790034738     -1.04586399389434      0.00000000000000      c
   -2.64348282517094     -0.13141435997713      1.68855816889786      h
   -2.23779643042546     -3.09026673535431      0.00000000000000      h
   -2.64348282517094     -0.13141435997713     -1.68855816889786      h
    1.31008893646566      3.07002878668872      1.68840815751978      h
    1.31008893646566      3.07002878668872     -1.68840815751978      h
    4.12184425921830      2.06288409251899      0.00000000000000      h
$end
```

5.5 Semiempirical Extended Tight-Binding Calculations

5.5.1 Purpose

`tb` optimizes a structure using GFN2-xtb as described in the papers of S. Grimme et. al [68, 69].

5.5.2 How to Perform a xTB Calculation

There are two ways to run xTB calculations:

- Either prepare a usual `TURBOMOLE` input with arbitrary basis set and method. Only the coordinates will be used, all other settings are ignored. If you want to add a charge, please note that you have to use the `$tb` keyword.
- Or just use a simple Cartesian coordinate file (`coord`) in `TURBOMOLE` format. No control file is needed. Start a geometry optimization by typing

```
jobex -level xtb
```

To add options the `control` file has to be modified:

```
$tb
  charge -1
  gfn <method>
  accuracy 1.0
  etemp 300
  broydamp
  maxiter <number>
```

The options are:

`charge 0`

optionally sets the molecular total charge. If not specified, a neutral input is assumed.

`gfn 2`
: choose method, GFN(1)-xTB or GFN2-xTB, default is 2

`accuracy 1.0`
: accuracy for SCC calculations, lower value means higher accuracy. Default is 1.0

`etemp 300`
: electronic temperature in Kelvin, default is 300K

`broydamp 0.4`
: Broyden damping for charge mixing, default is 0.4

`maxiter 250`
: maximum number of SCC iterations, default is 250

5.6 Molecular Dynamics Calculations

Ab initio molecular dynamics (MD) can be carried out on the ground and excited state Born–Oppenheimer potential hypersurface. In addition non-adiabatic Tully-type *Surface Hopping* MD can be performed using TDDFT. At the start of an MD run the user must specify the initial atomic positions and velocities and give some general instructions for the run. This is managed by running the interactive program `Mdprep` and generating the command file `mdmaster`. If this is successful, the MD run itself may be started: `jobex -md`. Time is then advanced in steps. The electronic potential energy and its gradients are calculated quantum mechanically at the required coordinates each time step (as detailed above, e.g. `dscf` and `grad`). The MD program `frog` uses the Leapfrog Verlet algorithm [70] to turn the gradients into new atomic positions and velocities. The atoms thus undergo classical Newtonian dynamics on the *ab initio* potential hypersurface. Trajectory information is recorded in a log file (`mdlog`). It is possible to instruct `frog` to heat or cool the system, use a thermostat for canonical dynamics, conserve total energy or read in new positions or velocities: the appropriate keywords are described in Section 23.2.29 below.

5.7 Global Structure Optimization – The DoDo Program

5.7.1 Genetic Algorithm

The reliable structure prediction of gas-phase clusters is complicated by a large number of possible structural isomers, often in combination with several low-lying electronic states. Brut force determination of the most stable configuration by manual construction of all models and following local structure optimizations is a formidable task. Instead, the `DoDo` program allows the automatic determination of the most stable molecular structures using the genetic algorithm (GA) to locate the global minimum of the total (electronic) energy. The GA is a search heuristic that employs principal mechanisms of natural evolution such as inheritance, mutation, selection, and crossover for exploration of the potential energy surface.

Generally, in GA a population of candidate structures is evolved toward improved solutions. Within the DoDo program the GA is initialized with a pool of randomly (automatically) generated structures, with initial structure models supplied by the user – so called seeds – or a combination of both. Every seed is used one or more times to create initial structures by randomly adding or removing atoms until the predefined composition and system size is reached. All initial structures are subsequently geometry optimized to the nearest local minimum. Next, two evolutionary operators – crossover and mutation – are used to exchange structure information between the members of the current population. In the crossover, pairs of structures are chosen to act as parents that will produce a new structure for the next generation. Within each parent pair, random pieces are exchanged to form the new mixed structure, the child, which is subsequently optimized to the nearest local minimum as well. The assumption is that the child will combine the good structural features of the parents and thus will be more stable than either one. Evolutionary pressure towards improved children is added to bias the search in the right direction. This is achieved by selecting parents that have a relatively high fitness that is defined as a function of their total energy:

$$f_i = \frac{\exp\left(-\alpha \cdot \frac{E_i - E_{min}}{E_{max} - E_{min}}\right)}{\sum_{i=1}^{n}\left(\exp\left(-\alpha \cdot \frac{E_i - E_{min}}{E_{max} - E_{min}}\right)\right)} \tag{5.5}$$

where f_i is the fitness of the ith individual, n is the number of individuals, α is a constant scaling factor, and E_i is the energy of that individual relative to the maximum (E_{max}) and minimum (E_{min}) energies of structures in the whole population. Since the child structures not always present a better fitness than their parents, elitism or natural selection is applied. It is achieved by simply replacing parents with a worse fitness by children with a better one in the structures pool. In order to maintain a maximum diversity during the GA runs similarity recognition is used that allows only distinct structures to be included in the pool. To prevent trapping of the population in local minima mutation is added in which random changes are introduced to randomly chosen structures in the pool.

The determination of the global minimum structure typically requires, depending on the system size, several hundreds to few thousands of local structure optimizations of the children and mutated structures. Therefore, the DoDo program uses a simple parallelization in which a predefined number (usually 10-100, depending on computer resources) of local optimizations are performed simultaneously, each preferably running on a single or just a few CPU cores. When a number of local optimizations finish, *i.e.* the number of running/queued geometry optimizations falls below a minimum value a, natural selection is applied and new child structures are generated by crossover or mutation and submitted for local optimization until the maximum number b of simultaneously running/queued optimizations is reached. This pool-based GA allows for an optimal utilization of computer resources since the algorithm is not required to wait for the completion of local optimizations of all children structures in a particular generation. The progress of the pool-based GA is then measured by number of locally optimized structures rather than by the generation count as in

case of the generation-based GA ($a = 1$).

More details of the implemented GA and application examples can be found in Ref. [71] and references therein.

5.7.2 How to Perform

In order to start a global optimization procedure one directory is required containing at least:

1. `DoDo input file` that specifies the parameters of the GA run (see also section 5.7.3).
2. `Tmole input file` that contains setup information for all local optimizations (see also `Tmole` documentation).
3. `job script` for the automatic submission of local optimization jobs to the queueing system during the GA. Such a script starts a geometry optimization using `tmole20` and, *e.g.*, a `Tmole input file` named "turbo.in" with the line:

```
tmole20 -l -c coord turbo.in > turbo.log &> turbo.err
```

A file containing seed structures can be optionally included in the directory. The GA is started by calling the `DoDo` main program with:

```
nohup genetic --gen_inp=INPUT_FILE >& OUTPUT_FILE &
```

This initiates a process running in background of the clusters front-end that: i) supervises the GA run, ii) stores the population and the history of the run, iii) performs the natural selection algorithm, iv) calls specialized genetic operator modules, v) performs input/output operations, vi) interacts with the queueing system. The generated output file summarizes the progress of the run including information about minimal and average energies of structures in a certain population, crossover and mutation operations as well as the status of queued jobs.

Each local optimization is performed in a separate directory and as an independent job (stored in "calculations"). The `Tmole input file` and `job script` supplied by the user are copied into the created working directory. Then, the molecular structure to be optimized is automatically exported as `coord` file and the job is submitted to the queuing system using the command compatible with the given job scheduling software (*e.g.* `qsub` for PBS). The `DoDo` package stores the unique identifier returned by the queuing system upon the job submission. Job progress is periodically checked using the identifier and the proper command (*e.g.* `qstat` for PBS). After the job is finished, the relevant output files created by `TURBOMOLE` are parsed in order to obtain the final structure definition and the corresponding energy.

The structures of the current population (with the highest fitness) are stored in a file named "POPULATION.dodo" while the file "DEAD.dodo" contains all structures

that were removed from population due to the natural selection algorithm. A complete history of previous populations and rejected structure models is stored in the directory "structures". These files as well as the files for the initial seed structures use the internal `DoDo` format (distance units: Bohr).

For conversion between the `DoDo` format and other commonly used file formats (*e.g.*, car and xyz) one can employ the script `dodoconvert` along with the following options:

`--inpfile=`*FILE*	name of file to dodoconvert (default: POPULATION.dodo)
`--inpform=`*FORMAT*	file format of base structure: car, xyz or dodo (default: dodo)
`--outform=`*FORMAT*	output file format: car, arc, xyz, mxyz (xyz file optimized for molden) or dodo (default: mxyz)
`--ab`	change coordinates from Ångstrom to Bohr (default: False)
`--ba`	change coordinates from Bohr to Ångstrom (default: False)

Should it be necessary to abort a GA run one can use `genetic --clear` to remove all submitted jobs from the queue before stopping the main process. In order to restart the GA, set the number of (random) initial structures to zero and remove the declaration of the seed file (if used) in the `DoDo input file`. Then, the latest "POPULATION.dodo" and "DEAD.dodo" files from the previous run are automatically read when `genetic` is executed again.

5.7.3 The DoDo Input File

A sample input file used for the `DoDo` genetic algorithm, "genetic_example.inp", is generated by calling:

```
genetic --tmole_example
```

Such an input file presents all keywords implemented in the program. For clarity, these keywords are separated in sections but the file is (almost) free formatted. The hash sign # starts a comment. The keywords and corresponding default values are summarized in the list presented below.

General settings

`population size` *SIZE(int)*
: Defines the maximum population size. $SIZE > 1$, default: $SIZE = 20$.

`min max queued structures` *MIN(int) MAX(int)*
: Defines the minimum and maximum number of the local optimization jobs present at any moment in the queue. Use $MIN = 1$ to have a 'standard' generation-based GA run. $MAX \geq MIN \geq 1$, defaults: $MIN = 1$, $MAX = 10$.

`optimized structures` *MAX(int)*
: Defines the total number of structures optimized during the run. Default: $MAX = 100$.

Initial structures generator settings

`gen ini max tries` *MAX(int)*
: Defines the maximum number of tries performed in order to generate a single initial structure model. Default: $MAX = 1000$.

`initial structures` *NUMBER(int)*
: Defines the total number of generated initial structures. Default: *NUMBER* $= 20$.

`atomic radii bounds` *A(float) B(float)*
: Defines the multipliers for atomic radii used to check whether two atoms are bound. Defaults: $A = 0.7$, $B = 1.2$.

`composition`
SYMBOL1(string) MIN1(int) MAX1(int)
SYMBOL2(string) MIN2(int) MAX2(int)
SYMBOL3(string) MIN3(int) MAX3(int)
...
`end composition`
: The composition block defines the desired atomic composition of the initial structures. *SYMBOL* is a chemical symbol of the given element. *MIN* and *MAX* define minimum and maximum number of atoms for that element. Only systems with constant composition during the GA run are currently available ($MAX = MIN$).

`seed file` *FILENAME(string)*
: Declares the file that contains seeds given in the internal `DoDo` format. Default: *FILENAME* = OLD_POPULATION.dodo.

`seeds amount` *NUMBER1(int) NUMBER2(int) NUMBER3(int)...*
: Defines how many initial species will be generated from each seed given in the `seed file`. The number of the values has to be the same as the count of the seeds given in the `seed file`, *i.e.* each seed in the `seed file` has to be accounted for.

Crossover and mutation settings

`constant composition mode`
: This line is a sanity check. If this line is present in the input file, *MIN* has to be equal to *MAX* for each *SYMBOL* in the `composition` section. Otherwise, an error is raised. Only this mode is currently available.

`crossover max tries` *MAX(int)*
: Defines the maximum number of tries performed in order to generate a single structure model. Default: $MAX = 1000$.

`fitness scaling` *FACTOR(float)*
: Defines the scaling factor α used in Eq. (5.5). Default: $FACTOR = 1.0$.

`mutation probability` *VALUE(float)*
: Defines the mutation probability of structures within the GA population. Default: $VALUE = 0.01$.

Selection settings

`selection RMS` *VALUE(float)*
: Defines the maximum root mean square of the distances between the corresponding atoms for which two structures are considered similar. Default: $VALUE = 0.3$.

`selection dstmax` *VALUE(float)*
: Defines the maximum distance [Å] used to seek for the corresponding atoms. Default: $VALUE = 0.3$.

Software control settings

`queueing system` *NAME(string)*
: Defines which job scheduling software interface should be used. $NAME =$ pbs or $NAME =$ hlrn, default: $NAME =$ pbs.

`time interval` *VALUE(int)*
: Defines how often [seconds] the GA checks the queuing system for the presence of submitted optimization jobs. Default: $VALUE = 600$.

`computational platform` *NAME(string)*
: Defines the computational platform (`TURBOMOLE` in combination with `Tmole`). $NAME =$ turbomole.

`job file` *NAME(string)*
: Defines the name of file which will be submitted to the queueing system. Default: $NAME =$ job.run.

`tmole file` *NAME(string)*
: This file is used by `Tmole2.0`. *NAME* given here HAS to agree with data given in `job file`.

5.8 Counterpoise-Corrections using the JOBBSSE Script

The shell script `jobbsse` controls and executes the automatic calculation of the counterpoise correction as it has been formulated by Boys and Bernadi (S. F. Boys and F. Bernardi, Mol. Phys., **19,** 553 (1970)) to estimate the Basis Set Superposition Error (BSSE). For a dimer, the cp-correction takes the form for the monomers A and B:

$$E_{AB}^{CP} = E_{AB} - (E_{A(B)} - E_A) - (E_{B(A)} - E_B)$$

Where parentheses denote ghost basis sets without electrons or nuclear charges. For a trimer `jobbsse` used by default the conventional so-called site-site functional counterpoise corrections:

$$E_{ABC}^{CP} = E_{ABC} - (E_{A(BC)} - E_A) - (E_{B(AC)} - E_B) - (E_{C(AB)} - E_C) \quad .$$

`jobbsse` works similar as the `jobex` script: it cycles through the SCF/DFT and, if needed, gradient and force relaxation programs and stops if either the maximum number of cycles is reached or the convergence criteria (change in the total energy, maximum norm of the gradient) are fulfilled. It does either only energy calculations or a full geometry optimization including up to three fragments. By default, the executable programs are taken from the load modules library within the `TURBOMOLE` directory.

Note that you need to set up the fragments (and possibly their symmetries using `define` in the geometry menu beforehand. The general structure of a `jobbsse` calculation is as follows:

1. `bsseenergy` is invoked to generate input files for `define`, which is then used to prepare the control files (including occupation numbers, initial guess MOs, etc.) for the different "ghost" and monomer calculations and shell scripts with commands for calculations on these fragments.

2. `jobbsse` cycles over the supermolecular complex and the fragments and computes the energies and, if requested, gradients for them. Then the counterpoise-corrected results are evaluated and written to the standard data groups ($energy and $grad).

3. For geometry optimizations one of the structure relaxation codes (`statpt` or `relax`) is invoked to update the coordinates and check for convergence. If the structure optimization is not converged `jobbsse` continues with the previous step.

Note, that counterpoise-corrected calculations with `jobbsse` are NOT as black-box as ordinary geometry optimizations with `jobex`. The input generated for the fragments are based on the default occupation numbers obtained from the EHT guess, default assignments for the frozen orbitals, memory, etc. Since this might be different from what is needed (or even fail), it is recommended to let `jobbsse` stop after the initial setup step using the flag `-setup` and to check carefully the assigned basis sets,

occupations number and subsystem symmetries. In particular, for MP2 or CC2 calculations with molecules containing not only the atoms H–Ar also the number of frozen orbitals should be checked, and if necessary corrected.

5.8.1 Options

Given a shell the usage is:

```
nohup jobbsse &
```

This command invokes cp-correction, and, if needed structure optimization using the default program `statpt`. Note, that the program needs to know which calculation is being done. Structure optimizations using program `relax` can be performed using `-relax` flag:

```
nohup jobbsse -opt -relax &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. `jobbsse` accepts the following arguments controlling the level of calculation, convergence criteria and many more (for example `nohup jobbsse -gcart 4 &`):

`-energy` *integer*	converge total energy up to $10^{(-<\text{integer}>)}$ Hartree (default: 6)
`-gcart` *integer*	converge maximum norm of Cartesian gradient up to $10^{(-<\text{integer}>)}$ atomic units (default: 3)
`-c` *integer*	perform up to *integer* cycles (default: 100)
`-gradient`	calculate the gradient as well
`-opt`	optimise the structure
`-relax`	use the `relax` program for force relaxation
`-level` *level*	define the optimization level, *level*=`scf`, `dft`, `mp2`, or `cc2` (default is `scf`). Note that the program needs this input! If the level is DFT, the grid will be automatically set to m4.
`-ri`	use RI modules `ridft` and `rdgrad` (fast Coulomb approximation) instead of `dscf` and `grad` as well as `rimp2` instead of `mpgrad`
`-l <path>`	employ programs from directory `<path>`
`-mem` *integer*	Is able to control the memory from outside `define` Note that if you did not define any memory, it is automatically set to 1 GB

`-trimer`	calculates, in case we have a trimer: Energy = ABC - AB(C) + AB - AC(B) + AC - BC(A) + BC rather than Energy = ABC - A(BC) + A - B(AC) + B - C(AB) + C (note that the first term neglects the BSSE in the dimer)
`-setup`	Interrupt calculation after the initial setup step to check and possibly correct the control files for the fragments and the supermolecule. To continue, start `jobbsse` without the `-setup` option.
`-help`	shows a short description of the commands above

5.8.2 Output

There will be an output written to file `bsse_out`. In this file, you will find all individual energies computed which were used to calculate the last cp-corrected energy. The same holds true for the last gradients, which are written to `grad_out`.

The convergence criteria and their current values are written out at the `not.converged` file. For the possible options to control convergence check the subsection for the optimization program used (`statpt`, which is used by default, or `relax`). Since for weak complexes the force constants for intra- and intermolecular bonds very strongly in magnitude, it is recommended to use whenever possible redundant internal coordinates.

5.9 Reaction Path Optimization

5.9.1 Background and Program structure

The goal of self-consistent optimization of the reaction path (RP) is usually to obtain an initial guess for Transition State Search or an approximation to the barrier. Methods that use reactant and product structure to compute the RP are often referred to as 'double-ended' methods or, if the the RP is discretized, 'chain-of-states' methods. [72–74]

The RP connects reactant and product, its highest point being the transition state. It is a steepest descent path, which means that its tangent $\mathbf{t}$ is always parallel to the gradient $\mathbf{g}$. The RP is in actual calculations discretized by a finite number of structures n. The tangents are parallel to the gradients for all structures $i = 1, .., n$. Assuming normalized tangents $\mathbf{t}_i^{\mathrm{T}}\mathbf{t}_i = 1$, this can be written as:

$$0 = (1 - \mathbf{t}_i\mathbf{t}_i^{\mathrm{T}})\mathbf{g}_i \tag{5.6}$$

Several approximations for the tangents $\mathbf{t}_i$ exist [74, 75], usually using finite difference schemes. The most common methods, the Nudged Elastic Band (NEB) [74, 75] and String Method (SM) [76] prevent the structures from 'sliding' down the reaction path towards products and reactants with additional springs or interpolation/redistribution algorithms. The method used here achieves equal spacing via constrained optimization assuming a quadratic potential. [77] An initial path is provided using a slight variation of the Linear Synchronous Transit [72].

The structure of the optimization is the same as in other TURBOMOLE structure optimizations: As the `jobex` script drives optimizations by calling `statpt`/`relax` as well the SCF and gradient modules.The `woelfling-job` script drives optimizations by calling `woelfling` as well as the SCF and gradient modules. The woelfling-job scripts reads the current path from file `path.xyz` which is the output of the `woelfling` program. `woelfling-job` then creates folders to run the calculations of each structure in, gathers coordinates and gradients, and then calls `woelfling` again.

The aim of RP optimization is usually not to optimize the RP to some accuracy, but to obtain an initial guess for a TS optimization. It is in general not possible to find a convergence criterion (and a corresponding threshold) that guarantees a good initial guess. The maximum number of iterations and the convergence threshold are therefore relatively high and tight. One can extract a TS guess also during the course of the optimization. If the TS search is successful (or not) you can stop (or restart) the RP optimization. Apart from simply killing the program you can add a 'stop' file in the (scratch) directory, in which the script runs. It will then terminate at the end of the current cycle and can easily be restarted.

5.9.2 Input Structure

Options can be modified using keywords in the $`woelfling` data-group. The most important options are:

```
$woelfling
 ninter           14
 riter 0
 ncoord            2
 align             0
 maxit          40
 dlst   3.00000000000000
 thr  1.0000000000000000E-004
 method q
```

The values above are the default values. If `$woelfling` is missing, it will be added during the first `woelfling` run and default values will be set. Most importantly, `ncoord` is the number of input structures provided, `ninter` is the number of structures to be used for discretization of the path and `maxit` is the number of cycles to run. If `align 0`, structures will be rotated/translated to minimize the cartesian distance, for `align 1` structures will be used as provided. Using `method qg` instead of `method q`, a reaction path will be grown as in the growing string method. To start a RP optimization you need to provide at least a reactant and a product structure `ncoord` ≥ 2. You may provide more structures if you have a guess for the reaction path. The input structures will be used to compute an initial guess with `ninter` structures that is then optimized. Reactant and product structure will stay fixed throughout the optimization. All structures *have to have* the same ordering of atoms!

The input structures are provided in a file `coords`, which contains merged `coord` files. All `ncoord` structures are given in the right order in the typical TURBOMOLE coord format.

5.9.3 How it works

Minimum Input/Quick and Dirty

1. Make a usual TURBOMOLE input using the coord file of either reactant of product structure.
2. Join the coord-files of reactant and product in a file `coords`.
3. Run `woelfling-job`
4. Check the output and the path (path.xyz) to extract a TS guess.

It is usually a good idea to check the initial path before starting the calculation. Once you have prepared the input, simply run `woelfling` directly and check `path.xyz`. If it looks reasonable, just run the `woelfling-job` script.

Unsuccessful Optimization If there is no reasonable TS guess or a frequency calculation does not give the correct number of imaginary frequencies, you can:

1. Check if have used the best structure as TS guess, maybe the structure with the highest energy is not the best.
2. Check if the RP has a reasonable amount of structures (if they are far apart, it is unlikely that a structure is close to the TS)
3. Check if the RP is reasonably converged (mean of rms($\mathbf{g}_i^{\perp}$) in output <`1.0d-3`; path is continuous in terms of energy and structure)
   (a) If it is not yet converged, converge it.
   (b) If it is not going to converge, provide useful structures for the initial guess or maybe use more structures for the path.

Parallelisation `woelfling-job` provides a few basic options to allow parallel calculation

`-mfile filename`
: specify a file that contains the names of the machines that should be used for parallel execution (machinefile format with one entry per CPU core). If the job is run in a PBS queuing system, the filename defined in the environment variable `$PBS_NODEFILE` will be used per default.

`-nthreads` n
: specify the number of threads (CPU cores) that should be used for the individual energy and gradient calculations. If not specified, each calculation will use all the cores speficied in the machinefile for a given machine name, one calculation will be started per unique machine name.

`-scrpath directory`
: specify a path to a local file system for scratch files. If not specified, the path defined in the environment variable `$TURBOTMPDIR` will be used instead, if set. If no path is given and `$TURBOTMPDIR` is not set, scratch files will be written into subdirectories of the job's working directory.

`woelfling-job` uses for parallel calculations (exclusively) the SMP-parallel binaries which will be invoked through ssh on the machines specified in the machinefile. If CPU cores on more than one machine are used, `woelfling-job` must be started in a working directory that is available on all machines. To avoid that I/O into a nfs filesystem cause a bottleneck, a scratch directory in a local file system (which must exist with the same name on all machine) should be specified with the `-scrpath` option or the `$TURBOTMPDIR` variable. On machines with a large number of cores the `-nthreads` option can be used to obtain a higher parallelization efficiency by starting more than one energy/gradient calculation (each with n cores) per machine.

Freezing of coordinates The current version of `woelfling` ignores fixed internal coordinates, only fixed cartesian coordinates are recognized.

Calculations with fixed cartesian coordinates require that the same atoms are marked as fixed in all start structures (e.g. first and last point of the pathway) and have in all start structures identical cartesian coordinates.

Restart

1. If you have stopped the calculation adding a 'stop' file, you can just run `woelfling-job` again.
2. If you have run the maximum number of cycles, just increase `maxit` and run `woelfling-job` again. It will then run from the old `maxit` to the new `maxit`.

If the calculation crashed 'on its own' it is likely that the SCF failed to converge - improve the corresponding options. If the optimization crashed in the middle of the run, it is most likely that it crashed during a SCF or gradient step, since basically all CPU time is spent there. In that case, remove at least the folder where the SCF and gradient program had been running when the program crashed. The files necessary for `woelfling` should be intact and you can simply restart it.

Restart - more details If the files are not intact, one can still use the optimized coordinates in one way or the other. There are basically three phases which are explicitly indicated by the number of iterations `riter` =:

0 `woelfling` reads `control`, `coords` to generate an initial guess `path.xyz`

1 `woelfling` reads `control`, `gradients` to compute an optimized `path.xyz`. Hessian are initialized and written to `hessians-new`

>1 `woelfling` reads `control`, `gradients`, `oldgradients` and `hessians` to update Hessians and compute an optimized `path.xyz`. Hessian are updated and written to 'hessians-new'.

Therefore repeated execution of `woelfling` will yield the same output (unless new energies/gradients are computed). The `woelfling-job` script takes care of the file-handling and should enable restart at any time. If files are damaged it will be hard to gather gradients and corresponding Hessian- and old gradient information. If you have an intact `gradients` or `oldgradients`-file (necessary condition: number of lines=$(3 + 2 \times \text{natoms}) \times$number of structures), name it `gradients`, set `riter` in the control-file to 1 and restart. If those file are not intact, you can extract whatever structure information you have obtained and use it to provide a better initial guess: Modify file `coords` and `ncoord` in the `control`-file accordingly, set `riter` to 0 and restart.

Chapter 6

Hartree–Fock and DFT Calculations for Molecular Systems

Energy and gradient calculations at the Hartree–Fock (HF) and DFT level can be carried out in two ways: `dscf` and `grad` perform conventional calculations based on four–center two–electron repulsion integrals (ERI's); `ridft` and `rdgrad` employ the RI–J approximation, as detailed below.

`dscf` and `grad` are modules for energy and gradient calculations at the HF and DFT level, which use an efficient semi–direct SCF algorithm. Calculation of the Coulomb and HF exchange terms is based on the conventional method employing four–center two–electron repulsion integrals (ERI's). These modules should be used for HF and DFT calculations with exchange-correlation functionals including HF exchange contribution, e.g. B3–LYP, if further approximations (RI–J) are to be avoided. All functionalities are implemented for closed–shell RHF and open–shell UHF reference wavefunctions. Restricted open shell treatments (ROHF) are supported on the HF level only, i. e. not for DFT.

The most important special features of the `dscf` and `grad` modules are:

- Selective storage of the most time consuming and frequently used integrals. The integral storage is controlled by two threshold parameters, `$thize` and `$thime`, related to integral size and computational cost.
- Efficient convergence acceleration techniques for energy calculations. They include standard methods for convergence acceleration (DIIS), which reduce the number of SCF iterations needed as well as methods to reduce the effort within each iteration when the calculation is almost converged (integral prescreening and differential density scheme).

`ridft` and `rdgrad` are modules for very efficient calculation of energy and gradient at the Hartree–Fock (HF) and DFT level [78]. Both programs employ the Resolution of the Identity approach for computing the electronic Coulomb interaction (RI–J). This approach expands the molecular electron density in a set of atom–centered auxiliary functions, leading to expressions involving three–center ERI's only. This usually leads to a more than tenfold speedup for non–hybrid DFT compared to the conventional method based on four–center ERI's (for example the `dscf` or `grad` module).

The combination of RI–J for Coulomb–interactions with a case–adapted conventional exchange treatment reduces the scaling behaviour of the (conventional) exchange evaluation required in HF–SCF and hybrid DFT treatments. Usage of `ridft` and `rdgrad` for HF and hybrid DFT is of advantage (as compared to `dscf` and `grad`) for larger systems, where it reduces computational costs significantly.

The most important special features of the `ridft` and `rdgrad` modules are:

- A very efficient semi-core algorithm for energy calculation. The most expensive three–center integrals are kept in memory which significantly reduces the computational time for small and middle sized molecules. The amount of stored integrals is controlled by simply specifying the amount of free memory using the keyword `$ricore`.
- Multipole accelerated RI for Coulomb (MARI–J) linear scaling ($O(N)$) method for large molecules. It significantly reduces calculation times for molecules with more than 1000 basis functions.

All algorithms implemented in `dscf`, `grad`, `ridft`, and `rdgrad` modules can exploit molecular symmetry for *all* finite point groups. Typically, the CPU time is proportional to $1/N_G$, where N_G is the order of the nuclear exchange group. Another important feature is a parallel implementation using the MPI interface.

Additionally `dscf` and `ridft` modules include the following common features:

- An UHF implementation [79] with automatic generation of optimal start vectors by solving the HF instability equations [80] in the AO basis (see the keyword `$scfinstab` for detailed information).
- Occupation number optimization using (pseudo-Fermi) thermal smearing.

RI-techniques can also be used for the Hartree–Fock exchange part of the Fock matrix (RI-HF). This is done by the ridft-module, if the keyword `$rik` is found in the `control` file. In this case ridft performs a Hartree–Fock-SCF calculation using the RI- approximation for both J and K, if suitable auxiliary basis sets (which differ from that used for fitting of the Coulomb part only) are specified. This is efficient only for comparably large basis sets like TZVPP, cc-pVTZ and larger.

HF-exchange can also be calculated semi-numerically [81]. The calculation of 4c-2e-Integrals is split into an analytical and a numerical part. The latter is evaluated on a dft-type integration grid. The semi-numerical calculation scales better with system size than RIK and is suitable for large molecules and large basis sets.

Prerequisites

Both `dscf` and `ridft` require the `control` file and starting orbitals obtained from the extended Hückel guess using `define`.

Energy calculations using `dscf` can be performed in a direct or semi-direct mode. In the direct mode all four-center ERI's are recalculated at each SCF iteration. The semi-direct mode uses a selective storage of the most time consuming and frequently used integrals. The amount of integrals stored is controlled by the keywords `$thize` and `$thime`, related to integral size and computational cost. The semi-direct mode requires a separate `dscf` statistics run to estimate the disk space needed for integral storage. The statistics run requires the keyword `$statistics dscf` to be present in the `control` file. It can be set either manually or using the tool `Stati`.

For `ridft` and `rdgrad` following additional prerequisites are required:

1. An auxiliary basis defined in the data group `$jbas`. This group is created automatically when using `ri` menu of `define`.
2. The maximum core memory the program is allowed to allocate should be defined in the data group `$ricore`; the recommended value is 75–85% of the available (physical) core memory.
3. Calculations using MARI-J method require the keyword `$marij`.
4. For RI-HF-calculations auxiliary bases defined in the data group `$jkbas` are needed. This group is created by the `rijk` menu in `define`.

How to Perform a Calculation

Single point calculations
: Call the `dscf` or `ridft` program after running `define`.

Geometry optimizations and molecular dynamics
: For HF or DFT calculations using `dscf` and `grad` simply invoke `jobex`. For DFT calculations using `ridft` and `rdgrad` type `jobex -ri`; see Section 5.1 for additional options and parameters for geometry optimizations and *ab initio* molecular dynamics calculations.

Special Options for Open-Shell Systems

Flipping of spins at a selected atom is possible via define. This requires converged unrestricted HF (UHF) or unrestricted Kohn–Sham (UKS) molecular orbitals and no symmetry (C_1). Go to the molecular orbital section of `define`and use the spin option. Note that this will localize the orbitals, assign them to the atoms and give the user the possibility to choose atoms at which alpha-orbitals are moved to beta orbitals, or vice versa. This is useful for spin-broken start orbitals, but not for spatial symmetry breaking.

If you encounter serious spin contamination in UHF or UKS calculations, you may use an additional spin constraint in the SCF procedure. This can be done with they keyword `spin constraint` followed by a value for the constraint. We refer to [82] and the Supporting Information of [83] for details.

6.1 Background Theory

In Hartree–Fock theory, the energy has the form,

$$E_{HF} = h + J - K + V_{nuc}, \tag{6.1}$$

where h is the one-electron (kinetic plus potential) energy, J is the classical Coulomb repulsion of the electrons, K is the exchange energy resulting from the quantum (fermion) nature of electrons, and V_{nuc} is the nuclear repulsion energy.

In density functional theory, the exact Hartree–Fock exchange for a single determinant is replaced by a more general expression, the exchange-correlation functional, which can include terms accounting for both exchange energy and the electron correlation which is omitted from Hartree–Fock theory. The DFT energy is expressed as a functional of the molecular electron density $\rho(\mathbf{r})$,

$$E_{DFT}[\rho] = T[\rho] + V_{ne}[\rho] + J[\rho] + E_x[\rho] + E_c[\rho] + V_{nuc}, \tag{6.2}$$

where $T[\rho]$ is the kinetic energy, $V_{ne}[\rho]$ is the nuclei-electron interaction, $E_x[\rho]$ and $E_c[\rho]$ are the exchange and correlation energy functionals.

The exchange and correlation functionals normally used in DFT are integrals of some function of the density and possibly the density gradient. In addition to pure DFT methods, `dscf` and `grad` modules support hybrid functionals in which the exchange functional includes the Hartree–Fock exchange, e.g. B3-LYP.

6.2 Exchange-Correlation Functionals Available

The following exchange-correlation functionals are available:

- LDAs: S-VWN, PWLDA
- GGAs: B-VWN, B-LYP, B-P, PBE, revPBE, SOGGA11, KT3
- MGGA: TPSS, SCAN, r2SCAN, r4SCAN, r++SCAN, TASK, revTPSS, PKZB, Tao-Mo, M06-L, M11-L, MN12-L, MN15-L
- hybrid functionals: BH-LYP, B3-LYP, PBE0, TPSSh, revTPSSh, r2SCANh, r2SCAN0, r2SCAN50, BMK, B97M(-V), SOGGA11-X, M05, M06, M06-2X, MN12, MN15, PW6B95
- range-separated hybrid functionals: CAM-B3LYP, HSE06, M11, revM11, MN12-SX, ωB97(X)(-V), ωB97M-V, LRC-ωPBE, LC-ωPBE class, CAM-QTP-00, CAM-QTP-01, CAM-QTP-02
- double–hybrid functional: B2-PLYP, B2GP-PLYP and XYG3 (energy calculations only, geometry optimization with numerical gradient)
- local hybrid functionals: Lh07t-SVWN, Lh07s-SVWN, Lh12ct-SsirPW92, Lh12ct-SsifPW92, Lh14t-calPBE, LH20t, PSTS, mPSTS, LHJ14, LHJ-HF, LHJ-HFcal, TMHF, TMHF-3P

For EXX and LHF, see Chapter 21

The XCFun library (Arbitrary-Order Exchange-Correlation Functional Library) by Ulf Ekström and co-workers has been included [84] and some of the functionals implemented there can now be utilized. Among them are the empirically fitted MGGAs M06 and M06-2X from the Truhlar group [85]. XCFun functionals are available for energy, gradient, vibrational frequencies, and TDDFT excited state energy calculations - with and without RI approximation. For details and the license of XCFun please refer to its web site https://github.com/dftlibs/xcfun

The LibXC 6.2.2 library has been included [86] and some of the functionals implemented there can now be utilized directly. Among them are the empirically fitted MGGAs from the Truhlar group as well as the range-separated wB97(X) group of Head-Gordon [87] and the r2SCAN hybrids or the CAM-QPT functional family. Note that the usage of LibXC is stated in the output with the required references. For details and the license of LibXC, please refer to its web site of the project https://tddft.org/programs/libxc/

See the next chapters for available functionals from LibXC and XCFun. For range-separated hybrid functionals see the notes and restrictions in the corresponding section.

In detail, the Turbomole own functional library consists of:

- The Slater–Dirac exchange functional only (S) [88, 89].

- The 1980 correlation functional (functional V in the paper) of Vosko, Wilk, and Nusair only (VWN) [90].
- A combination of the Slater–Dirac exchange and Vosko, Wilk, and Nusair 1980 (functional V) correlation functionals (S-VWN) [88–90].
- The S-VWN functional with VWN functional III in the paper. This is the same functional form as available in the Gaussian program [88–90].
- A combination of the Slater–Dirac exchange and Perdew-Wang (1992) correlation functionals [88, 89, 91].
- A combination of the Slater–Dirac exchange and Becke's 1988 exchange functionals (B88) [88, 89, 92].
- Lee, Yang, and Parr's correlation functional (LYP) [93].
- The B-LYP exchange-correlation functional (B88 exchange and LYP correlation functionals) [88, 89, 92, 93].
- The B-VWN exchange-correlation functional (B88 exchange and VWN (V) correlation functionals) [88–90, 92].
- The B-P86 exchange-correlation functional (B88 exchange, VWN(V) and Perdew's 1986 correlation functionals) [88–90, 92, 94].
- The Perdew, Burke, and Ernzerhof (PBE) exchange-correlation functional [88, 89, 91, 95].
- The Tao, Perdew, Staroverov, and Scuseria functional (Slater–Dirac, TPSS exchange and Perdew-Wang (1992) and TPSS correlation functionals) [88, 89, 91, 96].
- The Strongly Constrained and Appropriately Normed (SCAN) meta-GGA functional [97].
- The regularized-restored SCAN (r2SCAN) meta-GGA functional [98].
- The r2SCAN-3c meta-GGA functional including Grimme dispersion and basis-set superposition error corrections (-3c) [99].
- TMHF first-principles local hybrid functional
- TMHF-3P modified version of TMHF with less parameters

For the SCAN functional, be advised that the convergence of the total energy and energy gradients with respect to the radial grid used to integrate the exchange-correlation energy and potential is slower than previous non-empirical functionals [100]. A radial grid size of 40 is typically needed to converge the total energy (`dscf` or `ridft`) to μH accuracy, while a radial grid size of 50 is typically needed to converge the energy gradients (`grad` or `rdgrad`). Convergence with respect to the

angular grids remains unchanged. The radial grid size can be set in the control file through the `radsize` keyword.

For details on the integration grids, please see Sec. 23.2.10. Generally, gridsize 3 is recommended for (time-dependent) DFT calculations. The multiple grids such as gridsize m3 or m4 are often sufficient for energies and geometry gradients. For hybrid density functionals, the multiple grids do not result in a notable reduction of the computational costs and the standard grids are a better choice.

For the r2SCAN and r2SCAN-3c functionals used for geometry optimizations a gridsize of m4 is usually sufficient if the `radsize` keyword is set to 8.

Additionally, for all four modules (`dscf`, `grad`, `ridft`, and `rdgrad`) the following hybrid functionals are available (a mixture of Hartree–Fock exchange with DFT exchange-correlation functionals):

- The BH-LYP exchange-correlation functional (Becke's half-and-half exchange in a combination with the LYP correlation functional) [88, 89, 92, 93, 101].
- The B3-LYP exchange-correlation functional (Becke's three-parameter functional) with the form,

$$0.8S + 0.72B88 + 0.2HF + 0.19VWN(V) + 0.81LYP \tag{6.3}$$

  where HF denotes the Hartree-Fock exchange [88, 89, 92, 93, 102].
- The B3-LYP exchange-correlation functional with VWN functional V in the paper. This is the same functional form as available in the Gaussian program.
- The 1996 hybrid functional of Perdew, Burke, and Ernzerhof, with the form,

$$0.75(S + PBE(X)) + 0.25HF + PW + PBE(C) \tag{6.4}$$

  where PBE(X) and PBE(C) are the Perdew–Burke–Ernzerhof exchange and correlation functionals and PW is the Perdew–Wang correlation functional [88, 89, 91, 95, 103].
- The TPSSH exchange-correlation functional of Staroverov, Scuseria, Tao and Perdew with the form,

$$0.9(S + TPSS(X)) + 0.1HF + PW + TPSS(C) \tag{6.5}$$

  where HF denotes the Hartree–Fock exchange [88, 89, 91, 96, 104].

6.2.1 Double-Hybrid Functionals and Adiabatic Connection Models

In the fifth-rung of the DFT Jacob-ladder we find functionals which include the Kohn-Sham second-order perturbation theory correlation energy (KS-PT2) [105, 106].

$$E_c^{KS-PT2} \approx \frac{1}{2} \sum_{ia} \sum_{jb} \frac{(ia|jb)[(ia|jb) - (ib|ja)]}{e_i + e_j - e_a - e_b}. \tag{6.6}$$

which is a Møller-Plesset like term based on the KS orbitals. Note that the full E_c^{KS-PT2} includes also a singly-excited term [107], which is zero in the case of HF ground-state, and it is also neglected here.

The Double-Hybrid Functionals (DHDF) [108] include a linear mixing of the E_c^{KS-PT2} energy, in addition to a non-local exchange contribution (as in conventional hybrid-GGAs) . The mixing is described by two empirical parameters a_x and a_c in the following manner:

$$\begin{aligned} E_{xc}^{DHDF} &= (1-a_x)E_x^{GGA} + a_x E_x^{HF} \\ &+(1-a_c)E_x^{GGA} + a_c E_c^{KS-PT2}, \end{aligned} \tag{6.7}$$

where E_c^{GGA} is the energy of a conventional exchange functional, E_c^{GGA} is the energy of a conventional correlation functional and E_x^{HF} is the Hartree-Fock exchange. The method is self-consistent only with respect to the first three terms in Eq. (6.7): a SCF using a conventional hybrid-GGA is performed first and using these orbitals E_c^{KS-PT2} is evaluated afterwards and added to the total energy. Thus an MP2 calculation is required after the DFT step to get the correct DHDF energy.

In `TURBOMOLE` the following DHDFs are currently supported:

B2-PLYP: In the B2-LYP functional [109] the B88 exchange [92] and LYP correlation [93] are used with the parameters $a_x = 0.53$ and $a_c = 0.27$. Due to the relatively large Fock-exchange fraction, self-interaction error related problems are alleviated in B2-PLYP while unwanted side effects of this (reduced account of static correlation) are damped or eliminated by the PT2 term.

In the following options/restrictions in the present version of this method:

- single point calculations only (computed with the DSCF/RIDFT and RIMP2/RICC2 modules).
- UKS treatment for open-shell cases.
- can be combined with *resolution-of-identity* approximation for the SCF step (RI-JK or RI-J option).
- can be combined with the dispersion correction (DFT-D method, s_6(B2-PLYP)=0.55).

How to use B2-PLYP:

- during preparation of your input with DEFINE select `b2-plyp` in the DFT menu.
- carry out a DSCF run. Prepare and run a RI-MP2 calculation with either RIMP2 or RICC2 program modules.
- the RI-MP2 program directly prints the B2PLYP energy if this functional has been chosen before

Or use the `b2plypprep` script to setup up the calculation.

- define coord and basis set
- (optional: switch on ri or rijk and define jbasis or jkbasis)
- run b2plypprep
- run DSCF (or RIDFT) and RICC2

Or use the `hfacm` script.

B2GP-LYP The B2GP-PLYP [110] is very similar to the B2-PLYP functional but the coefficients are $a_x = 0.65$ and $a_c = 0.36$. The B2GP-PLYP is supported by the `hfacm` script.

XYG3: For the XYG3 functional [111] the following steps are required

i) A SCF calculation with the b3-lyp functional.

ii) Setting the XYG3 in the control file.

iii) One scf-step with DSCF or RIDFT to extract the required XYG3 energy.

iv) A final MP2 run.

The calculations for steps ii) and iii) can be carried out with the `xyg3` scripts, which print out the total energy and update the `energy` file.

Or use the `hfacm` script described hereafter.

Hartee-Fock Adiabatic Connection Model

The Hartee-Fock Adiabatic Connection Model (HFACM) [112] mixes the MP2 correlation energy with semilocal functionals for the strong-correlation (SC) limit [113] which are indicated as W_∞ and W'_∞. The HFACM theory is based on the exact HF ground-state. The general expression for the XC energy can be written as:

$$E_{xc}^{HFACM} = F(E_c^{MP2}, E_x^{HF}, W_\infty, W'_\infty) \tag{6.8}$$

where F is a complicated *non-linear formula*, with the following two limits:

$$E_{xc}^{HFACM} \rightarrow E_x^{HF} + E_c^{MP2} \quad \text{for } E_c^{MP2} \rightarrow 0 \tag{6.9}$$

$$E_{xc}^{HFACM} \rightarrow W_\infty \quad \text{for } E_c^{MP2} \rightarrow -\infty \tag{6.10}$$

Thus HFACMs can be also applied to metallic and strong-correlated systems with vanishing energy-gap, where DHDFs fail.

Calculations with an HFACM functionals require:

i) an HF ground-state,

ii) a DFT step for the calculation of the SC functionals,

iii) a MP2 run.

All calculations can be done with the `hfacm` script which also supports all the DHDFs. The `hfacm` script requires:

- a converged SCF calculation (HF for ACMs or DFT for DHDFs), but only with the RI-J (or RI-JK) approach.
- the selection of the MP2 model in the $ricc2 group and the specification of the $cbas auxiliary basis set.

Main options for the `hfacm` script are:

`-f or -formula` *string*	select the formula among *mp2, b2-plyp, xyg3* ,*isi, revisi, spl, lb* ,*spl2, mpacf1*. Default is *isi*.
`-w or -wfunc` *string*	select the SC functional among *pc, mpc, hpc*. Default is *hpc*.
`-scf`	rerun calculation for scf orbitals
`-s`	rerun all calculations
`-ro`	read previous outputs and recompute energies. Useful to check different ACM formulas.
`-help`	shows a short description of the commands above

One of the most know ACM functional is the Interaction Strength Interpolation (ISI). [114, 115] The most recent ACM functionals are *spl2* and *mpacf1* [116], which work very well for dispersion energies (without the addition of any empirical dispersion correction). The most recent SC functional is the Harmonium point-charge-plus-continuum (hpc) [117].

Options can also be specified in the control file:

```
$hfacm
  formula <formula>
  wfunc <wfun>
```

which is read and/or set by the `hfacm` script.

The output of the `hfacm` script (with default options) contains (e.g. for the water molecule):

```
RESULTS:
 ACM-isi-hpc Correlation energy:                  -0.2882062729
 ACM-isi-hpc Exchange-correlation energy:         -9.1330417810
 ACM-isi-hpc Total energy:                       -76.3310361448
```

For the B2-PLYP functional the output contains:

```
RESULTS:
 B2-PLYP Correlation energy:                    -0.3795686158
 B2-PLYP SCF Correlation energy:                 -0.1034395676
 B2-PLYP Exchange-correlation energy:            -9.2140304015
 B2-PLYP Total energy:                         -76.4193775803
```

where the "B2-PLYP Correlation" is the difference with the HF total energy (with B2-PLYP orbitals), whereas "B2-PLYP SCF Correlation" is the difference with the B2-PLYP SCF total energy, which is added as "MP2" correction to the *energy* file.

Geometry optimization can be done using a numerical gradient, i.e.

```
jobex -ri -level NumGrad
```

using

```
$numgrad
 level hfacm
```

Interaction energies of fragments can be computed with the `hfacm_int` script, which is based on the `jobbsse` functionality. The `hfacm_int` options are the same of the `hfacm` script. The `hfacm_int` script requires

- a converged SCF calculation (HF for ACMs or DFT for DHDFs), but only with the RIJ or RIJK approach.
- the selection of the MP2 model in the $ricc2 group and the specification of the $cbas auxiliary basis set.
- Spefication of fragments and the `jobbsse` run with the RI approach.

The output of the `hfacm_int` script (with default options) contains (e.g. for the He-Ne):

```
=== FINAL INTERACTION RESULTS ===
with Size-consistency Correction (SCC)

ACM-isi-hpc-SCC Correlation interaction energy:        -0.0000640020
ACM-isi-hpc-SCC XC interaction energy:                 -0.0001725929
ACM-isi-hpc-SCC Total interaction energy:              -0.0000382602
```

The Size-consistency Correction (SCC) [118] is required as HFACM are non-linear formulas. For DHDFs, SCC is zero.

6.2.2 Local Hybrid Functionals

Local hybrid functionals [119] feature an admixture of the exact-exchange energy density to a (semi-)local exchange energy density in real space managed by a local mixing function (LMF). They can therefore be viewed as a more flexible generalization of global hybrid functionals.

So far, local hybrid functionals are available in the modules `ridft` [120], `grad`, `rdgrad` [121], `escf` [37, 122–125], `egrad` [126], and `mpshift` [125, 127]. For the calculation of non-standard exact-exchange integrals, a semi-numerical integration scheme [128] is used. The screening procedure is outlined in [124] and the parallelization is described in [124] and [125]. The usual DFT grids are employed for this task and small grids are thus recommended if possible. In `escf`-calculations, a larger amount of main memory is required for the semi-numerical integration. Therefore, additional main memory of approximately $0.0004 \cdot N_{BF}^2$ MB should be provided (for each CPU).

The following local hybrid functionals are available so far:

- Lh07t-SVWN: [129] Slater-Dirac exchange [88, 89] and VWN [90] with t-LMF (prefactor of 0.48)
- Lh07s-SVWN: [130] Slater-Dirac exchange [88, 89] and VWN [90] with s-LMF
- Lh12ct-SsirPW92: [131] Slater-Dirac exchange [88,89] and self-correlation-reduced PW92 correlation with common t-LMF (prefactor of 0.646)
- Lh12ct-SsifPW92: [131] Slater-Dirac exchange [88, 89] and self-correlation-free PW92 correlation with common t-LMF (prefactor of 0.709)
- Lh14t-calPBE: [132] PBE exchange and correlation [88, 89, 91, 95] with t-LMF (prefactor of 0.5) and pig1 calibration [133]
- LH20t: [134] PBE exchange and refitted B95 correlation [88,89,91,95,135] with t-LMF (prefactor of 0.751) and pig2 calibration [133]
- PSTS and mPSTS: [125,136] (modified) functional of Perdew, Staroverov, Tao, and Scuseria (PSTS-LMF). Use option mpsts-noa2 for closed-shell systems.
- LHJ14: [125, 137] Johnson's local hybrid functional based on the correlation length (z-LMF).
- LHF-HF: [138] Reparameterized version of LHJ14 with Becke95 correlation.
- LHF-HFcal: [138] LHJ-HF with calibration function.
- TMHF: [138] First-principles local hybrid functional of Holzer and Franzke.
- TMHF-3P: [138] Alternative version of TMHF with 3 theoretically derived parameters.

D3 parameters for the t-LMF based local hybrid functionals have been implemented. [139] PSTS and mPSTS use the D3 parameters of TPSSh. Note that Lh14t-calPBE is not available in `egrad` and requires larger numerical grids in `escf`-calculations.

For gradient calculations it is strongly recommended to use the `weight derivatives` keyword in the `$dft` data group to avert numerical inaccuracies.

6.2.3 Exchange-Correlation Functionals from LibXC library

The LibXC library is taken from https://tddft.org/programs/libxc/

The current `TURBOMOLE` version uses a LibXC 6.2.2. that has been adapted and checked for the provided functionals. All listed functionals support up to 3rd derivatives.

For some functionals included in LibXC shortcuts have been set. If a shortcut is present LibXC functionals can be similar to `TURBOMOLE` own functionals

```
$dft
  functional wb97x
```

Shortcuts are available for the following functionals:

```
wb97  -- wb97 range-separated hybrid GGA
wb97x -- wb97x range-separated hybrid GGA
wb97x-v -- wB97X range-separated hybrid GGA with VV10 non-local correlation
wb97m-v -- wB97X range-separated hybrid metaGGA with VV10 non-local correlation
wb97x-d -- wB97X range-separated hybrid metaGGA with dispersion correction
sogga11 --  sogga11 GGA
sogga-11x -- sogga-11x hybrid GGA
m05 -- M05 hybrid metaGGA
m05-2x -- M05 hybrid metaGGA with doubled exact exchange
m06 -- M06 hybrid metaGGA
m06-2x -- M06 hybrid metaGGA with doubled exact exchange
m06-l -- M06-L metaGGA
m11 -- M11 hybrid metaGGA
m11-l -- M11-L metaGGA
revM11 -- revised M11 hybrid metaGGA
mn12-l -- MN12-L metaGGA
mn12-sx -- MN12-SX short-range-separated metaGGA
mn15 -- MN15 hybrid metaGGA
mn15-l -- MN15-L metaGGA
pkzb -- Perdew-Kurth-Zupan-Blaha metaGGA
```

```
task -- ultra-nonlocal metaGGA (TASK exchange + PW92 correlation)
bmk  -- BMK hybrid metaGGA
scan-libxc -- SCAN metaGGA functional
scan0 -- hybrid metaGGA functional based on SCAN
cam-b3lyp -- CAM-B3LYP range-separated hybrid GGA
tuned-cam-b3lyp -- CAM-B3LYP range-separated hybrid GGA tuned for excitations
cam-qtp-00 -- CAM-QTP-00 functional
cam-qtp-01 -- CAM-QTP-01 functional
cam-qtp-02 -- CAM-QTP-02 functional
hse06-libxc -- HSE06 short-range-separated GGA
lr-wpbe -- range-separated range-separated GGA based on PBE
lrc-wpbeh -- long-range corrected range-separated hybrid GGA based on PBE
mpsts -- modified PSTS functional
mpsts-noa2 -- modified PSTS functional for closed-shell systems
lhj14 -- Johnson's local hybrid functional
kt3 -- third generation Keal-Tozer functional for NMR
r2scanh -- r2SCAN hybrid with 10% exact exchange
r2scan0 -- r2SCAN hybrid with 25% exact exchange
r2scan50 -- r2SCAN hybrid with 50% exact exchange
tao-mo -- metaGGA of Tao and Mo
tmhf -- Tao-Mo-Holzer-Franzke
tmhf-3p -- 3 parameter Tao-Mo-Holzer-Franzke
```

Since TURBOMOLE V7.5 certain range-separated functionals can be modified using the following functional shortcuts and specifying the according parameters:

```
lc-wpbe_own omega
cam-b3lyp_own alpha beta omega
lrc-wpbeh_own alpha beta omega
hse_own alpha beta omega
```

alpha, beta, omega are real numbers. *alpha* specifies the amount of exact exchange in the short-range limit, while in the long-range limit *alpha+beta* is used. The screening parameter *omega* determines how fast the long-range limit will be obtained and is defined as usual. Note that *beta* may be negative, converting a long-range corrected into a short-range corrected functional.

Other functionals from the LibXC may be accessed using the index numbers from LibXC Readme or from the Turbomole user forum. To trigger the usage of LibXC functionals, use the keyword `libxc` in the `$dft` section.

```
$dft
```

```
functional libxc <index_of_main_exchange_functionals>
functional libxc add <no. of add. func.> <index_of_add_funcs>
functional libxc factors fac1 fac2 ... fac <no. of func.>
functional libxc set-rangesep alpha beta mu
functional libxc set-hybrid exx
```

Apart from the first line all following lines are optional. *fac1, fac2...* are real numbers specifying the weight of each component of the defined functionals in *exactly* the order they have been specified. *fac1* refers to the main functionals, while fac2 etc. refer to the additional functionals. If more than one functionals is specified set-rangesep and set-hybrid always refer to the main functionals given in the first line. If range-sep is specified the main functional must support range-separation, otherwise the program will stop.

The M11 hybrid metaGGA for example may be specified by the following lines

```
$dft
  functional libxc 297
  functional libxc add 1 76
```

In the first line the number (297) specifies the index number of the exchange part of the M11 functional. As LibXC has stores the definition of a functional (amount of HF exchange, range-separation parameters etc.) on the exchange part of a functional it is mandatory to first specify the exchange part and add the correlation part. The second line then specifies how many and which further functional parts are added. The first number (1) specifies that one additional part is used. The number (76) is the index number of the correlation part of the M11 functional. A maximum of three additional X/C functionals may be specified. If exchange and correlation are specified together then the second line can be skipped. A example for this case is the Keal-Tozer 1 functional with index number (167):

```
$dft
  functional libxc 167
```

The functionals described in this section can be used in all modules of `TURBOMOLE` earlier limitations have been removed. This also applies to self-defined functionals.

Upon request a dynamically linked `TURBOMOLE` version which allows to use your own version of LibXC may be requested from support@turbomole.com.

Using a dynamically linked version one has to take special care of the second derivatives of metaGGAs, where the Fortran interface of LibXC is inconsistent. In `TURBOMOLE` the following ordering of second derivatives in the mGGA case is expected: v2rho2, v2sigma2, v2lapl2, v2tau2, v2rhosigma, v2rholapl, v2rhotau, v2sigmalapl, v2sigmatau, v2lapltau. This does only affect second derivatives of metaGGAs and may change upon future versions.

6.2.4 Exchange-Correlation Functionals from XCFun library

The XCFun library is taken from: https://github.com/dftlibs/xcfun

The current TURBOMOLE version uses XCFun 1.99 and enables the usage of individual mixtures of the available exchange and correlation functionals.

To trigger the usage of XCFun functionals, use the keyword `xcfun` in the `$dft` section:

```
$dft
  functional xcfun set-gga
  functional xcfun <name1> <factor1>
  functional xcfun <name2> <factor2>
```

In addition to the name of the functional, it is necessary to tell TURBOMOLE whether the used functional is of GGA or MGGA type. Pure LDA functionals are currently not supported.

Available settings are:

- `functional xcfun set-gga` – sets a GGA functional
- `functional xcfun set-mgga` – sets a meta-GGA functional
- `functional xcfun set-hybrid 0.2` – defines a hybrid functional with a portion of 0.2 of Hartree-Fock exchange

Add the switch for either GGA or meta-GGA but not both in the same input!

List of available XCFun functionals (copied from XCFun documentation), in arbitrary order:

```
slaterx  -- Slater exchange
beckex -- Becke exchange
beckecorrx -- Becke GGA exchange
ktx -- Keal-Tozer exchange
pbex -- Perdew-Burke-Ernzerhof exchange
tpssx -- TPSS original exchange functional
m05x -- Truhlar M05 exchange
m05x2x -- Truhlar M05-2X exchange
m06x -- Truhlar M06 exchange
m06x2x -- Truhlar M06-2X exchange
m06lx -- Truhlar M06L exchange
m06hfx -- M06-HF Meta-Hybrid Exchange Functional
b97x -- B97 exchange
b97_1x -- B97-1 exchange
```

```
b97_2x -- B97-2 exchange
b97_dx -- B97-D exchange ($disp3 in addition required)
optx -- OPTX Handy & Cohen exchange GGA exchange functional
optxcorr -- OPTX Handy & Cohen exchange -- correction part only
brx -- Becke-Roussells exchange with jp dependence
brxc -- Becke-Roussells exchange and correlation with jp dependence
pw86xtot -- Perdew-Wang 86 GGA exchange including Slater part
pw91x -- Perdew-Wang 1991 GGA Exchange Functional
ldaerfx -- Short range exchange LDA functional
cam-b3lyp-xcfun -- range-separated hybrid version of B3LYP

vwn5c -- VWN5 correlation
vwn3c -- VWN3 correlation
lypc  -- LYP correlation
pw91c -- PW91 Correlation
pw92c -- PW92 LDA correlation
pz81c -- PZ81 LDA correlation
pbec -- PBE correlation functional
vwn_pbec -- PBE correlation functional with VWN LDA correlation
spbec -- Simplified PBE correlation functional for use with the SSB functionals
tpssc -- TPSS original correlation functional
revtpssc -- Revised TPSS correlation functional
p86c -- P86C GGA correlation
m05c -- M05 Meta-Hybrid Correlation Functional
m05x2c -- M05-2X Meta-Hybrid Correlation Functional
m06c -- M06 Meta-Hybrid Correlation Functional
m06lc -- M06-L Meta GGA Correlation Functional
m06x2c -- M06-2X Meta-Hybrid Correlation Functional
csc -- Colle-Salvetti correlation functional
brc -- Becke-Roussells correlation with jp dependence
b97_1c -- B97-1 correlation
b97_2c -- B97-2 correlation
b97_dc -- B97-D correlation ($disp3 in addition required)
b97c -- B97 correlation
ldaerfc -- Short range correlation LDA functional

pw91k -- PW91 GGA Kinetic Energy Functional
btk -- Borgoo-Tozer kinetic energy functional
```

```
tfk -- Thomas-Fermi Kinetic Energy Functional
vW  -- von Weizsaecker kinetic energy
```

Some common functionals are pre-defined in XCFun and their individual parts do not have to be set manually. Those aliases can be directly used as names with a following factor of 1.0:

blyp, pbe, bp86, kt1, kt2, kt3, pbe0, b3lyp, m06, m06-2x, m06L, b3lyp-g, b3p86, b97, b97d, olyp and some more.

Note that if the functional needs a portion of HF exchange, this has to be added manually in the control file using `functional xcfun set-hybrid` <number>

Example for B3-LYP using VWN3 instead of VWN5:

```
$dft
  functional xcfun set-gga
  functional xcfun b3lyp-g 1.0
  functional xcfun set-hybrid 0.2
```

CAM-B3LYP from the XCfun may be used by using the shortcut:

```
$dft
  functional cam-b3lyp
```

The functionals described in this section can be used for ground state energies, gradients and frequency calculations as well as TDDFT spectra. TDDFT analytic gradients are not yet supported up to `TURBOMOLE` version 7.4, please use the `TURBOMOLE` own functionals instead. Later versions of `TURBOMOLE` will support TDDFT analytic gradients from XCfun functionals.

Notes about range-separated hybrid functionals

`TURBOMOLE` now support RSHs in all modules. With RSHs the use of RI-K is not supported. Seminumerical ($senex etc.) exchange is fully supported for RSHs.

Notes about VV10 non-local correlation dependent functionals

For functionals using VV10 non-local correlation the following hints should be noted: VV10 can be included non-self-consistently (default) or self-consistently (keyword $doscnl). For energies non-self-consistent VV10 (default) usually is sufficient, while for geometry optimizations VV10 should be included self-consistently. `define` will automatically add $doscnl if the ωb97x-V or ωb97m-V functionals are selected. `escf`,

`egrad`, `mpshift` and `aoforce` simply ignore VV10 non-local contributions which should be an excellent approximation in most cases. If $doscnl is added we *strongly* recommend the use of m-grids (e.g. m3 which is the default).

Notes about DFT-D3, gCP and functionals using those corrections

For details about the options of DFT-D3 please see section 6.7.

In the original `TURBOMOLE` implementation of the B97-D functional only energy and gradient calculations are possible due to missing higher derivatives of the functional itself. Using the XCFun version of B97-D, analytic 2nd derivatives using `aoforce`and TDDFT excited state energies are possible. The names in the `$dft` section are b97-d for the `TURBOMOLE` own version and b97d for the XCFun version. However, the total energies of those two flavours are slightly different due to the fact that the parameters used are either the originally published ones (`TURBOMOLE`) or re-computed (XCFun). For properties like geometries and frequencies the differences are negligible, but one should not mix the total energies.

The PBEh-3c functional needs, besides the functional name `pbeh-3c` also DFT-D3 dispersion correction including the three-body term and geometrical counterpoise correction method called gCP. For details see: Stefan Grimme, University Bonn. In order to get the full version of PBEh-3c, your control file has to include:

```
$dft
   functional pbeh-3c
   gridsize   m4
$disp3 -bj -abc
```

Note: gcp is automatically added if pbeh-3c functional is used, but the D3 part has to be switched on manually by adding `$disp3` as given above. It is, however, sufficient to use `$disp3 -bj` since the abc term is added automatically for PBEh-3c and B97-3c.

To use HF-3c (R. Sure, S. Grimme, J. Comput. Chem. 2013, 34, 1672–1685), an input without DFT functional (and without `$dft` keyword) but with DFT-D3 correction is required. Calculations with and without RI are possible to perform, but due to the very small basis set non-RI calculations are usually as efficient as those with RI. Note that it is important to use the 'Minix' basis set and to select hf-3c as functional name for DFT-D3:

```
$disp3 -bj func hf-3c
```

The gCP correction (H. Kruse, S. Grimme, J. Chem. Phys. 2012, 136, 154101) will by default be added to the DFT-D3 correction term if pbeh-3c or hf-3c is selected.

6.3 Restricted Open-Shell Hartree–Fock

6.3.1 Brief Description

The spin-restricted open-shell Hartree–Fock method (ROHF) can always be chosen to systems where all unpaired spins are parallel. The `TURBOMOLE` keywords for such a case (one open shell, triplet e_g^2) are:

```
$open shells  type=1
  eg     1                (1)
$roothaan   1
  a=1  b=2
```

It can also treat more complicated open-shell cases, as indicated in the tables below. In particular, it is possible to calculate the $[xy]^{singlet}$ case. As a guide for expert users, complete ROHF `TURBOMOLE` input for O_2 for various CSFs (configuration state function) is given in Section 24.6. Further examples are collected below.

The ROHF ansatz for the energy expectation value has a term for interactions of closed-shells with closed-shells (indices k, l), a term for purely open-shell interactions (indices m, n) and a coupling term (k, m):

$$\begin{aligned} E =& 2\sum_k h_{kk} + \sum_{k,l}(2J_{kl} - K_{kl}) \\ &+ f[2\sum_m h_{mm} + f\sum_{m,n}(2aJ_{mn} - bK_{mn}) + 2\sum_{k,m}(2J_{km} - K_{km})] \end{aligned}$$

where f is the (fractional) occupation number of the open-shell part ($0 < f < 1$), and a and b are the Roothaan parameters, numerical constants which depend on the particular configuration of interest.

6.3.2 One Open Shell

Given are term symbols (up to indices depending on actual case and group) and a and b coefficients. n is the number of electrons in an *irrep* with degeneracy n_{ir}. Note that not all cases are Roothaan cases.

All *single electron* cases are described by:

$$a = b = 0$$

Table 6.1: Roothaan-coefficients a and b for cases with degenerate orbitals.

n_{ir}=2: e (div. groups), π, δ ($C_{\infty v}$, $D_{\infty h}$)						
n	f	e^n	π^n	δ^n	a	b
2	1/2	^{3}A	$^3\Sigma$	$^3\Sigma$	1	2
		^{1}E*	$^1\Delta$	$^1\Gamma$	1/2	0
		^{1}A	$^1\Sigma$	$^1\Sigma$	0	−2
3	3/4	^{2}E	$^2\Pi$	$^2\Delta$	8/9	8/9
1 n_{ir}=3: p (O(3)), t (T, O, I)†						
n	f	p^n			a	b
2	1/3	^{3}P			3/4	3/2
		^{1}D**			9/20	−3/10
		^{1}S			0	−3
3	1/2	^{4}S			1	2
		^{2}D**			4/5	4/5
		^{2}P			2/3	0
4	2/3	^{3}P			15/16	9/8
		^{1}D**			69/80	27/40
		^{1}S			3/4	0
5	5/6	^{2}P			24/25	24/25
only irrep g(I) (mainly high spin available)						
n	f	g^n			a	b
1	1/8	^{2}G			0	0
2	1/4	††			2/3	4/3
		^{1}A			0	−4
3	3/8	^{4}G			8/9	16/9
4	1/2	^{5}A			1	2
5	5/8	^{4}G			24/25	32/25
6	3/4	††			26/27	28/27
		^{1}A			8/9	4/9
7	7/8	^{2}G			48/49	48/49

continues on next page

Table 6.1: Roothaan-coefficients a and b for cases with degenerate orbitals (continued).

d(O3), h(I) (mainly high-spin cases work)				
n	f	d^n	a	b
1	1/10	^{2}D	0	0
2	1/5	^{3}F+^{3}P††	5/8	5/4
		^{1}S	0	-5
3	3/10	^{4}F+^{4}P††	5/6	5/3
4	2/5	^{5}D, ^{5}H	15/16	15/8
5	1/2	^{6}S, ^{6}A	1	2
6	3/5	^{5}D, ^{5}H	35/36	25/18
7	7/10	^{4}F+^{4}P††	95/98	55/49
8	4/5	^{3}F+^{3}P††	125/128	65/64
		^{1}S	15/16	5/8
9	9/10	^{2}D, ^{2}H	80/81	80/81

* except cases (e.g. D_{2d} or D_{4h}) where e^2 gives only one-dimensional irreps, which are not Roothaan cases.

† only p^n given, the state for groups T_d etc. follows from S $\rightarrow$ A (T,O,I) P $\rightarrow$ T (T,O,I) D $\rightarrow$ H (I), E+T (T,O)

** This is not a CSF in T or O, (a, b) describes average of states resulting from E+T

†† (a, b) describes weighted average of high spin states, not a CSF.

Example

The $4d^9\,5s^2$ ^{2}D state of Ag, in symmetry I

```
$closed shells
  a       1-5                                    ( 2 )
  t1      1-3                                    ( 2 )
  h       1                                      ( 2 )
$open shells type=1
  h       2                                      ( 9/5 )
$roothaan         1
  a = 80/81      b = 80/81
```

6.3.3 More Than One Open Shell

A Half-filled shell and all spins parallel

All open shells are collected in a single open shell and

$$a = 1 \qquad b = 2$$

Example: The $4d^5\,5s^1\ {}^7$S state of Mo, treated in symmetry I

```
$roothaan         1
  a = 1      b = 2
$closed shells
  a      1-4                                    ( 2 )
  t1     1-3                                    ( 2 )
  h      1                                      ( 2 )
$open shells type=1
  a      5                                      ( 1 )
  h      2                                      ( 1 )
```

Two-electron singlet coupling

The two MOs **must** have different symmetries (not required for triplet coupling, see example 6.3.3). We have now two open shells and must specify three sets of (a, b), i.e. one for each pair of shells, following the keyword `$rohf`.

Example: CH_2 in the $^1B_2$ state from $(3a_1)^1\ (1b_2)^1$, molecule in (x,z) plane.

```
$closed shells
  a1      1-2                                    ( 2 )
  b1      1                                      ( 2 )
$open shells type=1
  a1      3                                      ( 1 )
  b2      1                                      ( 1 )
$roothaan         1
$rohf
  3a1-3a1  a = 0      b = 0
  1b2-1b2  a = 0      b = 0
  3a1-1b2  a = 1      b = -2
```

Two open shells

This becomes tricky in general and we give only the most important case:

shell 1 is a Roothaan case, see 6.3.2

shell 2 is one electron in an a (s) MO ($n_{ir} = 1$)
with parallel spin coupling of shells.

This covers e.g. the $p^5\, s^1\ {}^3$P states, or the $d^4 s^1\ {}^6$D states of atoms. The coupling information is given following the keyword `$rohf`. The (a, b) within a shell are taken from above (6.3.2), the cross term (shell 1)–(shell 2) is in this case:

$$a = 1 \quad \text{always}$$

$$b = 2 \quad \text{if } n \le n_{ir} \qquad b = \frac{(2n_{ir})}{n} \quad \text{if } n > n_{ir}$$

where n_{ir} and n refer to shell 1.

Example 1: The $4d^4\, 5s^1\ {}^6$D state of Nb, in symmetry I

```
$closed shells
 a       1-4                                    ( 2 )
 t1      1-3                                    ( 2 )
 h       1                                      ( 2 )
$open shells type=1
 a       5                                      ( 1 )
 h       2                                      ( 4/5 )
$roothaan         1
$rohf
5a-5a      a = 0      b = 0
5a-2h      a = 1      b = 2
2h-2h      a = 15/16  b = 15/8
```

Example 2: The $4d^5\, 5s^1\ {}^7$S state of Mo, symmetry I (see Section 6.3.3) can also be done as follows.

```
$roothaan         1
$rohf
5a-5a      a = 0      b = 0
5a-2h      a = 1      b = 2
2h-2h      a = 1      b = 2
```

```
$closed shells
 a      1-4                                    ( 2 )
 t1     1-3                                    ( 2 )
 h      1                                      ( 2 )
$open shells type=1
 a      5                                      ( 1 )
 h      2                                      ( 1 )
```

The shells $5s$ and $4d$ have now been made *inequivalent*. Result is identical to 6.3.3 which is also more efficient.

Example 3: The $4d^9\,5s^1\ {}^3$D state of Ni, symmetry I

```
$closed shells
 a       1-3                                    ( 2 )
 t1      1-2                                    ( 2 )
$open shells type=1
 a       4                                      ( 1 )
 h       1                                      ( 9/5 )
$roothaan         1
$rohf
 4a-4a a = 0     b = 0
 1h-1h a = 80/81 b = 80/81
 4a-1h a =1      b = 10/9
```

(see basis set catalogue, basis `SV.3D` requires this input and gives the energy you must get)

6.3.4 Miscellaneous

Valence states

Valence states are defined as the weighted average of *all* CSFs arising from an electronic configuration (occupation): $(\mathrm{MO})^n$. This is identical to the average energy of all Slater determinants.

$$a = b = \frac{2n_{ir}(n-1)}{(2n_{ir}-1)n}$$

This covers, e.g. the cases $n = 1$ and $n = 2n_{ir} - 1$: p^1, p^5, d^1, d^9, etc, since there is only a single CSF which is identical to the average of configurations.

Totally symmetric singlets for 2 or ($2n_{ir}$-2) electrons

$$
\begin{aligned}
n &= 2 & a &= 0 \qquad b = -n_{ir} \\
n &= (2n_{ir} - 2) & a &= \frac{n_{ir}(n_{ir}-2)}{(n_{ir}-1)^2} \\
& & b &= \frac{n_{ir}(n_{ir}-3)}{(n_{ir}-1)^2}
\end{aligned}
$$

This covers the ^{1}S states of p^2, p^4, d^2, d^8, etc.

Average of high-spin states: n electrons in MO with degenerate n_{ir}.

$$
\begin{aligned}
a &= \frac{n_{ir}\big(4k(k+l-1)+l(l-1)\big)}{(n_{ir}-1)n^2} \\
b &= \frac{2n_{ir}\big(2k(k+l-1)+l(l-1)\big)}{(n_{ir}-1)n^2}
\end{aligned}
$$

where: $k = \max(0, n - n_{ir})$, $l = n - 2k = 2S$ (spin)

This covers most of the cases given above. A CSF results only if $n = \{1, (n_{ir}-1)$, n_{ir}, $(n_{ir}+1)$, $(2n_{ir}-1)\}$ since there is a single high-spin CSF in these cases.

The last equations for a and b can be rewritten in many ways, the probably most concise form is

$$
\begin{aligned}
a &= \frac{n(n-2)+2S}{(n-2f)n} \\
b &= \frac{n(n-2)+(2S)^2}{(n-2f)n}\,.
\end{aligned}
$$

This applies to shells with one electron, one hole, the high-spin couplings of half-filled shells and those with one electron more ore less. For d^2, d^3, d^7, and d^8 it represents the (weighted) average of high-spin cases: ^{3}F + ^{3}P for d^2,d^8, ^{4}F + ^{4}P for d^3, d^7.

6.4 Relativistic effects

TURBOMOLE provides two different possibilities for the treatment of relativistic effects: Via effective core potentials (ECPs) or via all-electron approaches (X2C, DKH, BSS). Both techniques can be employed in a one-component (scalar-relativistic) or two-component (including spin-orbit coupling) framework. The latter is only available in the modules `ridft`, `rdgrad`, `escf`, `mpshift`, and `ricc2`.

6.4.1 One- and two-component relativistic methods

Incorporation of scalar-relativistic effects leads to additional contributions to the one-electron integrals (either from ECP or all-electron approach). The program structure is the same as in non-relativistic theory (all quantities are real). Two-component treatments allow for self-consistent calculations including spin-orbit interactions. These may be particularly important for compounds containing heavy elements (additionally to scalar-relativistic effects). Two-component treatments require the use of complex two-component orbitals (spinors)

$$\psi_i(\mathbf{x}) = \begin{pmatrix} \psi_i^{\alpha}(\mathbf{r}) \\ \psi_i^{\beta}(\mathbf{r}) \end{pmatrix}$$

instead of real (non-complex) one-component orbitals in non-relativistic or scalar-relativistic treatments. The Hartree–Fock and Kohn–Sham equations are now spinor equations with a complex Fock operator

$$\begin{pmatrix} \hat{F}^{\alpha\alpha} & \hat{F}^{\alpha\beta} \\ \hat{F}^{\beta\alpha} & \hat{F}^{\beta\beta} \end{pmatrix} \begin{pmatrix} \psi_i^{\alpha}(\mathbf{r}) \\ \psi_i^{\beta}(\mathbf{r}) \end{pmatrix} = \epsilon_i \begin{pmatrix} \psi_i^{\alpha}(\mathbf{r}) \\ \psi_i^{\beta}(\mathbf{r}) \end{pmatrix}.$$

The wavefunction is no longer an eigenfunction of the spin operator, the spin vector is no longer an observable.

In case of DFT for open-shell systems, rotational invariance of the exchange-correlation energy is ensured by the non-collinear approach. In this approach, the exchange-correlation energy is a functional of the particle density and the absolute value of the spin-vector density $\vec{m}(\mathbf{r})$ ($\vec{\boldsymbol{\sigma}}$ are the Pauli matrices)

$$\vec{m}(\mathbf{r}) = \sum_i \psi_i^{\dagger}(\mathbf{x}) \vec{\boldsymbol{\sigma}} \psi_i(\mathbf{x}).$$

This quantity replaces the spin-density (difference between density of alpha and beta electrons) of non- or scalar-relativistic treatments.

For closed-shell species, the Kramers-restricted scheme, a generalization of the RHF-scheme of one component treatments, is applicable.

Effective core potentials

The most economic way to account for relativistic effects is via effective core potentials by choosing either the one- or the two-component ECP (and for the latter

additionally setting `$soghf` in the control file or in `define`). The theoretical background and the implementation for the two-component SCF procedure is described in Ref. [140]. For recommendations concerning specific ECPs and corresponding basis sets see below.

Relativistic all-electron approaches (X2C, DKH, BSS)

Relativistic calculations are based on the Dirac rather than on the Schrödinger Hamiltonian. Since the Dirac Hamiltonian introduces pathological negative-energy states and requires extensive one-electron basis set expansions, methods have been devised which allow one to calculate a matrix representation of that part of the Dirac Hamiltonian, which describes electronic states only. For this, a unitary transformation is employed to block-diagonalize the Dirac Hamiltonian and thus to decouple the negative-energy states from the electronic states. For reasons of efficiency, this transformation is carried out only for the one-electron part of the full Hamiltonian (as a consequence, the two-electron interaction will then be slightly affected by a so-called picture-change effect). The resulting quantum chemical approach, "exact two-component" (X2C), was developed by several groups starting with formal work in the mid-1980s. X2C is related to the step-wise Douglas–Kroll–Hess (DKH) approach, which achieves decoupling in sequential manner. For the latter, the number of transformation steps is called the order of DKH. Infinite-order DKH yields identical results compared to X2C, but – in contrast to the latter – not feasible. Eighth order DKH usually yields results similar to X2C at similar cost. X2C is also related to the Barysz–Sadlej–Snijders (BSS) method, that first applies the free-particle Foldy-Wouthuysen transformation (which is the first mandatory step in DKH), and then constructs the one-step exact decoupling transformation of X2C. These three approaches have been reviewed and directly compared in terms of formalism and results, respectively, in Ref. [141] (see also this reference for a complete bibliography on exact-decoupling methods).

Essentially, X2C methods change the one-electron Hamiltonian in a basis-set representation. The Schrödinger one-electron Hamiltonian (including the external potential of the atomic nuclei) is replaced by the transformed (upper-left block of the) Dirac Hamiltonian. Since the transformation is carried out in the fully decontracted primitive basis, all matrix operations needed for the generation of the relativistic one-electron Hamiltonian can be cumbersome and even prohibitive if the molecule is large. In order to solve this unfavorable scaling problem, a rigorous local approach, called DLU, has been devised [142] and is strongly recommended.

X2C, DKH, and BSS exist in full two-component (spin-(same-)orbit coupling including) and in a one-component scalar-relativistic form. Both have been implemented into the TURBOMOLE package and all details on the efficient implementation have been described in Ref. [143]. Additionally implemented modifications like a finite nucleus model based on a Gaussian charge distribution [144] and a screened nuclear spin-orbit (SNSO) approach [145–147] are documented in Ref. [148], together with the implementation of analytical one- and two-component X2C gradients, also in their local variant.

Calculation of analytical energy gradients is available within the (local) X2C approach. The implementation is described in Ref. [148]. The finite nucleus model based on a Gaussian charge distribution can be used for energy and gradient calculations. To model the effect of spin-orbit coupling on the two-electron interaction a (modified) screened-nuclear-spin-orbit approximation can be applied to the spin-dependent one-electron integrals. The scalar-relativistic approach can be used with the modules `grad`, `rdgrad`, `egrad`, `ricc2`, `mpgrad`, and `rirpa`. The two-component Hamiltonian is only available in `rdgrad`.

In relativistic all-electron calculations, additional contributions from point charges or COSMO (see chapter 19.2) are not part of the relativistic decoupling. These contributions are evaluated after the X2C, DKH, or BSS step. Thus, they are calculated without picture-change transformation.

6.4.2 How to use

Scalar-relativistic calculations with ECPs

In case of scalar-relativistic calculations with ECPs, relativity is taken into account simply by the choice of a relativistic ECP together with a suited basis set. A reasonable choice are the Dirac–Hartree–Fock effective core potentials by the Stuttgart-Köln group labeled dhf-ecp in `TURBOMOLE`, together with optimized basis sets termed dhf-XVP (X = S, TZ, QZ) [149]. These ECPs and bases are available for Rb to Xe, except for f-elements. For H to Kr dhf and def2 bases are identical non-relativistic all-electron basis sets. For lanthanides and actinides at present no dhf ECPs are available. The usage of Wood-Boring type ECPs [150], for lanthanides together with def2-bases [151], for actinides together with the original Stuttgart–Köln bases [152, 153] (labeled def in `TURBOMOLE`) is recommended here. Note that for def2-bases the underlying ECPs are of different type (due to history). or p-elements, def2-bases are optimized for Dirac–Hartree–Fock ECPs, for the other elements for Wood–Boring ECPs.

Two-component calculations (general)

The keyword `$soghf` enforces the two-component calculations. Keywords for specification of the method of calculation are the same as for the one-component case. Additionally, for closed-shell species a Kramers invariant density functional formalism can be switched on with the keyword `$kramers`. The DIIS scheme for complex Fock operators (GDIIS) can be activated by `$gdiis`. For improvements on the SCF convergence behavior and gradients, please see Ref. [154]. It is recommended to use exact exchange for double- and triple-zeta bases instead of semi-numerical [128] or RI-K approximations. These keywords can be inserted into the control file manually or added in the `scf` section of `define`. By default, the RI-J approximation is employed. The analytical Coulomb integrals can be used with the option `$coulex` in `ridft`, `rdgrad`, and `mpshift`.

As start wavefunctions Hückel or SCF wavefunctions may be used. We recommend to use converged one-component molecular orbitals or a two-component superposition of atomic densities (`$atmgs2c` combined with hcore -atoms) as starting point. The first option means that first a scalar-relativistic calculation is carried out without the keyword `$soghf`. Then, the keywords for a two-component calculation can be added and the two-component calculations can be started. This allows for a much smoother convergence. Here, the one-component orbitals are transformed to the two-component picture in the first iteration. By default, this is done with the Pauli matrix of the z component, meaning that the wavefunction is an eigenfunction of the z spin operator after the first iteration. Alternatively, you may specify the desired spin alignment with `$sxeig`, `$syeig`, or `$szeig`. The second option means that a non-collinear superposition of atomic densities with the desired spin alignment is formed. Note that this formalism uses Hartree–Fock and adds a spin–orbit energy directly in the first iteration.

For open-shell molecules it is often helpful to increase the value for `$scforbitalshift closedshell`; a value of ca. 1.0 may serve as a rough recommendation. Likewise, the default value for the automatic orbitalshift can be adapted to improve the convergence. Additionally, canonical orthogonalization may be helpful to improve the convergence behavior, see Sec. 6.6 for details.

Generally, spin–orbit coupling induces a (paramagnetic) current density. Functionals such as meta-GGAs and local hybrids, which depend on the kinetic energy density, require the inclusion of this current density from a formal point of view. That is, functionals need to be constructed in the framework of current density functional theory (CDFT) [42]. This is applied by the keyword `$curswitchengage` in `ridft`, `rdgrad`, `mpshift`, and `escf`. Currently, this formalism is restricted to a common gauge origin in `mpshift`.

It is possible to turn off spin–orbit coupling with the keyword `$noso`. Then, no one-electron spin–orbit integrals are employed. Complex algebra is still employed. Likewise, the one-electron spin–orbit can be scaled with `$soscal` followed by the desired scaling factor. Default is one. Note that not only the respective integrals for the SCF procedure but also those for properties are rescaled. This option is available for 2c ECP and DKH/BSS/X2C calculations.

Two-component calculations with ECPs

For spin-orbit treatments, the two-component variants of ECPs (suffix -2c) are required, the use of extended basis sets accounting for the spatial splitting of inner p-shells (also suffix -2c) is recommended. ECPs dhf-ecp-2c and bases dhf-XVP-2c (X = S, TZ, QZ) [149] are available for Rb to Kr (f elements excepted); for references concerning ECPs see `http://www.tc.uni-koeln.de/PP/clickpse.en.html`. The two-component formalism may be most easily prepared and applied in the following way:

- Run `define`: choose C_1-symmetry; select ECPs and basis sets with suffixes -2c for the respective elements. RI-J and RI-JK auxiliary basis sets are the

same for dhf and def2 bases. Moreover, they are of sufficient flexibility for two-component treatments and provided automatically upon request. Switch on `soghf` and further desired options in the `scf` menu.

- Start the two-component calculation with `ridft`.
- At the end of the SCF procedure real and imaginary parts of spinors are written to files spinor.r and spinor.i, eigenvalues and spinor occupations are collected in the file EIGS, the total energy is added to data group `$energy`. The data groups `$closed shells` (`$alpha shells` and `$beta shells` for open shell cases) are no longer significant, but nevertheless kept in the control file; additionally the spinor occupations are deposited in data group `$spinor`.

One- and two-component all-electron calculations

All-electron calculations are prepared similarly to ECP calculations, however, relativistic all-electron basis sets are necessary. It is recommended to use the x2c-type basis sets and RI-J auxiliary basis sets x2c-XVPall for one-component and x2c-XVPall-2c for two-component treatments [155–157]. Presently, they are available for X = S, TZ, QZ for H to Rn. The keywords `$rx2c`, `$rbss` and `$rdkh` n (where n stands for the order of DKH) are used to activate the X2C, BSS or DKH Hamiltonian. n defaults to 4. It is not recommended to go beyond, but to use X2C instead. For details on the arbitrary-order DKH Hamiltonians, see Ref. [158] for details on the infinite-order DKH theory, [159] for the implementation, and [160] for a conceptual review of DKH theory. The local approach (DLU) can be optionally activated by `$rlocal` for all one- and two-component all-electron Hamiltonians. A picture-change correction for several expectation values in the proper section is available for relativistic all-electron Hamiltonians (`$rdkh`, `$rbss`, `$rx2c`, also in their local variant `$rlocal`) and enforced by `$pcc`.

For symmetric molecules, point-group symmetry is not exploited by default, but can be used in the one-component case by setting `$rsym`. Without setting `$rsym`, symmetry is only exploited in the rest of the program. Note that `$rsym` cannot be combined with `$rlocal`. The exploitation of symmetry is also supported in X2C gradient calculations [41]. Therein, nuclear symmetry is exploited without setting `$rsym`. The finite nucleus model based on a Gaussian charge distribution is selected by `$finnuc`.

Two-component calculations are invoked by the keyword `$soghf` as outlined above. The SNSO approach may be used to estimate the effect of spin-orbit coupling on the two-electron interaction by using the keyword `$snso`. By default, the modified parameters [146,147] (mSNSO) are taken, however, the default parameters [145] can be applied by setting `$snsopara` to 0. The latter set of parameters was derived for lower-order DKH whereas the modified parameters were optimized for X2C and greatly improve the spinor energies for virtual states. All options can be specified in the scf section of `define`. The SNSO, GDIIS, and scforbitalshift parameters are set in the soghf part of the scf section. Overall, we recommend the mSNSO-DLU-X2C Hamiltonian with the finite nucleus model is recommended.

It is further strongly recommended to use grids with an increased number of radial points [156] in DFT calculations. These grids are selected by appending "a" to the gridsize, i.e. gridsize 4a.

In NMR shielding calculations, the use of tailored basis sets, i.e. x2c-SVPall-s, x2c-TZVPall-s [156] and x2c-QZVPall-s [157] etc., is recommended.

6.5 Finite magnetic fields

General framework of finite-magnetic-field calculations

Calculations of magnetic properties in finite magnetic fields can be carried out in the 2c framework (see section 6.4). The Hamiltonian in a magnetic field ($\mathcal{B}$) can be written as:

$$\hat{H} = \hat{H}_0 + \frac{1}{2}\sum_i \mathcal{B}\cdot\mathbf{l}_{iO} + \sum_i \mathcal{B}\cdot\mathbf{s}_i + \frac{1}{8}\sum_i \left(\mathcal{B}^2\mathbf{r}_{iO}^2 - (\mathcal{B}\cdot\mathbf{r}_{iO})^2\right)$$

Here, $\hat{H}_0$ is the zero-field Hamiltonian. The terms dependent on the angular momentum operator $\mathbf{l}_{iO}$ and spin operator $\mathbf{s}_i$ are called orbital-Zeeman term and spin-Zeeman term, respectively. Both are referred to as paramagnetic terms since they are linearly dependent on the magnetic field. The terms which are quadratically dependent on the magnetic field are called diamagnetic terms. In order to avoid a dependence on the gauge origin O of the system, London orbitals are used.
Using this framework, calculations in finite magnetic fields can be performed at various levels of theory. The starting point of any such calculation is always a 2c calculation using the `ridft` module which has to include the `$soghf` and `$magnetic field` keywords. The x, y and z coordinates of the magnetic field as well as its absolute value have to be specified. An exemplary input into the control file might look like this:

```
$soghf
$magnetic field
  1 1 2 0.1
```

This input assumes atomic units and thus this would correspond to a magnetic field of $|\mathcal{B}| = 0.1$ being applied in the $(112)^T$ direction. Within the `$magnetic field` keyword, a number of additional useful things may be specified. Writing 'tesla' will assume that the magnetic field is not given in atomic units but in units of tesla instead. Writing 'spin xxx' will scale the Spin-Zeeman term by a constant of xxx. This might help if one wants to converge towards different spin states. Adding a larger, positive number will make it easier to converge to spin states with larger multiplicity. In the last iteration, the Spin-Zeeman term is not scaled anymore and thus the correctly non-scaled result is displayed at the end of any calculation.

Available quantum-chemical methods

Calculations in finite magnetic fields are possible employing a number of different quantum-chemical methods. The RI approximation may be used throughout. If an exact calculation of the Coulomb-part is desired, the `$coulex` keyword needs to be added.

- Generalized Hartree-Fock (GHF) and Density functional theory (DFT) using the `ridft` module. Molecular gradients of a converged GHF or DFT calculation can be obtained using the `rdgrad` module. Consequently, it is possible to optimize the structure of a molecule on the GHF level using the `jobex` script if the '-ri' option is specified. It should be noted that most symmetries are broken using this ansatz and the resulting wave function needn't be an eigenfunction to either $\hat{S}^2$, $\hat{S}_z$ or the time-reversal operator $\hat{T} = -\mathrm{i}\sigma_y$. Therefore, the results are often spin-contaminated (check $< \hat{S}^2 >$!) and not Kramers-symmetric.
- Post-Hartree-Fock methods MP2 and CC2 for ground states using the Ricc2 module. As always, a converged GHF solution needs to be present.
- Calculation of excited states and related properties using TDHF, TDDFT or GW/BSE employing the `escf` module. Excitation energies (but not properties!) can be calculated on the CC2 and ADC(2) level using the Ricc2 module.

6.6 Canonical orthogonalization

When performing an SCF calculation like Hartree-Fock or Kohn-Sham DFT, it is necessary to expand the wave function using basis sets. In general, this leads to a generalized, hermitian eigenvalue problem:

$$\mathbf{H}\mathbf{C}_i = E_i\mathbf{S}\mathbf{C}_i$$

In order to transform this, it is necessary to orthogonalize the basis in such a way that $\mathbf{S}$ becomes the unit matrix and we generate a proper eigenvalue problem:

$$\mathbf{S}^{-1/2}\mathbf{H}\mathbf{S}^{-1/2}\mathbf{C}_i = E_i\mathbf{C}_i$$

This orthogonalization of the wave function is usually done at the beginning of any SCF calculation and it is usually done every 5-th iteration or so in order to avoid numerical instabilities. Since this includes the calculation of $\mathbf{S}^{-1/2}$, it is common to perform such a calculation using a Cholesky decomposition $\mathbf{S} = \mathbf{S}^{1/2\,\dagger}\mathbf{S}^{1/2}$ and subsequently inverting $\mathbf{S}^{1/2}$ since it is an upper triangular matrix. However, if a basis set with (near) linear-dependencies is used, this procedure will result in a lot of numerical inaccuracies which can lead to a very poor convergence behaviour with some systems not being able to converge at all. This is usually the case for larger systems and larger basis sets, i.e. a typical Turbomole calculation.

It is possible to use the canonical orthogonalization instead of a Cholesky decomposition, that is, diagonalizing $\mathbf{S}$ by calculating its eigenvalues and eigenvectors by making use of the fact that $\mathbf{U}^\dagger\mathbf{S}\mathbf{U} = \mathbf{D}$ where $\mathbf{D}$ is a diagonal matrix containing the eigenvalues of $\mathbf{S}$. The inverse square root can the be calculated using $\mathbf{S}^{-1/2} = \mathbf{U}\mathbf{D}^{-1/2}\mathbf{U}^\dagger$. If, however, the basis set has near-linear dependencies, then $\mathbf{S}$ (which is a positive-definite matrix) will have eigenvalues close to 0 and consequently

$\mathbf{D}^{-1/2}$ will contain very large numbers. This can be avoided if all eigenvalues below a certain threshold (corresponding to near-linear dependencies) are filtered. This can be done using the keyword `$canonorth`:

```
$canonorth 1.0d-7
```

Here, all eigenvalues of $\mathbf{S}$ below the treshold of $1.0d-7$ are filtered. In principle, any number can be entered, but if it is chosen to be too low the 'problematic' eigenvalues are not filtered and if it is chosen to be too high, the result will be way above the desired variational minimum. Usually, a value between $1.0d-5$ and $1.0d-8$ is a good choice. If one encounters convergence issues for large systems, especially if larger basis sets are employed, it is usually a good idea to check if the canonical orthogonalization solves this issue.

6.7 Dispersion Correction for DFT Calculations

Grimme DFT+Dispersion, DFT-D3, DFT-D4

Based on an idea that has earlier been proposed for Hartree-Fock calculations [161, 162], a general empirical dispersion correction has been proposed by Stefan Grimme for density functional calculations [163]. A modified version of the approach with extension to more elements and more functionals has been published in ref. [164]. A more recent implementation [165] is less empirical, i.e. the most important parameters are computed by first principles, and it provides a consistent description across the whole periodic system. This version, named DFT-D3 with Becke-Johnson damping, has been the default for dispersion correction ever since.

In 2017 a new model, termed D4, was published [166, 167] and released 2018. For details see below.

- The first version (DFT-D1) can be invoked by the keyword `$olddisp` in the `control` file.
- The second version (DFT-D2) is used if the keyword `$disp` is found.
- For the usage of DFT-D3 just add keyword `$disp3` to the `control` file. For DFT-D3 with Becke-Johnson damping, please add `$disp3 -bj`.
- The latest version, DFT-D4, can be invoked by the keyword `$disp4`.

Only one of the four keywords is expected to be present.

DFT-D3

If DFT-D3 is used, the total energy is given by

$$E_{DFT-D3} = E_{KS-DFT} - E_{disp} \tag{6.11}$$

where E_{KS-DFT} is the usual self-consistent Kohn-Sham energy as obtained from the chosen functional and E_{disp} is a dispersion correction given by the sum of two- and three-body energies

$$E_{disp} = E^{(2)} + E^{(3)}, \tag{6.12}$$

with the dominating two-body term

$$E^{(2)} = \sum_{AB} \sum_{n=6,8,10,\ldots} s_n \frac{C_n^{AB}}{r_{AB}^n} f_{d,n}(r_{AB}). \tag{6.13}$$

The first sum runs over all atom pair, C_n^{AB} denotes the nth-order dispersion coefficient for atom pair AB, r_{AB} is their interatomic distance, and $f_{d,n}$ is a damping function.

Becke-Johnson (BJ) damping can be invoked by adding the option bj or -bj to the `$disp3` keyword: `$disp bj` If you use this damping option please also cite [168].

The three-body term can be switched on by adding `abc` to the `$disp3` input line, i.e. to use it in combination with Becke-Johnson damping just add `$disp3 bj abc`

It is also possible not to use the functional name given in the control file but to tell the DFT-D3 routines to use the parameters which have been fitted to a specific functional. Just as in the original DFT-D3 routines, this can be selected by adding the `func` option, for example `$disp3 bj func pbe0`. It is recommended to use this option as the last one in the `$disp3` input line.

Please have look at DFT-D3 homepage, Grimme group Bonn for more detailed information.

DFT-D4

DFT-D4 computes molecular dipole-dipole and dipole-quadrupole dispersion coefficients based on dynamic dipole polarizabilities calculated with TD-PBE38/daug-def2-QZVP used as atomic reference polarizabilities for elements up to radon (Z=86). In DFT-D4, all atomic reference polarizabilities are scaled according to Mulliken partial charges which are obtained by semi-empirical quantum mechanical tight binding, within the 2018 developed GFN2-xTB framework [69]. Alternatively a fallback option is provided in case GFN2-xTB fails to converge the electronic structure. Either classical, non-interative EN and CN dependent Gasteiger charges (option 'gasteiger') or (externally provided) Hirshfeld-charges optained at least at PBE0/def2-TZVP level of theory can be used (option 'hirshfeld').

For the scaling of the atomic reference polarizabilities, a special charge-function was designed containing one global parameter and the chemical hardnesses as elementspecific parameter to get a smooth scaling behavior. The charge-scaled dipole polarizabilities are Gaussian interpolated according to an empirically generated fractional coordination number to obtain the fully weighted dynamic polarizabilities $\alpha(i\omega)$ for all atoms which are then numerically integrated via the Casimir-Polder scheme to obtain charge- and geometry-dependent dipole-dipole dispersion coefficients. The two-body energy expression has the usual sum over pair interactions form for dipole-dipole and dipole-quadrupole interactions.

Based on this expression, a self-consistent dispersion potential has been developed and implemented into the GFN2-xTB Hamiltonian to circumvent a costly coupled-perturbed SCF procedure when calculating analytical gradients. Furthermore, dynamic polarizabilities $\alpha(i\omega)$ are used within an RPA-like expression to capture many-body interactions beyond the two-body terms.

The DFT-D4 energy expression differs for single point energies and gradient calculations, for reasons of computational efficiency. The gradient of the RPA-like dispersion energy scales as $O(N^4)$ and cannot be screened, therefore the gradient is approximated by the ATM three-body dispersion expression as used in DFT-D3, which is computational less demanding and yields approximately similar gradients.

Parameters for the rational damping scheme introduced by Becke and Johnson are included for a large number of density functionals (including all standard functionals implemented in Turbomole) in the DFT-D4 code. The rational damping is the only scheme available in DFT-D4, also many-body dispersion terms are included by default.

The DFT-D4 keyword `$disp4` can have, similar to `$disp3`, a couple of options. To see the list of options, just add `-help` to the keyword and inspect the output file.

As mentioned above, the gradient of the many body dispersion energy is done by using the Axilrod-Teller-Muto (ATM) three-body term, which is only an approximation compared to the term used in the energy evaluation. Hence, for larger molecules, the DFT-D4 energy and the DFT-D4 gradient do not match exactly. Very tight convergence criteria for both energy and gradient norm can not be expected in all cases.

Density-based dispersion corrections of non-local vdW-DF type

A non-local, electron density dependent dispersion correction which is based on Vydrov and Van Voorhis' VV10 [169] has been implemented by the Grimme group [170] and is available for `ridft` and `rdgrad`. This correction can either be applied in a post-SCF and non-self-consistent way for energy calculations or self-consistently which is required to compute the gradients.

To switch on DFT-NL in a non-self-consistent way, just add

```
$donl
```

to the control file. For a self-consistent treatment of the dispersion correction add

```
$doscnl
```

instead. Note that dispersion corrections of DFT-DN and NL–DFT type must not be combined. The grid size for the non-local integration is set automatically by adapting the grid for the quadrature of the functional evaluation.

Currently only C_1 symmetry and serial jobs are possible. DFT-NL is an interesting scientific alternative to DFT-D3, but we recommend to use DFT-D3 for applications instead.

6.8 Energy Decomposition Analysis (EDA)

The interaction energy between molecules can be calculated with the supermolecular approach: one performs calculations for the supersystem and for the subsystems with size-consistent methods and derive the interaction energy ΔE by taking the energy difference. The energy decomposition analysis (EDA) allow a partitioning of the Hartree-Fock (HF) or DFT interaction energy in physically meaningful contributions: the classical electrostatic interaction ΔE_{ele}, the exchange-repulsion ΔE_{exrep}, the orbital relaxation energy ΔE_{orb} and additionally for DFT the correlation interaction ΔE_{cor}:

$$\Delta E_{\text{HF}} = \Delta E_{\text{ele}} + \Delta E_{\text{exrep}} + \Delta E_{\text{orb}} \quad , \tag{6.14}$$

$$\Delta E_{\text{DFT}} = \Delta E_{\text{ele}} + \Delta E_{\text{exrep}} + \Delta E_{\text{orb}} + \Delta E_{\text{cor}}. \tag{6.15}$$

Further details and derivations of the different energy contributions can be found in [171].

6.8.1 How to perform

The EDA scheme is implemented in the module `ridft` and can be done with RI-Hartree-Fock and with all local, gradient corrected, hybrid and meta density functionals (please note that the functionals included in the XCFun library are not supported!).

- Calculation of the subsystems:
  In HF and hybrid DFT calculations please insert `$scfdenapproxl 0` in the `control` file, at least in a second run. The EDA scheme needs the exchange and Coulomb energies of every system separately. After the subsystem calculation you will find under `$subenergy` the different energy contribution to the total energy of the system: the one electron energy, Coulomb- and exchange energy, correlation energy in case of DFT calculations, nuclear repulsion energy and optionally the dispersion energy.

- Preparation of the supersystem `control` file:
  First run `define` for the supersystem and take for consistency the same basis set and the same method (i.e. the same functional and the same grid). Please use in case of DFT calculations not the multiple grids `m3` to `m5`, because this would lead to erroneous orbital relaxation energies. If the subsystems are open-shell species the occupation in the EHT submenue of `define` of the supersystem must be chosen open-shell, too. For open-shell systems the Fermi-smearing is recommended. The sequence of the supersystem coordinates must have the same sequence as the subsystem coordinates. In the case of HF and hybrid DFT calculation use again `$scfdenapproxl 0`.
  Then please insert in the `control` file :

```
$subsystems
  molecule#1 file=sub1/control
  molecule#2 file=sub2/control
```

If you use the supermolecular basis set for the calculation of the monomers please insert after `$subsystems` the option `copo`:

```
$subsystems copo
   molecule#1 file=sub1/control
   molecule#2 file=sub2/control
```

It is possible to generate orthogonal product wave functions when you use `opro` instead of `copo`. But with this choice it is not possible to calculate the different energy contributions of the interaction energy.
You can choose at most ten subsystems.

- Generation of the product molecular wave functions:
  The module `promowa` generates RHF and UHF product molecular wave functions. The new (product) start vectors can be found in the files `mos` for closed-shell systems or in `alpha` and `beta` for open-shell systems. Please note that the molecular orbitals of the different subsystems are not orthogonal to each other.

- Energy decomposition analysis:
  After the supersystem `ridft` calculation you will find the following output with the different contributions of the interaction energy in Hartree:

```
 ------------------------------------------------
| * Total Interaction energy  =      -0.0058700698  |
 ------------------------------------------------
: * Electrostatic Interaction =      -0.0134898233  :
:             Nuc---Nuc       =      18.2843363825  :
:             1-electron      =     -36.5372802833  :
:             2-electron      =      18.2394540775  :
: * Exchange-Repulsion        =       0.0112325934  :
:             Exchange Int.   =      -0.0139477002  :
:             Repulsion       =       0.0251802936  :
: * Orbital Relaxation        =      -0.0036128399  :
.................................................
```

Chapter 7

DFT Calculations for Molecular and Periodic Systems

7.1 Functionalities of RIPER

The `riper` module is an implementation of Kohn–Sham DFT with Gaussian-type orbitals (GTO) as basis functions that treats molecular and periodic systems of any dimensionality on an equal footing. Its key component is a combination of resolution of identity (RI) approximation and continuous fast multipole method (CFMM) applied for the electronic Coulomb term [172, 173]. This RI-CFMM scheme operates entirely in the direct space and partitions Coulomb interactions into far-field part evaluated using multipole expansions and near-field contribution calculated employing density fitting. Evaluation of the exchange-correlation term is performed using an octree-based adaptive numerical integration scheme [174]. `riper` offers computational efficiency and favorable scaling behavior approaching $O(N)$ for the formation of Kohn–Sham matrix [172] and gradient calculation [173]. In addition, for calculations on very large molecular systems a low-memory modification of the RI approximation has been implemented [175]. This low-memory iterative density fitting (LMIDF) scheme is based on a combination of CFMM and a preconditioned conjugate gradient (CG) solver. Compared with the standard RI implementation, up to 15-fold reduction of the memory requirements is achieved at a cost of only slight increase in computational time.

Functionalities of `riper`:

- Kohn–Sham DFT for molecular systems and systems with 1D, 2D, and 3D periodicity
- closed– and open–shell energies and gradients, structure optimization including optimization of cell parameters
- metals and semiconductors can be treated using fractional occupation numbers scheme employing Gaussian smearing

- efficient **k** point sampling scheme for periodic systems allowing for consistent results across different definitions of unit cells
- sequential and parallel runs (OpenMP parallelization for shared-memory computers, see Sec. 3.4.2)
- all LDA, GGA and meta-GGA exchange-correlation functionals including interface to the XCFun library (see Sec. 6.2)
- DFT-D3 dispersion correction for energies and gradients (see Sec. 6.7)
- favorable scaling behavior for the formation of the Kohn–Sham matrix approaching $O(N)$
- memory-efficient calculations for very large molecular systems using the LMIDF scheme
- real time-time dependent DFT (RT-TDDFT) for molecular systems

Limitations of `riper`:

- only C_1 symmetry point group for molecules and P_1 space group for periodic systems

7.2 Theoretical Background

Detailed description of methods implemented in the `riper` module is provided in Refs [172–177] and references therein. Here, only a short summary of the underlying theory is provided.

7.2.1 Kohn-Sham DFT for Molecular and Periodic Systems

In periodic systems translational symmetry of solids leads to Bloch orbitals $\psi_{p\sigma}^{\mathbf{k}}$ and one-particle energies $\varepsilon_{p\sigma}^{\mathbf{k}}$ depending on the band index p, spin σ, and the wave vector $\mathbf{k}$ within the Brillouin zone (BZ), which is the unit cell of reciprocal space. The orbitals

$$\psi_{p\sigma}^{\mathbf{k}}(\mathbf{r}) = \frac{1}{\sqrt{N_{\mathrm{UC}}}} \sum_{\mathbf{L}} e^{i\mathbf{k}^{\mathrm{T}}\mathbf{L}} \sum_{\mu} C_{\mu p\sigma}^{\mathbf{k}} \mu_{\mathbf{L}}(\mathbf{r}) \tag{7.1}$$

are expanded in GTO basis functions $\mu(\mathbf{r} - \mathbf{R}_\mu - \mathbf{L}) \equiv \mu_{\mathbf{L}}(\mathbf{r})$ centered at atomic positions $\mathbf{R}_\mu$ in direct lattice cells $\mathbf{L}$ over all N_{UC} unit cells. This results in unrestricted Kohn-Sham equations

$$\mathbf{F}_{\sigma}^{\mathbf{k}}\mathbf{C}_{\sigma}^{\mathbf{k}} = \mathbf{S}^{\mathbf{k}}\mathbf{C}_{\sigma}^{\mathbf{k}}\varepsilon_{\sigma}^{\mathbf{k}}, \tag{7.2}$$

which may be solved separately for each $\mathbf{k}$ in the BZ. The same equations hold for the molecular case, where only $\mathbf{L} = \mathbf{k} = \mathbf{0}$ is a valid choice and N_{UC} is one. Equation (7.2) contains the reciprocal space Kohn-Sham and the overlap matrices $\mathbf{F}_{\sigma}^{\mathbf{k}}$ and $\mathbf{S}^{\mathbf{k}}$, respectively, obtained as Fourier transforms of real space matrices

$$F_{\mu\nu\sigma}^{\mathbf{k}} = \sum_{\mathbf{L}} e^{i\mathbf{k}^{\mathrm{T}}\mathbf{L}} F_{\mu\nu\sigma}^{\mathbf{L}} \qquad S_{\mu\nu}^{\mathbf{k}} = \sum_{\mathbf{L}} e^{i\mathbf{k}^{\mathrm{T}}\mathbf{L}} S_{\mu\nu}^{\mathbf{L}}. \tag{7.3}$$

The elements $F^{\mathbf{L}}_{\mu\nu\sigma}$ contain three contributions: elements $T^{\mathbf{L}}_{\mu\nu}$ of the kinetic energy matrix, elements $J^{\mathbf{L}}_{\mu\nu}$ of the Coulomb matrix, and elements $X^{\mathbf{L}}_{\mu\nu\sigma}$ of the exchange-correlation matrix,

$$F^{\mathbf{L}}_{\mu\nu\sigma} = T^{\mathbf{L}}_{\mu\nu} + J^{\mathbf{L}}_{\mu\nu} + X^{\mathbf{L}}_{\mu\nu\sigma}. \tag{7.4}$$

The total energy per unit cell E is calculated as the sum of the kinetic T, Coulomb J, and exchange-correlation E_{XC} contributions,

$$E = T + J + E_{\mathrm{XC}}. \tag{7.5}$$

7.2.2 RI-CFMM Approach

The key component of `riper` is a combination of RI approximation and CFMM applied for the electronic Coulomb term [172, 173, 176]. In the RI scheme the total crystal electron density ρ^{cryst} is approximated by an auxiliary crystal electron density $\tilde{\rho}^{\mathrm{cryst}}$

$$\rho^{\mathrm{cryst}} \approx \tilde{\rho}^{\mathrm{cryst}} = \sum_{\mathbf{L}} \tilde{\rho}_{\mathbf{L}}, \tag{7.6}$$

composed of unit cell auxiliary densities $\tilde{\rho}_{\mathbf{L}}$ with

$$\tilde{\rho}_{\mathbf{L}} = \sum_{\alpha} \mathbf{c}^{\mathrm{T}} \boldsymbol{\alpha}_{\mathbf{L}}, \tag{7.7}$$

where $\boldsymbol{\alpha}_{\mathbf{L}}$ denotes the vector of auxiliary basis functions translated by a direct lattice vector $\mathbf{L}$. The vector of expansion coefficients $\mathbf{c}$ is determined by minimizing the Coulomb repulsion D of the residual density $\delta\rho = \rho - \tilde{\rho}$

$$D = \iint \delta\rho\left(\mathbf{r}\right) \frac{1}{\mid \mathbf{r} - \mathbf{r}' \mid} \sum_{\mathbf{L}} \delta\rho_{\mathbf{L}}\left(\mathbf{r}'\right) \, d\mathbf{r}\, d\mathbf{r}' = \sum_{\mathbf{L}} \left(\delta\rho \mid \delta\rho_{\mathbf{L}}\right) = \sum_{\mathbf{L}} \left(\rho - \tilde{\rho} \mid \rho_{\mathbf{L}} - \tilde{\rho}_{\mathbf{L}}\right). \tag{7.8}$$

The RI approximation allows to replace four-center electron repulsion integrals (ERIs) by two- and three-center ones. In this formalism, elements $J^{\mathbf{L}}_{\mu\nu}$ of the Coulomb matrix are defined as

$$J^{\mathbf{L}}_{\mu\nu} = \sum_{\mathbf{L}'} \left(\mu\nu_{\mathbf{L}} \mid \tilde{\rho}_{\mathbf{L}'} - \rho_{\mathrm{n}\mathbf{L}'}\right), \tag{7.9}$$

where ρ_{n} denotes the unit cell nuclear charge distribution. The total Coulomb energy including the nuclear contribution is

$$J = \sum_{\mu\nu\mathbf{L}} D^{\mathbf{L}}_{\mu\nu} J^{\mathbf{L}}_{\mu\nu} - \frac{1}{2} \sum_{\mathbf{L}} \left(\tilde{\rho} + \rho_{\mathrm{n}} \mid \tilde{\rho}_{\mathbf{L}} - \rho_{\mathrm{n}\mathbf{L}}\right), \tag{7.10}$$

with the real space density matrix elements obtained by integration

$$D^{\mathbf{L}}_{\mu\nu\sigma} = \frac{1}{V_k} \int_{\mathrm{BZ}} D^{\mathbf{k}}_{\mu\nu\sigma} e^{i\mathbf{k}^{\mathrm{T}}\mathbf{L}} d\mathbf{k}, \tag{7.11}$$

of the reciprocal space density matrix

$$D^{\mathbf{k}}_{\mu\nu\sigma} = \sum_{p} f^{\mathbf{k}}_{p\sigma} \left(C^{\mathbf{k}}_{\mu p\sigma}\right)^{*} C^{\mathbf{k}}_{\nu p\sigma} \tag{7.12}$$

over the BZ with volume V_k.

Equations (7.9) and (7.10) as well as other expressions appearing in the RI scheme require calculation of infinite lattice sums of the form

$$\sum_{\mathbf{L}} (\rho_1 \mid \rho_{2\mathbf{L}}) , \tag{7.13}$$

where the distribution ρ_1 in the central cell interacts with an infinite number of distributions $\rho_{2\mathbf{L}}$, *i.e.*, ρ_2 translated by all possible direct lattice vectors $\mathbf{L}$. In the RI-CFMM scheme [172, 173] the sum in Eq. (7.13) is partitioned into crystal far-field (CFF) and crystal near-field (CNF) parts. The CFF part contains summation over all direct space lattice vectors $\mathbf{L}$ for which the overlap between the distributions ρ_1 and $\rho_{2\mathbf{L}}$ is negligible. This part is very efficiently calculated using multipole expansions. The CNF contribution is evaluated using an octree based algorithm. In short, a cubic parent box enclosing all distribution centers of ρ_1 and ρ_2 is constructed that is large enough to yield a predefined number n_{targ} of distribution centers per lowest level box. The parent box is successively subdivided in half along all Cartesian axes yielding the octree. In the next step, all charge distributions comprising ρ_1 and ρ_2 are sorted into boxes based on their extents. Interactions between charges from well-separated boxes are calculated using a hierarchy of multipole expansions. Two boxes are considered well-separated if the distance between their centers is greater than sum of their lengths times 0.5×`wsicl`, where `wsicl` is a predefined parameter ≥ 2. The remaining contribution to the Coulomb term is obtained from direct integration. This approach results in nearly linear scaling of the computational effort with the system size.

7.2.3 k Point Sampling Scheme

The integral in Eq. (7.11) is evaluated approximately using a set of sampling points $\mathbf{k}$. `riper` uses a Γ-point centered mesh of $\mathbf{k}$ points with weights $w_{\mathbf{k}}$, so that Eq. (7.11) can be written as

$$D^{\mathbf{L}}_{\mu\nu\sigma} \approx \sum_{\mathbf{k}} w_{\mathbf{k}} e^{i\mathbf{k}^{\mathrm{T}}\mathbf{L}} D^{\mathbf{k}}_{\mu\nu\sigma} . \tag{7.14}$$

In 3D periodic systems each sampling point is defined by its components k_1, k_2 and k_3 along the reciprocal lattice vectors $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ as

$$\mathbf{k} = k_1\mathbf{b}_1 + k_2\mathbf{b}_2 + k_3\mathbf{b}_3 . \tag{7.15}$$

For 2D periodic systems $k_3 = 0$. In case of 1D periodicity $k_3 = 0$ and $k_2 = 0$. In `riper` the three components k_j $(j = 1, 2, 3)$ of $\mathbf{k}$ are given as

$$k_j = \frac{i}{n_j} \text{ with } i = -\frac{n_j - 1}{2}, -\frac{n_j - 1}{2} + 1, \ldots, \frac{n_j - 1}{2} - 1, \frac{n_j - 1}{2} . \tag{7.16}$$

with n_j $(j = 1, 2, 3)$ as integer numbers. `riper` reduces the number of $\mathbf{k}$ points employed in actual calculation by a factor of two using time-inversion symmetry, i.e., the vectors $\mathbf{k}$ and $-\mathbf{k}$ are symmetry equivalent. The $\mathbf{k}$ point mesh can be specified providing the integer values n_j within the data group `$kpoints`.

The number of **k** points required in a calculation critically depends on required accuracy. Generally, metallic systems require considerably more **k** points than insulators to reach the same precision. For metals, the number of **k** point also depends on parameters of the Gaussian smearing [178] used in `riper`. Please refer to Ref. [178] for more details.

7.2.4 Metals and Semiconductors: Gaussian Smearing

Achieving reasonable accuracy of DFT calculations for metals requires a higher number of **k** points than for semiconductors and insulators. The convergence with respect to the number of **k** points can be improved applying partial occupancies [178]. To achieve this, `riper` uses the Gaussian smearing method in which occupation numbers $f_{n\mathbf{k}}$ are calculated as

$$f_{n\mathbf{k}}\left(\frac{\epsilon_{n\mathbf{k}}-\mu}{\sigma}\right)=\frac{1}{2}\left(1-\mathrm{erf}\left[\frac{\epsilon_{n\mathbf{k}}-\mu}{\sigma}\right]\right)\ , \tag{7.17}$$

where $\epsilon_{n\mathbf{k}}$ are band (orbital) energies, μ is the Fermi energy and σ is the width of the smearing.

When smearing is applied the total energy E has to be replaced a generalized free energy F

$$F=E-\sum_{n\mathbf{k}}\sigma S(f_{n\mathbf{k}}) \tag{7.18}$$

in order to obtain a variational functional. `riper` output file reports the values of F as "FREE ENERGY" and the term $-\sum_{n\mathbf{k}}\sigma S(f_{n\mathbf{k}})$ as "T*S". In addition, the value of E for $\sigma\rightarrow 0$ is given as "ENERGY (sigma->0)".

Gaussian smearing can be switched on for `riper` calculations by simply providing the value of $\sigma>0$ within the `$riper` data group using the keyword `sigma`, e.g., `sigma 0.01`. The value of σ should be as large as possible, but small enough to yield negligible value of the "T*S" term. Note, that the value of `sigma` has to be provided in atomic units. Please refer to Ref. [178] for a more detailed discussion.

The use of Gaussian smearing often requires much higher damping and orbital shifting. Please adjust the values for `$scfdamp` and `$scforbitalshift` if you encounter SCF convergence problems.

The optional keyword `desnue` can be used within the `$riper` data group to constrain the number of unpaired electrons. This can be used to force a certain multiplicity in case of an unrestricted calculation, e.g., `desnue 0` for singlet and `desnue 1` for doublet.

7.2.5 Low-Memory Iterative Density Fitting Method

For calculations on very large molecular systems a low-memory modification of the RI approximation has been implemented within the `riper` module [175]. In the RI approximation minimization of the Coulomb repulsion of the residual density, Eq. (7.8), yields a system of linear equations

$$\mathbf{Vc}=\boldsymbol{\gamma}, \tag{7.19}$$

where $\mathbf{V}$ is the Coulomb metric matrix with elements $V_{\alpha\beta} = (\alpha \mid \beta)$ representing Coulomb interaction between auxiliary basis functions and vector $\boldsymbol{\gamma}$ is defined as

$$\gamma_\alpha = \sum_{\mu\nu} (\alpha \mid \mu\nu)\, D_{\mu\nu}. \tag{7.20}$$

In the LMIDF approach a conjugate gradient (CG) iterative method is used for solution of Eq. (7.19). In order to decrease the number of CG iterations a preconditioning is employed, *i.e.*, Eq. (7.19) is transformed using a preconditioner $\mathbf{P}$ to an equivalent problem

$$\left(\mathbf{P^{-1}V}\right)\mathbf{c} = \mathbf{P^{-1}}\boldsymbol{\gamma} \tag{7.21}$$

with an improved condition number resulting in faster convergence of the CG method. The iterative CG solver in `riper` employs one of the following preconditioners that are formed from blocks of the $\mathbf{V}$ matrix corresponding to the strongest and most important interactions between the auxiliary basis functions such that $\mathbf{P^{-1}V} \approx \mathbf{I}$:

- atomic block preconditioner

$$P^{at}_{\alpha\beta} = \begin{cases} \left(\alpha^I \mid \beta^I\right), & I \in A_I, A_I \text{ are all atoms in molecule} \\ 0, & \text{otherwise} \end{cases}$$

- ss block preconditioner: $P^{at}_{\alpha\beta} \cup P^{ss}_{\alpha\beta}$

$$P^{ss}_{\alpha\beta} = \begin{cases} (\alpha \mid \beta), & \alpha, \beta \in \{S\}, S \text{ are all s auxiliary basis functions} \\ 0, & \text{otherwise} \end{cases}$$

- sp block preconditioner: $P^{at}_{\alpha\beta} \cup P^{sp}_{\alpha\beta}$

$$P^{sp}_{\alpha\beta} = \begin{cases} (\alpha \mid \beta), & \alpha, \beta \in \{S, P\}, P \text{ are all p auxiliary basis functions} \\ 0, & \text{otherwise} \end{cases}$$

The costly matrix-vector products of the $\mathbf{Vc}$ type that need to be evaluated in each CG iteration are not calculated directly. Instead, the linear scaling CFMM implementation presented above is applied to carry out this multiplication since the elements of the $\mathbf{Vc}$ vector represent Coulomb interaction between auxiliary basis functions α and an auxiliary density $\tilde{\rho}$

$$\left(\mathbf{Vc}\right)_\alpha = \sum_\beta (\alpha \mid \beta)\, c_\beta = \left(\alpha \mid \sum_\beta c_\beta \beta\right) = (\alpha \mid \tilde{\rho}). \tag{7.22}$$

Hence, in contrast to conventional RI neither the $\mathbf{V}$ matrix nor its Cholesky factors need to be stored and thus significant memory savings are achieved.

7.2.6 RT-TDDFT

To investigate the electron dynamics in real time, RT-TDDFT based on Magnus propagator is implemented in `riper` module [177]. In RT-TDDFT, the time evolution of electron density

$\rho(\mathbf{r},t)$, represented by the single particle reduced density matrix $\mathbf{D}(t)$ with elements

$$D_{\mu\nu}(t) = \sum_{m=1}^{N_{\mathrm{MO}}} f_m C^{\dagger}_{\mu m}(t) C_{\nu m}(t) \tag{7.23}$$

is governed by the von Neumann equation

$$i\frac{\partial \mathbf{D}(t)}{\partial t} = [\mathbf{F}(t), \mathbf{D}(t)] \tag{7.24}$$

where F(t) is the time-dependent KS matrix in the orthonormal basis of MO. The von Neumann equation (7.24) is efficiently integrated numerically using the Magnus expansion which evolves the density matrix in time using a unitary operator $\mathcal{U}(t+\Delta t, t) = \mathrm{e}^{\Omega_1+\Omega_2+\Omega_3+\cdots}$ that conserves the idempotency of $\mathbf{D}(t)$. Second and fourth order Magnus expansions are implemented.

External perturbation is provided in the form of an electric field $\mathbf{E}$ which is assumed to be uniform over the whole molecule. The elctric field contribution to the KS matrix can be written as

$$\mathbf{F}^{\mathrm{E}}_{\mu\nu} = -\sum_{j=x,y,z} M^{j}_{\mu\nu} E_j, \quad j = x, y, z \tag{7.25}$$

with the electric field vector (E_x, E_y, E_z) and the dipole moment matrices $\mathbf{M^j}$

$$M^{j}_{\mu\nu} = -\int \mu(\mathbf{r}) j\nu(\mathbf{r}) \mathrm{d}\mathbf{r} \tag{7.26}$$

Two time integration methods have been implemented, the self-consistent field (SCF) procedure and the predictor-corrector (PC) scheme.

In the SCF procedure, starting from the ground state electron density $\mathbf{D}(0)$ and KS $\mathbf{F}(0)$ matrices, a guess for $\mathbf{F}(t+\frac{\Delta t}{2})$ is made through linearly extrapolation. Next, $\mathbf{D}(t)$ is propagated and used to calculate $\mathbf{F}(t+\Delta t)$, which is followed by a linear interpolation for a better guess for $\mathbf{F}(t+\frac{\Delta t}{2})$ until convergence is achieved.

In predictor-corrector scheme, $\mathbf{F}(t+\Delta t/4)$ is predicted by linear extrapolation from previous values and this is used to step $\mathbf{D}$ forward by $\Delta t/2$. In contrast to the SCF procedure, which requires multiple KS matrix builds per time step, the PC scheme requires no new KS builds and is comparatively cheaper. However, it is prone to unstability for longer time steps.

Absorption spectra is calculated using the following expression for the dipole strength function

$$S(\omega) = \frac{1}{3} \cdot \frac{4\pi\omega}{c} \mathrm{Tr}\left(\mathrm{lm}\left[\alpha_{ij}\right]\right), \quad i, j = x, y, z \tag{7.27}$$

where α_{ij} is the complex polarizability tensor, given as

$$\alpha_{ij}(\omega) = \frac{\int_{-\infty}^{\infty} \mathrm{e}^{i\omega t} \mu^{\mathrm{ind}}_j(t) \mathrm{e}^{-\gamma \mathrm{t}} \mathrm{d}t}{\int_{-\infty}^{\infty} \mathrm{e}^{i\omega t} E_i(t) \mathrm{d}t} \tag{7.28}$$

where γ is the damping factor (typically in the range of $0.003 - 0.005\mathrm{au} = 124 - 207\mathrm{ps}^{-1}$), $\mu^{\mathrm{ind}}_j(t)$ is the time-dependent induced dipole moment and E_i is the electric field component along i direction.

7.3 How to Perform a Calculation

7.3.1 Basis Sets for Periodic Calculations

In periodic systems too diffuse basis functions result in near-singularity of the overlap matrix. Therefore, it is recommended that the exponents of the Gaussian basis sets used for periodic calculations are not smaller than 0.01. Optimally, the exponents should be larger than 0.1. The easiest way to achieve this is to remove the small exponent basis functions from the basis sets. Optionally, basis sets optimized for periodic calculations can be used, such as the ones in Ref. [179]. They are available in the TURBOMOLE basis sets library as `pob-TZVP`.

7.3.2 Prerequisites

Calculations using `riper` require the `control` file and starting orbitals generated using `define`. DFT method needs to be specified in the `$dft` data group. All LDA, GGA and meta-GGA exchange-correlation functionals including interface to the XCFun library (see Sec. 6.2) are supported. Moreover, auxiliary basis sets defined in the data group `$jbas` are required. For periodic calculations additional keywords specifying the system periodicity and number of **k** points (if used) have to be added manually to the `control` file. Optionally, the `$riper` group containing control options specific to `riper` (including the LMIDF keywords) can be added. The input preparation steps are summarized below. More detailed information about `riper` keywords are provided in Sec. 23.2.14.

7.3.3 Creating the Input File

- Run `define`: in the geometry menu choose C_1 symmetry. Create data groups `$dft` and `$jbas` using `dft` and `ri`, respectively, in the general menu.
- `riper` performs molecular (0D) calculation by default if no other options are specified. For periodic calculations (1D, 2D and 3D periodicity) specify system dimensionality using the keyword `$periodic` n ($n = 1, 2, 3$) and unit cell parameters or lattice vectors using the keyword `$cell` or `$lattice`, respectively. By default, the parameters have to be provided in atomic units and degrees. Alternatively, Å can be employed using the keyword `$cell angs` or `$lattice angs`.

  Specification of cell parameters using the `$cell` keyword depends on the periodicity of a system:

  - For 3D periodic systems six unit cell parameters $|\mathbf{a}|$, $|\mathbf{b}|$, $|\mathbf{c}|$, α, β and γ need to be defined. Here, $|\mathbf{a}|$, $|\mathbf{b}|$ and $|\mathbf{c}|$ are lengths of the appropriate cell vectors, α is the angle between vectors **b** and **c**, β is the angle between vectors **a** and **c**, and γ is the angle between vectors **a** and **b**. `riper` assumes that the cell vectors **a** and **b** are aligned along the x axis and on the xy plane, respectively.
  - For 2D periodic systems three surface cell parameters $|\mathbf{a}|$, $|\mathbf{b}|$ and γ have to be provided. Here, $|\mathbf{a}|$ and $|\mathbf{b}|$ are lengths of the appropriate cell vectors and γ is

the angle between **a** and **b**. `riper` assumes that the cell vectors **a** and **b** are aligned along the x axis and on the xy plane, respectively.

– For 1D periodic systems only one parameter specifying the length of the unit cell has to be provided. `riper` assumes that periodic direction is along the x axis.

Alternatively, lattice vectors can be provided using the `$lattice` keyword:

– For 3D periodic systems three (three-dimensional) lattice vectors have to be specified.

– For 2D periodic systems two (two-dimensional) lattice vectors have to be provided. `riper` assumes that the lattice vectors are aligned on the xy plane.

– For 1D periodic systems only one parameter specifying the length of the lattice vector has to be provided. `riper` assumes that periodic direction is along the x axis.

Example 1: 1D periodic system with a unit cell $|\mathbf{a}| = 5$ bohr:

```
$periodic 1
$cell
  5.0
```

The same input using the `$lattice` keyword:

```
$periodic 1
$lattice
  5.000000000000000
```

Example 2: 2D periodic system with a unit cell $|\mathbf{a}| = 5$ and $|\mathbf{b}| = 8$ bohr, the angle between **a** and **b** of 60°:

```
$periodic 2
$cell
  5.0 8.0 60.0
```

Similar input using the `$lattice` keyword:

```
$periodic 2
$lattice
  5.000000000000000  0.000000000000000
  6.928203230275509  4.000000000000000
```

Example 3: 3D periodic system with a triclinic unit cell $|\mathbf{a}| = 5$, $|\mathbf{b}| = 8$ and $|\mathbf{c}| = 6$ bohr, the angle between **b** and **c** of 70°, between **a** and **c** of 60°, and between **a** and **b** of 90°:

```
$periodic 3
$cell
  5.0 8.0 6.0 70.0 60.0 90.0
```

Similar input using the `$lattice` keyword:

```
$periodic 3
$lattice
  5.000000000000000  0.000000000000000  0.000000000000000
  6.928203230275509  4.000000000000000  0.000000000000000
  4.328764234233443  3.324345345565466  6.354331231232453
```

- When **k** points are used specify their number using the keyword `$kpoints`. You have to specify the number of **k** points components along each periodic direction. Alternatively, a custom set of **k** points can also be provided.
- Optionally, create the `$riper` data group and add relevant keywords provided in Sec. 23.2.14.

The following examples illustrate the additions to the `control` file required for calculations using the `riper` module.

Example 1: In this example a 3D periodic system is defined (`$periodic 3`). The unit cell is specified using the `$cell` keyword, with lengths and angles given in atomic units and degrees, respectively. A uniform grid of 27 $(3 \cdot 3 \cdot 3)$ **k** points for numerical integration over the BZ is specified using the keyword `$kpoints`. The section `$riper` (see Sec. 23.2.14) is added with the keyword `lenonly` set to `on`. It forces `riper` to skip the calculation of energy gradients (by default both energy and gradients are calculated in one run). In addition, Gaussian smearing is switched on by providing `sigma 0.01` (see Sec. 7.2.4).

```
$periodic 3
$cell
  18.5911 16.5747 16.5747 90. 90. 90.
$kpoints
  nkpoints 3 3 3
$riper
  lenonly on
  sigma 0.01
```

The same input using the `$lattice` keyword:

```
$periodic 3
$lattice
  18.5911   0.0000   0.0000
    0.0000 16.5747   0.0000
    0.0000   0.0000 16.5747
$kpoints
  nkpoints 3 3 3
$riper
  lenonly on
  sigma 0.01
```

Example 2: In this example a 2D periodic system is defined (`$periodic 2`). The unit cell is specified using the `$cell` keyword, with lengths and angles given in Å and degrees, respectively. A uniform grid of 9 ($3 \cdot 3$) k-points for numerical integration over the BZ is defined using the keyword `$kpoints`. The section `$riper` (see Sec. 23.2.14) is added with the keyword `lmaxmom 30`. This changes the maximum order of multipole expansions used in CFMM to 30 (default value is 20).

```
$periodic 2
$cell angs
  18.5911 16.5747 90.0
$kpoints
  nkpoints 3 3
$riper
  lmaxmom 30
```

The same input using the `$lattice` keyword:

```
$periodic 2
$lattice angs
  18.5911   0.0000
    0.0000 16.5747
$kpoints
  nkpoints 3 3
$riper
  lmaxmom 30
```

Example 3: In this example a 1D periodic system is defined (`$periodic 1`). The dimension of the periodic direction in atomic units is specified using the `$cell` keyword. A one dimensional grid of 3 k-points for numerical integration over the BZ is defined using the keyword `$kpoints`.

```
$periodic 1
$cell
  18.5911
$kpoints
  nkpoints 3
```

The same input using the `$lattice` keyword:

```
$periodic 1
$lattice
  18.5911
$kpoints
  nkpoints 3
```

Example 4: In this example a molecular system is defined (default if no `$periodic` is specified). The section `$riper` (see Sec. 23.2.14) is added with the keyword `lpcg on`, which activates the low-memory modification of the RI approximation for calculations on very large molecular systems.

```
$riper
  lpcg on
```

Example 5: In this example a cubic 3D periodic system is defined (`$periodic 3`). The unit cell is specified using the `$cell` keyword, with lengths and angles given in atomic units and degrees, respectively. A set of 5 custom **k** points ($k_x(i)$, $k_y(i)$, $k_z(i)$) along with their weights ($w(i)$) for numerical integration over the BZ are supplied using the keyword `$kpoints` followed by the option `custom nks`. One can refer to the literature for a list of special **k** points and the corresponding weights for a particular Bravais lattice. Weights are integers that are renormalized internally such that the sum of all weights is 1. Therefore, only the relative ratios of the weights matter in a calculation.

```
$periodic 3
$cell
  16.5747 16.5747 16.5747 90. 90. 90.
$kpoints
  custom 5
  0.00  0.00  0.00   1 #Gamma point
  0.25  0.25  0.25   1 #Arbitrary point
  0.50  0.50  0.50   1 #R point
  0.00  0.50  0.00   1 #X point
```

7.3.4 Single Point Energy and Gradient

By default `riper` calculates single point energy and gradient in one run. For this, simply invoke `riper` as

```
nohup riper > riper.out &
```

For energy calculation only add `lenonly on` to the `$riper` section in the `control` file, *i.e.*,

```
$riper
  lenonly on
```

The parallel OpenMP version of `riper` can be invoked ad described in Sec. 3.4.2.

7.3.5 Structure Optimization

`jobex` will automatically use `riper` when the keyword `$periodic` is present in the `control` file. Alternatively, the use of `riper` can be forced by specifying `-riper` argument of `jobex`, *i.e.*, invoking

```
nohup jobex -riper > jobex.out &
```

Note, that at present only structure optimization using Cartesian coordinates is possible when using periodic boundary conditions. Using internal coordinates for periodic systems will lead to convergence problems in structure optimizations.

7.3.6 Optimization of Cell Parameters

Simultaneous optimization of atomic positions and cell parameters or lattice vectors can be performed by specifying the keyword `$optcell` in the `control` file and then invoking the `jobex` script as described in 7.3.5.

The calculation of energy first derivatives with respect to lattice vectors usually requires higher accuracy than nuclear gradients. Therefore, in case of convergence issues it is recommended to decrease the SCF convergence threshold to at least 1.0×10^{-7} by specifying `$scfconv 7`. In addition, convergence can often be improved by including weight derivatives by adding the option `weight derivatives` in the `$dft` data group.

7.3.7 Band Structure Plots

Band structure plots show the values of band (orbital) energies for values of **k** along lines connecting symmetry points. In `riper` these lines can be specified by providing their start and end points as well as the number of **k** points along the line within the `$kpoints` section. They will be calculated at the end of SCF procedure and written to the file `bands.xyz`.

In the following example band energies are calculated along four lines, as specified by the keyword `kptlines 4`. Each line definition starts in a new line with the keyword `recipr`, followed by three real numbers defining the start point of the line and three real numbers defining its end point. Finally, the number of **k** points along the line is given as an integer number. Thus, the first line starts at the point `(0.500 0.500 0.500)`, ends at `(0.000 0.000 0.000)` and contains 40 **k** points.

```
$kpoints
 kptlines 4
 recipr 0.500 0.500 0.500 0.000 0.000 0.000 40
 recipr 0.000 0.000 0.000 0.500 0.500 0.000 40
 recipr 0.500 0.500 0.000 0.746 0.373 0.373 40
 recipr 0.746 0.373 0.373 0.000 0.000 0.000 40
```

Each line of the resulting `bands.xyz` file contains five real numbers: the coordinates k_1, k_2 and k_3 of the $\mathbf{k}$ point, its length $|\mathbf{k}|$ and the corresponding band energy $\epsilon_{n\mathbf{k}}$.

7.3.8 Calculation of Densities and MOs on Grids

How to Perform

Calculation of data on grids for molecular and periodic systems using `riper` requires the keyword `$pointvalper` in the `control` file. The values are calculated on a 3D grid and written to appropriate output files. These files can be generated either running a single point energy `riper` calculation or invoking

```
nohup riper -proper > riper.out &
```

provided that converged orbitals and occupation numbers from previous calculation are present in the files `RIPER.BANDS` and `RIPER.BANDS.OCCUPATIONS`, respectively.

The format of the output files can be specified using the keyword `fmt=` following `$pointvalper` in the same line. Supported formats are:

`plt` (default) The generated data is written to binary files that can be read by gOpenMol and other external visualization programs. This format uses orthogonal grids. Therefore, for non-orthogonal unit cells grid data is generated for a rectangular box that contains the supercell (unit cell and its periodic images). By default, the values at grid points outside of the supercell are set to zero. For strongly non-orthogonal systems this may lead to large files. The option `full` switches off the zeroing of values on grid points outside the supercell.

`cub` Gaussian cube format. Can be imported by several external visualization programs.

`xyz` Coordinates of grid points and calculated values are stored in a text file. Each line contains Cartesian grid point coordinates and the corresponding value.

`upt` Values and grid data is output to binary files. First 8 (integer) records are: 1-2) rank value and descriptor, 3-8) number of grid points and number of periodic unit cell images in each periodic direction, 9...) Cartesian coordinates and the corresponding value repeated for all grid points.

Output files in `cub` format contain information about atomic coordinates. For other formats additional `coord.xyz` file containing atomic coordinates is generated.

The following options can be used along with the keyword `$pointvalper`:

`nimg n1 n2 n3`
: Number of unit cell images `n1`, `n2` and `n3` in the periodic directions **a**, **b** and **c**, respectively, for which plot data is generated.

`npts n1 n2 n3`
: Number of grid points `n1`, `n2` and `n3` along each periodic direction. If not specified, value 100 is used for each `n`.

`eps real`
: Specifies the distance `real` in bohr around the system for which plot grid is generated in aperiodic directions. Default value is 5 bohr.

`ngrdpbx n`
: Number of grid points stored in one octree box during density calculations. Default value is 50. For very large systems or high resolution it may be necessary to increase this parameter to avoid memory allocation problems.

`full`
: Only valid for `plt` output format. Switches off zeroing of values on grid points outside the supercell.

Electron Density

Plots of electron density require option `dens` in the `$pointvalper` data group

```
$pointvalper
  dens
```

The values of the total density and, in case of UHF calculations also the spin density are written to files `td.fmt` and `sd.fmt`, respectively, where `fmt` is the selected format.

Molecular Orbitals

Plot files for visualization of molecular orbitals can be generated by setting `orbs n` in the `$pointvalper` data group, where `n` is the number of orbitals to plot. The way to specify the orbitals is best explained by some examples:

example 1: An open shell system is considered. Number of **k** points along the reciprocal lattice vectors $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ is 3, 5, and 7, respectively. Possible **k** point components k_j calculated from equation 7.16 are therefore $k_1 \in \left(\frac{-1}{3}, 0, \frac{1}{3}\right)$, $k_2 \in \left(\frac{-2}{5}, \frac{-1}{5}, 0, \frac{1}{5}, \frac{2}{5}\right)$, and $k_2 \in \left(\frac{-3}{7}, ..., \frac{3}{7}\right)$ The `kpoints` and `$pointvalper` groups are:

```
$kpoints
 nkpoints 3 5 7
$pointvalper
  orbs 2
  k 1 4 6 a 1
  k 2 3 4 b 3
```

Here, two orbitals will be plotted. They are defined in two lines following the `orb 2` option. Each line starts with `k` followed by three numbers that specify which of possible components k_j are used. In this example **k** point components for the first orbital are $\left(\frac{-1}{3}, \frac{1}{5}, \frac{2}{7}\right)$. For the second one $\mathbf{k} = (0, 0, 0)$, *i.e.*, this orbital is at the Γ-point. Subsequently, type and number of the orbitals are defined (orbitals at each **k**-point are ordered by increasing energy). Here the first one is `a 1` - the lowest α orbital. Similarly, `b 3` denotes the third lowest energy β orbital. For a non-Γ **k** point, the calculation will generate three plottable files corresponding to the: (i) real component of Ψ, (ii) imaginary component of Ψ, and (iii)Ψ^2, for a given orbital. Note: Only the real components are plotted for the Γ-point.

example 2: In this example a closed shell system with 3, 2 and 4 **k** points in each direction is given:

```
$kpoints
 nkpoints 3 2 4
$pointvalper
 orbs 3
 k 1 2 1 a 1
 k 0 0 0 a 2
```

Possible **k** point components from equation 7.16 are $k_1 \in \left(\frac{-1}{3}, 0, \frac{1}{3}\right)$, $k_2 \in \left(\frac{-1}{4}, \frac{1}{4}\right)$ and $k_2 \in \left(\frac{-3}{8}, \frac{-1}{8}, \frac{1}{8}, \frac{3}{8}\right)$. In analogy to the previous example, three data files for the real component, imaginary component, and square of the wavefunction (MO) will be generated for the lowest energy orbital at **k** point $\left(\frac{-1}{3}, \frac{1}{4}, \frac{-3}{8}\right)$. For the second molecular orbital only the real component will be generated at the Γ-point. Orbitals at the Γ-point are defined by using 'special' `0 0 0` component indices, as it is not included in the grid for even number of **k** points.

example 3: In this example an input for a Γ-point calculation is shown:

```
$pointvalper fmt=plt
 orbs 2
 k 1 1 1 a 5
 k 1 1 1 b 5
```

Here `k 1 1 1` is the only valid option. For Γ-point calculation, orbitals are real, thus it is the only component that is plotted. In this case the fifth lowest energy orbitals with spin components α and β are plotted. The output format is set to `plt` using the `fmt=` keyword.

example 4: In this example an input for a `custom` **k** point calculation is shown:

```
$kpoints
 custom 2
 0.00  0.00  0.00   1
 0.50  0.50  0.25   1
$pointvalper fmt=cub
```

```
orbs 2
k 1 a 5
k 2 a 6
```

Instead of three indices after `k` (like in the previous examples), only a single index is required, specifying the index of the custom **k** point in the list given by the user. The above example will plot the fifth lowest energy molecular orbital at the first **k** point (Γ) and the sixth MO at the second, (`0.50 0.50 0.25`) **k** point. Note: If the list of custom **k** points doesn't include a Γ-point, one can still plot the MOs at this point, as a Γ-point with 0 weight is prepended to the supplied list with an index 0. In such a case, to plot the fifth lowest energy orbital with spin α at the Γ-point, the input would be `k 0 a 5` .

7.3.9 Density of States

A simulated density of states (DOS) is calculated by broadening the discrete energy levels with Gaussians and superimposing them. The output files (`dos` for RKS calculations, and `dos_a+b`, `dos_a-b`, `dos_alpha`, `dos_beta` in case of UKS) contain energies (first column) and corresponding total DOS (second column) as well as s-, p-, ... contributions (following columns). The calculation is controlled by the keyword `$dosper`:

```
$dosper width=real emin=real emax=real scal=real npt=integer
```

The parameters in example above have the following meaning:

`width`	the width of each Gaussian, default value is 0.01 a.u.
`emin,emax`	lower/upper bounds for energy in DOS calculation
`scal`	scaling factor for DOS (total and s-, p-, ... contributions)
`npt`	resolution (number of points)

All parameters are optional, default values are usually sufficient to generate a satisfactory plot. DOS files can be generated either running a single point energy `riper` calculation or using `-proper` option

```
nohup riper -proper > riper.out &
```

provided that converged orbitals and occupation numbers from previous calculation are present in the files `RIPER.BANDS` and `RIPER.BANDS.OCCUPATIONS`, respectively.

7.3.10 RT-TDDFT

How to Perform

RT-TDDFT calculations require data-groups: `$fields`, `$electric field` and `$rttddft` to provide the information about the type of field, electric field and the RT-TDDFT calculation, respectively. Additionally, keywords `$rtdipol` and `$rtenergy` followed by file names may

be provided in the `control` file to print out the dipole moment and the energies respectively every n number of steps. It is recommended to monitor the values in these files as the calculation is running to ensure that the simulation is stable and not diverging. To calculate the absorption spectrum and print it, the keyword `$rtspectrum` followed by the units is needed. Possible values of the units are `ev`, `nm`, `cm-1`, and `hartree`. Example:

```
$rtspectrum ev
```

For now, only electric fields are supported. So, the `$fields` block would look like

```
$fields
  electric on
```

The above would enable the use of an electric field.

The information pertaining to the electric field is provided within the `$electric field` data-group.

There are currently three types of electric fields supported.

- Gaussian
- Static
- Laser

Static fields only require the Cartesian components of the amplitude. The Gaussian field requires two more parameters: peak position and peak width. Finally, the Laser field requires the Cartesian components, frequency and the full width at half maximum (FWHM) of the field envelope. Additionally, the phase may also be provided.

Note: The absorption spectrum can only be calculated for the Gaussian field.

To define a static field

```
$electric field
  amplitude x=2.0E-5 y=2.0E-5 z=2.0E-5
  static
```

To define a Gaussian ("kick") field

```
$electric field
  amplitude x=2.0E-5 y=2.0E-5 z=2.0E-5
  gaussian tzero=3.0 width=0.2
```

To define a Laser field

```
$electric field
  amplitude x=0.005d0 y=0.0050 z=0.0d0
  phase x=1.570796d0 y=0.0d0 z=0.0d0
  laser omega=0.056961 sigma=750.0d0
```

Note: the amplitudes are provided in atomic units (1 au = 514.2 V/nm). The Gaussian electric field is defined as

$$\vec{E}(t) = \vec{A}\mathrm{e}^{-\frac{(t-t_0)^2}{2w^2}} \tag{7.29}$$

`tzero` in the above code snippet corresponds to t_0 (center of the peak) and `width` corresponds to w (which gives a measure of the width of the pulse). Note: both are in atomic units of time (1 au = 0.02419 fs)

The Laser electric field is defined as

$$\mathbf{E}(t) = f(t)\left(E_x \sin\left(\omega_0 t + \phi_x\right)\mathbf{n_x} + E_y \sin\left(\omega_0 t + \phi_y\right)\mathbf{n_y} + E_z \sin\left(\omega_0 t + \phi_z\right)\mathbf{n_z}\right) \tag{7.30}$$

where $E_{x,y,z}$ are the amplitudes along x, y, z axes, ω_0 is the carrier frequency, $\phi_{x,y,z}$ are the carrier-envelope phases, and $f(t)$ is an envelope/shape function given as

$$f(t) = \begin{cases} \cos^2\left(\frac{\pi}{2\sigma}(t-\sigma)\right) & \text{if } 0 \le t \le 2\sigma \\ 0 & \text{otherwise} \end{cases}, \tag{7.31}$$

where σ is the FWHM of the field envelope. The total pulse duration is 2σ. The laser pulse reaches its maximum at $t = \sigma$, corresponding to a mean intensity of

$$I = \frac{1}{2}\varepsilon_0 c|E|^2 \text{ (SI units)}, \tag{7.32}$$

where ε is the permittivity of vacuum, $|E|^2 = E_x^2 + E_y^2 + E_z^2$, and c is the velocity of light.

`omega` in the above code snippet for Laser field corresponds to ω the frequency of the laser pulse and `sigma` corresponds to the FWHM of the laser envelope. The expected units for frequency is a.u. (Note: 1 eV = 0.0367493 a.u.). Similarly, `sigma` is expected in a.u. of time (1 au = 0.02419 fs). Finally, `phase` corresponds to the phase angle ϕ in eq. 7.30. It is expected in radians (1 rad= 57.2958^o). Phase is useful to create a circularly polarized laser pulse.

Finally, the parameters regarding the RT-TDDFT calculation go within the `$rttddft` data-group. The meanings are explained below

`magnus`
: Can take values 2 or 4. "2" for second order Magnus expansion and "4" for fourth order Magnus expansion. Default value is 2.

`scf`
: if `on`, then SCF procedure is used for the time integration. If `off` then Predictor-Corrector scheme is used instead.

`iterlim`
: Max SCF cycles if `scf` is `on`. Default value is 15.

`time`
: Specifies the evolution time in au. (1 au= 0.02419 fs)

`tstep`
: The time step for the time evolution in au. 0.1 au is usually a good starting point.

`print step`
: Specifies the number of steps n after which the dipole moments and energies are printed out if requested. Default value is 100. That means the quantities are printed out at every 100 steps. To have all the information for post processing, a value of 1 is recommended.

`damping`
: Only valid for absorption spectrum calculation. It is the factor gamma in the equation to calculate the complex polarizability tensor. Default value is 0.004 au. Recommended values in the range of 0.003 au to 0.005 au.

`min energy`
: Only valid for absorption spectrum calculation. Specifies the minimum value of the energy range used to perform the Fourier transform from time to frequency space. Units: au. Default value is 0.15 au.

`max energy`
: Only valid for absorption spectrum calculation. Specifies the maximum value of the energy range used to perform the Fourier transform from time to frequency space. Units: au. Default value is 0.625 au.

`energy step`
: Only valid for absorption spectrum calculation. Specifies the step value or energy interval `dE` at which to sample the energy values for Fourier transform and absorption spectrum plotting. Units: au. Default value is 0.005 au.

Consider the following example.

Example 1: Here, an absorption spectrum calculation for a molecule is performed using RT-TDDFT. The following keywords are expected in the `control` file.

```
$fields
  electric on
$electric field
  amplitude x=2.0E-5 y=2.0E-5 z=2.0E-5
  gaussian tzero=3.0 width=0.2
$rttddft
  magnus 2
  scf off
  time 1000.0d0
  tstep 0.1d0
  print step 1
  min energy = 0.013d0
  max energy = 0.5d0
  energy step 0.001d0
$rtdipol
```

```
$rtenergy
$rtspectrum ev
```

The data-group `$fields` is used to specify that only electric field is needed using `electric on`. Here, a Gaussian "kick" pulse is used to excite the system. The parameters for which are specified in the `$electric field` section. The pulse has a small amplitude in all three directions. The amplitude is specified in au (1 au = 514.2 V/m). The position of the maximum of the pulse is at t =3 au, specified using `tzero`. The width of the pulse is specified using `width` parameter Eq. (7.29). The simulation goes on for 1000 au (1 au= 0.02419 fs) in steps of 0.1 au as specified by the `time` and `tstep` keywords in the `$rttddft` section, respectively. Second order Magnus expansion is used for the time-evolution operator indicated by `magnus 2`. Predictor-corrector scheme is used for time-integration since `scf` is `off`. Notice here, that we are also requesting to print out energies and dipole moments using the `$rtenergy` and `$rtdipol` keywords, respectively. The default names of the corresponding files are `rtenrgy` and `rtdipo`. To print out to a custom file name use `$rtenergy file filename` instead. It is recommended to have these files printed as they allow to monitor the energies in real time and an unstable simulation can be detected early. The `print step 1` in the `$rttddft` section specifies that the dipole moments and the energies are printed at each iteration, i.e. every step. This is useful if post-processing is needed after the calculation. Default value of `print step` is 100. Finally, to calculate the spectrum, we provide the `$rtspectrum` keyword followed by the units (in this case electron volts). The energy range for which the absorption spectrum is calculated is 0.35 eV (0.013 au) to 13.6 eV (0.5 au), as specified using the `min energy` and `max energy` keywords with the step interval of 0.001 au. The spectrum is printed out in a file called `rtspec`.
The calculation is performed by running
`nohup riper > riper.out &`
To keep a track of the progress of the calculation, i.e. how many steps have been completed, check the `rtenrgy` file.
The RT-TDDFT absorption spectrum can be plotted from the `rtspec` file after the calculation ends.

Example 2: In this example the electron density is plotted at each time step, to visualize the electron dynamics in real-time. The following keywords are expected in the `control` file.

```
$fields
  electric on
$electric field
  amplitude x=2.0E-5 y=2.0E-5 z=2.0E-5
  gaussian tzero=3.0 width=0.2
$rttddft
  magnus 2
  scf off
  time 1000.0d0
```

```
  tstep 0.5d0
  print step 1
$rtdipol
$rtenergy
$rtdens
$pointvalper fmt=cub
  dens
```

Two new keywords are noticed here compared to example 1. `$rtdens` keyword tells the program to print out the difference of the excited state and ground state density. Second, the `$pointvalper` section, explained already, is used to specify the format in which the electron density is printed out.

Chapter 8

Hartree–Fock and DFT Response Calculations: Stability, Dynamic Response Properties, and Excited States

8.1 Functionalities of ESCF and EGRAD

`escf` and `egrad` are designed as efficient tools for response and excited state calculations on large molecules. `escf` serves to compute the following properties for HF and KS reference states:

- Eigenvalues of the electronic Hessian (stability analysis)
- Frequency-dependent polarizabilities and optical rotations
- Frequency-dependent electronic hyperpolarizabilities
- Vertical electronic excitation energies (TD-DFT)
- Vertical electronic excitation energies from the Bethe-Salpeter equation
- Transition moments, oscillator and rotatory strengths of electronic excitations
  ⇒ UV-VIS and CD spectra
- Two-photon transition moments
  ⇒ 2PA spectra
- Nuclear spin-spin coupling constants (SSCCs)
- Damped response TD-DFT and BSE
- Two-component TD-DFT and *GW*-BSE including spin-orbit coupling
- TD-DFT and *GW*-BSE in magnetic fields

Spin-restricted closed-shell and spin-unrestricted ground states (except for stability analysis) are supported. For 2c calculations, TD-DFT supports Kramers symmetric references, while *GW*-BSE supports all references. The RI-J approximation in conjunction with LDA, GGA, and meta-GGA (MGGA) functionals is implemented for all properties. The seminumerical semiJK algorithm is also available for all properties. Excitation energies and transition moments can be computed either within the full time-dependent HF (TDHF) or time-dependent DFT (TDDFT) formalisms or within the Tamm-Dancoff approximation (TDA).

Excited state first order properties can be evaluated analytically using `egrad`. They include:

- Gradients of the excited state energy with respect to nuclear positions
  ⇒ Excited state equilibrium structures (`jobex`), adiabatic excitation energies, emission spectra
- Exited state densities ⇒ Charge moments, population analysis
- Excited state force constants by numerical differentiation of gradients (using the script `NumForce`)
- First-order derivative couplings between the ground and an excited state as well as between two excited-states (state-to-state)
- Transition moments, oscillator strengths between two TDHF/TDDFT excited-states (state-to-state)

Moreover, analytical gradients of static and frequency-dependent polarizabilities are available from `egrad`. Together with vibrational normal modes from the `aoforce` or `NumForce` they are used to calculate vibrational Raman intensities. Excited state gradients for *GW*-BSE are presently unavailable.

Again, ground states may be spin-restricted closed-shell or spin-unrestricted, RI-J is available, and either full TDDFT/TDHF or the TDA can be used. For further details we refer to a recent review [180].

8.2 Theoretical Background

We briefly state the basic working equations in the following, as far as required to understand the program output. For a more detailed treatment of the theory see refs. [38, 180–184] and refs. therein. The following discussion is restricted to the one-component (nonrelativistic) treatment, for the sake of convenience.

The first-order frequency dependent response of the density matrix can be expanded as

$$\gamma(x,x') = \sum_{ai} \{X_{ai}\varphi_i(x)\varphi_a^*(x') + Y_{ai}\varphi_a(x)\varphi_i^*(x')\}. \tag{8.1}$$

The (real) expansion coefficients X_{ai} and Y_{ai} are conveniently gathered in a "super-vector"

$$|X,Y\rangle = \begin{pmatrix} X \\ Y \end{pmatrix} \tag{8.2}$$

on L, the linear space of products of occupied and virtual ground state MOs $\varphi_i(x)\varphi_a^*(x')$ plus their complex conjugates. X and Y describe the first-order change of the ground state MOs due to an external perturbation which is represented by $|P,Q\rangle$ on L. For example, if an oscillating electric dipole perturbation along the z axis is applied, $|P,Q\rangle = |\mu_z\rangle$, where μ is the electric dipole operator.

Next we define the 2×2 "super-matrices"

$$\Lambda = \begin{pmatrix} A & B \\ B & A \end{pmatrix}, \quad \Delta = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{8.3}$$

where the four-index quantities A and B are the so-called "orbital rotation Hessians". Explicit expressions for the standard A and B can be found, e.g., in ref. [38]. For MGGA functionals, the linear response of the paramagnetic current density leads to additional XC kernel matrix elements, and subsequently to modified definitions of A and B [185]. The vector $|X,Y\rangle$ is determined as the solution of the TDHF/TDDFT response problem,

$$(\Lambda - \omega\Delta)|X,Y\rangle = -|P,Q\rangle. \tag{8.4}$$

If $|X_\alpha, Y_\alpha\rangle$ arises from an electric dipole perturbation $|\mu_\alpha\rangle$, the electronic dipole polarizability at frequency ω is

$$\alpha_{\alpha\beta}(\omega) = -\langle X_\alpha, Y_\alpha|\mu_\beta\rangle, \tag{8.5}$$

$\alpha, \beta \in \{x, y, z\}$. Similarly, if $|m_\alpha\rangle$ is a component of the magnetic dipole moment operator, the optical rotation is [186]

$$\delta_{\alpha\beta}(\omega) = -\tfrac{c}{3\omega}\mathrm{Im}\langle X_\alpha, Y_\alpha | m_\beta\rangle, \tag{8.6}$$

where c is the light velocity.

Excitation energies Ω_n are the poles of the frequency-dependent density matrix response. They are thus the zeros of the operator on the left-hand side of Eq. (8.4),

$$(\Lambda - \Omega_n \Delta)|X_n, Y_n\rangle = 0. \tag{8.7}$$

The corresponding eigenvectors $|X_n, Y_n\rangle$ are the transition density matrices for a given excitation (also called "excitation vectors" in the following). They are required to be normalized according to

$$\langle X_n, Y_n | \Delta | X_n, Y_n\rangle = 1. \tag{8.8}$$

Transition moments are evaluated by taking the trace with one-particle operators, e.g.,

$$\boldsymbol{\mu}^{0n} = \langle X_n, Y_n | \boldsymbol{\mu}\rangle \tag{8.9}$$

for the electric and

$$\mathbf{m}^{0n} = \langle X_n, Y_n | \mathbf{m}\rangle \tag{8.10}$$

for the magnetic transition dipole moments.

The full TDHF/TDDFT formalism is gauge-invariant, i.e., the dipole-length and dipole-velocity gauges lead to the same transition dipole moments in the basis set limit. This can be used as a check for basis set quality in excited state calculations. The TDA can formally be derived as an approximation to full TDHF/TDDFT by constraining the Y vectors to zero. For TDHF, the TDA is equivalent to configuration interaction including all single excitations from the HF reference (CIS). The TDA is not gauge invariant and does not satisfy the usual sum rules [181], but it is somewhat less affected by stability problems (see below). For MGGA functionals, the response of the paramagnetic current density is required to ensure gauge invariance and is included by default.

Stability analysis of closed-shell electronic wavefunctions amounts to computing the lowest eigenvalues of the electric orbital rotation Hessian $A+B$, which decomposes into a singlet and a triplet part, and of the magnetic orbital rotation Hessian $A-B$. Note that $A-B$ is diagonal for non-hybrid and non-MGGA DFT, while $A+B$ generally is not. See refs. [26, 185, 187] for further details.

The **two-component relativistic TDDFT** eigenvalue problem for excitations of (Kramers-restricted) closed-shell systems taking (approximately) into account the effect of spin-orbit coupling is

$$M\,(X+Y)_n = \Omega_n^2\,(X+Y)_n\,. \tag{8.11}$$

M is a Hermitian (complex) matrix containing spinor energy differences, Coulomb matrix elements as well as matrix elements of the two-component noncollinear exchange-correlation kernel. An explicit expression for M can be found in ref. [27]. $(X+Y)_n$ are complex

two-component excitation vectors. In the case of HF exchange, one has to resort to the TDA

$$AX_n = \Omega_n X_n, \tag{8.12}$$

see ref. [28].

Properties of excited states are defined as derivatives of the excited state energy with respect to an external perturbation. It is advantageous to consider a fully variational Lagrangian of the excited state energy [38],

$$\begin{aligned} L[X,Y,\Omega,C,Z,W] = E_{\mathrm{GS}} + \langle X,Y|\Lambda|X,Y\rangle - \Omega\big(\langle X,Y|\Delta|X,Y\rangle - 1\big) \\ + \sum_{ia} Z_{ia}F_{ia} - \sum_{pq} W_{pq}(S_{pq} - \delta_{pq}). \end{aligned} \tag{8.13}$$

Here E_{GS} denotes the ground state energy, F and S are the Fock and overlap matrices, respectively, and indices p, q run over all, occupied and virtual MOs.

First, L is made stationary with respect to *all* its parameters. The additional Lagrange multipliers Z and W enforce that the MOs satisfy the ground state HF/KS equations and are orthonormal. Z is the so-called Z-vector, while W turns out to be the excited state energy-weighted density matrix. Computation of Z and W requires the solution of a *single* static TDHF/TDKS response equation (8.4), also called coupled and perturbed HF/KS equation. Once the relaxed densities have been computed, excited state properties are obtained by simple contraction with derivative integrals in the atomic orbital (AO) basis. Thus, computation of excited state gradients is more expensive than that of ground state gradients only by a constant factor which is usually in the range of $1 \ldots 4$.

TDHF/TDDFT expressions for components of the frequency-dependent polarizability $\alpha_{\alpha\beta}(\omega)$ can also be reformulated as variational polarizability Lagrangians [188]

$$\begin{aligned} &L^{\alpha\beta}[X_\alpha, Y_\alpha, X_\beta, Y_\beta, C, Z^{\alpha\beta}, W^{\alpha\beta}](\omega) \\ &\quad = \langle X_\alpha, Y_\alpha|(\Lambda - \omega\Delta)|X_\beta, Y_\beta\rangle + \langle X_\alpha, Y_\alpha|\mu_\beta\rangle + \langle \mu_\alpha|X_\beta, Y_\beta\rangle \\ &\quad + \sum_{ia\sigma} Z^{\alpha\beta}_{ia\sigma}F_{ia\sigma} - \sum_{pq\sigma, p\le q} W^{\alpha\beta}_{pq\sigma}(S_{pq\sigma} - \delta_{pq}). \end{aligned} \tag{8.14}$$

The stationary point of $L^{\alpha\beta}(\omega)$ equals to $-\alpha_{\alpha\beta}(\omega)$. The requirement that $L^{\alpha\beta}(\omega)$ be stationary with respect to all variational parameters determines the Lagrange multipliers $Z^{\alpha\beta}$ and $W^{\alpha\beta}$. All polarizability components $\alpha\beta$ are processed simultaneously which allows for computation of polarizability derivatives at the computational cost which is only 2–3 higher than for the electronic polarizability itself.

Within TDDFT and TDHF, the X and Y coefficients are normalized as follows:

$$\sum_{ia}(|X_{ia}|^2 - |Y_{ia}|^2) = 1, \tag{8.15}$$

where i and a label occupied and virtual MOs, respectively. Thus, the squared "coefficient" of a single electron excitation from orbital i to orbital a can be defined as

$$|c_{ia}|^2 = |X_{ia}|^2 - |Y_{ia}|^2. \tag{8.16}$$

escf prints out $|c_{ia}|^2 * 100$ starting with the largest coefficient, until the sum of the coefficients is 0.9 or greater. TDA is contained as special case with Y_{ia}=0.

The first hyperpolarizability is computed from

$$\beta_{\alpha\beta\gamma}(\omega_\alpha, \omega_\beta) = \mathrm{tr}\left(K^{(\alpha\beta)} v^{(\gamma)}\right) - \langle X^{(\alpha)}, Y^{(\alpha)} | P^{(\alpha\beta)}, Q^{(\alpha\beta)} \rangle \tag{8.17}$$

where $K^{(\alpha\beta)}$ is the unrelaxed density,

$$K_{ij}^{(\alpha\beta)} = -\sum_a \left[X_{ja}^{(\alpha)} Y_{ia}^{(\beta)} + X_{ja}^{(\alpha)} Y_{ia}^{(\beta)} \right], \tag{8.18}$$

$$K_{ab}^{(\alpha\beta)} = \sum_i \left[X_{ia}^{(\alpha)} Y_{ib}^{(\beta)} + X_{ia}^{(\alpha)} Y_{ib}^{(\beta)} \right], \tag{8.19}$$

and $P^{(\alpha\beta)}$ and $Q^{(\alpha\beta)}$ are second-order right-hand-sides defined in Ref. [184].

Two-photon absorption (2PA) amplitudes are

$$\mu_{(\alpha\beta)}^{0n}(\omega) = \langle X^n, Y^n | P^{(\alpha\beta)}, Q^{(\alpha\beta)} \rangle \tag{8.20}$$

where the frequencies in the right-hand side are $\omega_\alpha = \omega$ and $\omega_\beta = \Omega_n - \omega_\alpha$.

State-to-state transition moments can be derived by examining the double residue of the quadratic response function. [184, 189] The state-to-state 1-particle transition moment (1TDM) is

$$\boldsymbol{\gamma}^{nm,\mathrm{QR}} = \begin{pmatrix} -\left(\mathbf{X}^n (\mathbf{X}^m)^T + \mathbf{Y}^n (\mathbf{Y}^m)^T\right) & \mathbf{X}^{nm} \\ (\mathbf{Y}^{nm})^T & (\mathbf{X}^n)^T \mathbf{X}^m + (\mathbf{Y}^n)^T \mathbf{Y}^m \end{pmatrix} \tag{8.21}$$

where the off-diagonal blocks require the solution of a dynamic-polarizability-like equation,

$$|X^{nm}, Y^{nm}\rangle = -\left(\boldsymbol{\Lambda} - \Omega_{nm} \boldsymbol{\Delta}\right)^{-1} |P^{nm}, Q^{nm}\rangle. \tag{8.22}$$

For **static polarizabilities** and **nuclear spin-spin coupling constants** (SSCC), a linear system of equations (LSE) of the following type has to be solved:

$$\sum_{bj} G_{ai,bj} \lambda_{bj} = -R_{ai}\,, \tag{8.23}$$

where $G = A + B$ or $G = A - B$, depending on the exact type of calculation. Details can be found in [190] (static polarizabilities) and [123, 191, 192] (SSCC). Both quantities are also available in a two-component (2c) spin–orbit formalism (polarizabilities [37], SSCCs [193, 194]). R are the right-hand side integrals which can be computed rather easily. There are three (one for each Cartesian direction) LSE for static polarizability and ten (2c: three) per nucleus for coupling constants. The final quantity of interest is a tensor obtained (in a simplified manner) as

$$K_{KL} = \sum_{ai} \lambda_{ai,K} R_{ai,L} = \sum_{ai} R_{ai,K} \lambda_{ai,L}\,, \tag{8.24}$$

where K and L are nuclei (for static polarizabilities, omit these indices).

For SSCCs, this is by default scaled by the gyromagnetic ratios of the nuclei,

$$J_{KL} = h\frac{\gamma_K}{2\pi}\frac{\gamma_L}{2\pi}K_{KL}\,. \tag{8.25}$$

8.3 Implementation

Without giving details, we discuss features of the implementation in `escf` and `egrad` that matter for applications. The interested reader is referred to the refs. given in the program headers as well as ref. [195].

Simultaneous vector iteration. The solutions of Eqs. (8.4) and (8.7) (Eq. (8.11)) are expanded in a subspace of L which is iteratively expanded (Davidson method [196]). The iteration is stopped when the Euclidean norm of the residual vector is smaller than 10^{-k}. The default for k is 5, which usually gives excitation energies accurate to $8-10$ digits and properties accurate to $4-5$ digits (k can be changed by specifying `$rpaconv` k). Several roots, i.e., several excited states or frequencies can be treated simultaneously, which is very effective and permits the calculation of whole excitation spectra and dispersion curves. During the iteration, the vectors are kept on scratch files `vfile_<IR>`,`wfile_<IR>`, and/or `rhs_<IR>`, where `IR` denotes an IRREP of the point group (see below). Before the programs terminate, the converged vectors are written onto formatted files ⟨type⟩⟨IR⟩, where type is an abbreviation for the type of response calculation performed (cf. `$scfinstab`). Given these files in the working directory, `escf` and `egrad` calculations can be restarted or continued, e.g., with a larger number of roots.

Integral direct algorithm. In the iterative method outlined above, the super-matrices A and B never need to be set up explicitly; only the products of A and B with some suitable basis vectors are required. These matrix-vector-products are evaluated very efficiently in the AO basis, because the required four-index integrals can be computed "on the fly" and need not be transformed or stored on disk. In addition, prescreening techniques based on rigorous bounds are straightforward to apply. This leads to a low-order scaling $O(N^2) - O(N)$ for the time-determining steps. Due to the similarity to ground state fock matrix construction, the same keywords are used to control these steps as in semi-direct SCF, namely `$thime`, `$thize`, `$scfintunit`, see Chapter 6. The same is true for DFT and RI keywords such as `$dft`, `$ridft`, `$ricore`.

Point group symmetry. `escf` and `egrad` can exploit point group symmetry for *all* finite point groups (with up to 99-fold symmetry axes, → `$symmetry`). The calculation of SSCCs is only possible for D_{2h} and its subgroups. The response and eigenvalue problems (eqs. (8.4) and (8.7)) decompose into separate problems for each IRREP that are solved independently. For excited state and instability calculations, it is thus necessary to specify

the IRREPs to be treated (`$soes`, see below). For response calculations, the perturbation is automatically subduced into irreducible components. The overall speedup compared to C_1 symmetry is approximately $1/g$, where g denotes the point group order. For spin-restricted closed-shell ground states, spin symmetry is used to further reduce the dimension of the response and eigenvalue problems by a factor of 2. Point group symmetry cannot be exploited in two-component calculations.

Other features. `escf` and `egrad` fully support external fields (using the keyword `$electrostatic field`; specify `geofield on` in `$fldopt`), point charges (using the keyword `$point_charges`), and effective core potentials (using `$ecp`). In `escf` calculations, occupied and virtual MOs can be frozen (using `$freeze`).

8.4 How to Perform

The most convenient way to set up an `escf` or `egrad` calculation is to use the `ex` option of the last ("general") `define` menu, see Chapter 4. `define` will automatically provide most of the keywords discussed below.

A large number of (not necessarily realistic) sample inputs is contained in the `escf` and `egrad` subdirectories of the test suite (`TURBOTEST` directory).

8.4.1 Preliminaries

All response calculations require a complete set of converged (occupied and virtual) SCF MOs. It is strongly recommended to use *well converged MOs*, since the error in the ground-state wavefunction enters linearly in all response properties. Thus, before starting `escf` or `egrad`, specify the keywords

```
$scfconv 7
$denconv 1d-7
```

in `control`, perform a `dscf` statistics run if semi-direct integral processing is to be used (see Chapter 3), and (re-)run `dscf` or `ridft`,

```
dscf > dscf.out &
```

or

```
ridft > ridft.out &
```

in case of RI-J.

The above tight convergence criteria are also recommended for excited state geometry optimizations. It is also recommended to avoid multiple grids as they negatively influence the numerical stability (use 3 instead of m3). To perform a two-component TDDFT calculation, the two-component version of `ridft` has to be run before (see Chapter 6.4) using the keywords `$soghf` and `$kramers`.

8.4.2 Polarizabilities and Optical Rotations

The calculation of dynamic polarizabilities is controlled by the keyword

`$scfinstab dynpol` *unit*
list of frequencies

unit specifies the unit of the following frequencies and may be `ev`, `nm`, `1/cm`, or `a.u.` (default). The frequencies may be either purely real or purely imaginary. For example, to calculate dynamic polarizabilities at 590 nm and 400i nm (i is the imaginary unit), specify

```
$scfinstab dynpol nm
  590
  400 i
```

and run `escf`,

```
escf > escf.out &    .
```

The resulting polarizabilities and rotatory dispersions are given in a.u. in the program output (`escf.out` in the above example).

The conversion of the optical rotation in a.u. to the specific rotation $[\alpha]_\omega$ in deg·[dm·(g/cc)]$^{-1}$ is given in Eq. (15) of ref. [186].

$$[\alpha]_\omega = C \cdot \delta(\omega) \tag{8.26}$$

where $C = 1.343 \cdot 10^{-4}\omega^2/M$ with M being the the molar mass in g/mol, ω the frequency in cm^{-1}, and $\delta(\omega)$ is 1/3 trace of the electronic rotatory dispersion tensor given in atomic units.
Please note that $\delta(\omega)$ has the wrong sign in older TURBOMOLE versions. It has been corrected in version 6.2.

Note that convergence problems may occur if a frequency is close to an electronic excitation energy. This is a consequence of the (physical) fact that the response diverges at the excitation energies, and not a problem of the algorithm.

Static polarizabilities are calculated most efficiently by specifying

```
$scfinstab polly
```

before starting `escf`. This keyword can be combined with `$soghf` and the other relativistic keywords such as `$rdkh`, `$rbss`, `$rx2c`, `$rlocal`, and `$pcc`. Please see Sec. 6.4 for details.

8.4.3 Damped Response Calculations

A damped response calculation (adding a complex constant to a polarizability) can be invoked by additionally adding the following keywords to the control file:

```
$scfinstab dynpol nm
  590
$damped_response 0.25 eV
```

The last keyword distinguishes it from a simple polarizability calculation described above. Defined units for the frequency and the damping factor may be different. Valid units are eV, cm-1 and nm; if nothing is specified atomic units are assumed. Further it is possible to specify an evenly spaced list of frequencies:

```
$scfinstab dynpol eV
  2.0 3.0 10
$damped_response 0.25 eV
```

This will perform a damped response calculation of 10 frequencies between 2.0 eV and 3.0 eV with a complex damping factor of 0.25 eV. Damped response polarizabilities can be performed for 1c (open and closed shell) and 2c Kramers-symmetric quasi-relativistic references (ECPs or relativistic all-electron approaches) within the time-dependent DFT (including local hybrid functionals) and also using the Bethe-Salpeter equation. The latter is especially recommended if core states are targeted. Note that in the case of GW-BSE damped response calculations the targeted core states should be included in the orbital range when the GW quasiparticle energies are evaluated. A picture-change correction of the dipole moment is available for relativistic all-electron Hamiltonians (`$rdkh`, `$rbss`, `$rx2c`, also in their local variant `$rlocal`)and enforced by `$pcc`. For details on relativistic effects please, see Sec. 6.4.

8.4.4 Dynamic First Hyperpolarizability

Hyperpolarizability calculations are run through `escf` and can be requested by including

```
$scfinstab hyperpol <unit>
<freq1>
<freq2>
<pair1_a> <pair2_b>
...
```

in the control file. A hyperpolarizability calculation will be performed using the pairs given on each line AND using all combinations of the single frequencies given alone on each line. Specifying no frequencies initiates a static calculation. The same unit specifications are accepted as for dynamic polarizability calculations. For example,

```
$scfinstab hyperpol nm
800
1000
1064 1064
```

will compute hyperpolarizability tensors with frequency pairs

- 1000 nm, 1000 nm
- 1000 nm, 800 nm
- 800 nm, 800 nm
- 1064 nm, 1064 nm

8.4.5 Stability Analysis

Stability analysis of spin-restricted closed-shell ground states is enabled by

`$scfinstab singlet`
: for singlet instabilities,

`$scfinstab triplet`
: for triplet instabilities (most common), and

`$scfinstab non-real`
: for non-real instabilities.

`$scfinstab complex`
: for general complex instabilities (2c Kramers symmetric reference).

After that, it is necessary to specify the IRREPs of the electronic Hessian eigenvectors ("orbital rotations") to be considered. Without additional knowledge of the system one usually needs to calculate the lowest eigenvalue within every IRREP:

```
$soes all 1
```

Positivity of the lowest eigenvalues in all IRREPs is sufficient for stability of the ground state solution. If one is interested in, say, the lowest eigenvalues in IRREPs e_g and t_{2g} only, one may specify:

```
$soes
eg   1
t2g  1
```

Triplet instabilities in the totally symmetric IRREP indicate open shell diradical states (singlet or triplet). In this case, start MOs for spin-symmetry broken UHF or UKS ground state calculation can be generated by specifying

```
$start vector generation
```

`escf` will provide the start MOs (→ `$uhfmo_alpha`, `$uhfmo_beta`) as well as occupation numbers (→ `$alpha shells`, `$beta shells`) for a spin-unrestricted calculation with equal numbers of α and β electrons (pseudo-singlet occupation).

8.4.6 Vertical Excitation and CD Spectra

The calculation of excited states within the TDHF(RPA)/TDDFT approach is enabled by

`$scfinstab rpas`
: for closed-shell singlet excitations,

`$scfinstab rpat`
: for closed-shell triplet excitations, and

`$scfinstab urpa`
: for excitations out of spin-unrestricted reference states.

If it is intended to use the TDA instead, specify

`$scfinstab ciss`
: for closed-shell singlet excitations,

`$scfinstab cist`
: for closed-shell triplet excitations,

`$scfinstab ucis`
: for excitations out of spin-unrestricted reference states, and

`$scfinstab spinflip`
: for spin-flip (z-component of the total spin changes by ± 1) excitations out of spin-unrestricted reference states. For details concerning the theory see ref. [197]. In practice, this functionality can be used for the calculation of triplet-singlet, quartet-doublet, ... excitations (see ref. [198] also for further information about the implementation). It is only available within the TDA in combination with LDA functionals and the HF exchange. It is strongly recommended to increase `$escfiterlimit`.

In the two-component case, specify

`$scfinstab soghf`
: for two-component excitation energy calculations on closed-shell systems. [27] This implementation is only available in combination with LDA and GGA functionals; since version 7.4 also hybrid functionals are supported [29].

`$scfinstab tdasoghf`
: for two-component excitation energy calculations on closed-shell systems using the TDA, where in addition HF exchange is accessible. [28]

```
open-shell systems can be accessed using TD-HF or the Bethe-Salpeter equation $bse
```

The keywords `$soghf` and `$kramers` in case of closed-shell systems also have to be set. Note that in two-component TD-DFT with metaGGAs and up, the current-dependent response is

set as default. In two-component local hybrid functional TD-DFT calculations, the mixing between the exact exchange energy density and the LMF is neglected.

The Bethe-Salpeter equation (BSE) for excited states can be invoked by adding the keyword `$bse`. The correlation-augmented Bethe-Salpeter equation (cBSE) for excited states can be invoked by adding the keyword `$cbse` [30]. In BSE calculation RI-K is mandatory and automatically set.

Next, the IRREPs of the excitations need to be defined, which is again accomplished using `$soes`. For example, to calculate the 17 lowest excitations in IRREP b_{1g}, the 23 lowest excitations in IRREP e_u, and all excitations in IRREP t_{2g}, use

```
$soes
b1g  17
eu   23
t2g  all
```

and run `escf`. Since point group symmetry cannot be exploited in two-component calculations, there is only the totally symmetric IRREP a.

Note that `$soes` specifies the IRREP of the *excitation vector* which is not necessarily identical to the IRREP of the *excited state(s)* involved. In general, the IRREP(s) of the excitation(s) from the ground to an excited state is given by the direct product of the IRREPs of the two states. For example, to calculate the first A_2 state in a C_{2v}-symmetric molecule with a B_2 (open-shell) ground state, it is necessary to specify

```
$soes
b1   1
```

The number of excitations that have to be calculated in order to cover a certain spectral range is often difficult to determine in advance. The total number of excitations within each IRREP as provided by the `define ex` menu may give some hint. A good strategy is to start with a smaller number of excitations and, if necessary, perform a second `escf` run on a larger number of states using the already converged excitation vectors as input.

To compute absorption and CD spectra, it is often sufficient to include optically allowed transitions only. This leads to substantial reduction of computational effort for molecules with higher symmetry. For example, in the UV-VIS spectrum of an O_h symmetric molecule, only t_{1u} excitations are optically allowed. The IRREPs of the electric and magnetic dipole moments as well as of the electric quadrupole moment are displayed automatically in the `define ex` menu.

If a large number of states is to be calculated, it is highly recommended to provide extra memory by specifying

`$rpacor` m

the integer m being the core memory size in megabytes (default is 20). The larger m, the more vectors can be processed simultaneously without re-calculation of integrals. As a rule

of thumb, m should be ca. 90% of the available main memory. If RI-J is used (`$ridft`), it is recommended to set `$ricore` to a small value and `$rpacor` to a large value if the number of states is large, and vice versa if it is small. Since two-component calculations are more demanding concerning computation time and required memory it is strongly recommended to increase `$rpacor`.

By specifying

`$spectrum` *unit* and/or

`$cdspectrum` *unit*

a list of excitation energies and oscillator and/or rotatory strengths of the optically allowed transitions is written onto file `spectrum` and/or `cdspectrum`. As above, unit specifies the energy unit and may be `ev`, `nm`, `1/cm`, or `a.u.` (default). The files `spectrum` and `cdspectrum` may conveniently be used for further processing, e.g., using a plotting program such as Gnuplot.
Additionally, a spectrum broadened by gaussian functions can be generated by the Peak ANAlyzing MAchine (`panama`). [199] It reads in excitation energies and their corresponding oscillator strengths from the `escf` output file and prints the resulting spectrum to `data.plot` (on an eV axis) or to `data.nm.plot` (on a nanometer axis) which can be plotted by programs such as Gnuplot.

Both differential densities and non-relaxed difference densities can be visualized to get an impression of the character and localization of the excitation(s). Differential densities (`ed.plt`) are generated by `egrad` after setting the keyword `$pointval` in combination with the `-proper` option (see Sec. 20.2). A computational less demanding alternative are non-relaxed difference densities which are directly obtained from the `escf` output file by first running `panama` and subsequently `dscf -proper` or `ridft -proper`. [199]

Exact TDDFT transition density can also be plotted by the `escf` program, see section 20.2.

By specifying

`$curswitchdisengage`

inclusion of the current-density response for MGGA calculations is disabled. *Note that the results of calculations using this flag will no longer be gauge-invariant and will differ from results obtained with the standard gauge-invariant implementation.*

In general, CD spectra calculated with the length representation of the electric transition dipole moment are not gauge-invariant. Using GIAOs for the magnetic transition dipole moment, however, makes CD spectra gauge-invariant even if the length representation is used for the electric transition dipole moment. Calculating gauge-invariant rotatory strength tensors and thus CD spectra is possible for closed-shell molecules if the keyword `$mgiao` is added. First, an `mpshift` calculation needs to be run in order to obtain the perturbed density

which is written on disk in a filed called 'umunu'. Then, a regular `escf` calculation can be run in order to calculate magnetic transition dipole moments which lead to gauge-invariant rotatory strength tensors for the length representation.

8.4.7 Two-photon absorption

2PA calculations are run through `escf` and can be requested with

```
$scfinstab twophoton <excited state method>
<freq1>
...
$soes
...
$exopt
...
```

where `<excited state method>` should be a description of the excited state method (`rpas`, `ciss`, `urpa`, `ucis`). A 2PA tensor will be computed for each excited state specified by the `$soes` block using the frequencies `<freq1>` and Ω_n−`<freq1>`. Also, `half` may be provided as a frequency, which then uses half the excitation energy for each frequency (this is the default). 2PA amplitudes for specific states can be computed by specifying the states in `$exopt`. `$exopt` for different irreps can be specified just as for `$soes` or also as a comma separated list of indices (e.g., "3") and ranges (e.g. "5-7"). For example

```
$scfinstab twophoton rpas
$soes
a1 6
a2 8
$exopt
a1 1-3, 5
a2 all
```

will compute

- 6 excited states in the a_1 IRREP and 8 excited states in a_2 and
- 2PA amplitudes with frequencies equal to $\Omega_n/2$ for states 1, 2, 3, and 5 in irrep a_1, and all 8 computed states in a_2.

2PA amplitudes require the calculation of dynamic polarization vectors $|X^{(\alpha)}, Y^{(\alpha)}\rangle$ at the frequencies specified (e.g., $\Omega_n/2$). If 2PA amplitudes are requested for many states, these frequencies can become (nearly) resonant with lower-lying excitation energies, which can cause severe convergence problems. If such problems are encountered, try limiting the number of 2PA amplitudes computed in a single pass using `$exopt`.

8.4.8 Excited State Geometry Optimizations

The input for computing excited state gradients and properties using `egrad` is exactly the same as for an excited state calculation using `escf`, see the previous section. Gradients and properties are calculated only for one state at a time. By default, this is the highest excitation specified by `$soes` (only one IRREP is allowed). Sometimes, e.g. close to excited state intersections, it may be necessary to include higher excited states in the initial excitation vector calculation to prevent root flipping. This is accomplished using

`$exopt` n

which explicitly enforces treatment of the n-th state; n must be less or equal the number of states specified in `$soes`.

After the input for the ground and excited state calculations has been set up, an excited state geometry optimization can be started by issuing the command

```
nohup jobex -ex &
```

The option `-ex` forces `jobex` to call `egrad` instead of `grad` (or `rdgrad` if `-ri` is also specified). In each geometry step, the excitation energy is written on the fourth column in `$energy`, and the data group `$last excitation energy change` is updated. Otherwise, the excited state optimization proceeds in exactly the same way as a ground state optimization (see Chapter 3).

8.4.9 Excited State Force Constant Calculations

Excited state vibrational frequencies can be calculated by numerical differentiation of analytic gradients using `NumForce` (see Chapter 15). A `NumForce` calculation for an excited state may be started by the command

`nohup NumForce -ex` n `> force.out &`

where n is the number of the excited state *in* C_1 *symmetry*. In order to determine n, it is recommended to perform an `escf` calculation in C_1 symmetry. Note that numerical calculation of excited state force constants *is likely to fail* if there are other states nearby (in C_1), because the roots may flip when the molecule is distorted. Note also that it may be necessary to include higher excited states (using `$exopt`, see above) in C_1 calculations of molecules with higher symmetry in order to enforce convergence to the correct state. In any case, it should be checked that the energy change due to the displacements (available in the `numforce/KraftWerk/*.log` files) is reasonably small.

For a `NumForce` run, the convergence criteria should be tightened. It is recommended to use at least

```
$scfconv 8
```

in all `NumForce` calculations. Other `NumForce` options such as `-central`, `-d`, `-np` work in exactly the same way as they do for ground states.

8.4.10 Polarizability Derivatives and Raman Spectra

Calculations of polarizability derivatives by the `egrad` program use the same specifications in the `$scfinstab` data group as polarizability calculations by `escf`.

```
$scfinstab polly
```

specifies derivatives of the static polarizability, while

`$scfinstab dynpol` *unit*
frequency

requests derivatives of the dynamical polarizability at the given frequency. Note that, unlike polarizability calculations, multiple frequencies are not allowed. Polarizability derivatives have to be projected onto vibrational normal modes to obtain Raman intensities, see Chapter 15 for further details.

8.4.11 State-to-state properties

All state-to-state properties, including transition moments and derivative couplings, are computed using the state-to-state derivative coupling machinery in `egrad`, as the state-to-state 1TDM is a byproduct of computing the derivative coupling. Only C_1 symmetry is supported.

To trigger the calculation of state-to-state derivative couplings, an option must be given to the `$nacme` keyword.

```
$nacme <full/pseudo or response>
```

Providing either `full` or `pseudo` will compute derivative couplings under the pseudowavefunction approximation. This is the recommended option since pseudowavefunction couplings are well-behaved and stable. Providing `response` will compute derivative couplings with quadratic response theory. Couplings computed within response theory can diverge unphysically, so caution is advised.

For derivative couplings, it is also recommended to neglect the antisymmetric overlap integrals which are responsible for translational variance. This is approximately equivalent to incorporating electron translation factors (ETF) and is enabled by placing

```
$do_etf
```

in the control file.

The `$coupled states` keyword will control which states are coupled. `$coupled states` understands a comma separated list of numbers and ranges. Couplings will be computed between every pair of states specified on `$coupled states`. By default, `$coupled states` has the value "all", which computes couplings between every state specified in `$soes`. For example,

```
$coupled states 1-3, 6, 8
```

will compute couplings between each pair of states in the list 1, 2, 3, 6, 8.

```
$coupled states all
```

will compute couplings between all available states.

When using `$nacme` is found, the `$exopt` key has added flexibility that allows the simultaneous calculation of multiple excited state gradients. It understands the same syntax as `$coupled states` such that

```
$coupled states 1-3, 6, 8
```

will compute gradients corresponding to states 1, 2, 3, 6, 8.

```
$coupled states all
```

will likewise compute gradients for all available states.

8.4.12 Nuclear spin-spin coupling constants

All four terms of Ramsey's theory [200] (Fermi-contact, spin–dipole as well as paramagnetic and diamagnetic spin–orbit contributions, abbreviated FC, SD, PSO, and DSO, respectively) can be calculated. The FC/SD cross contributions are included in the full tensor [123].

The calculation is triggered by the keyword `$ncoupling` in the control file (no `$scfinstab` is needed). This keyword along with several options can be set in the last menu in `define`. To get reliable results, a basis set with steep s functions (for example Jensen's pcJ-n basis sets [201,202]) is needed. Starting with Version 7.6, the kinetic energy density is generalized with the paramagnetic current density by default [125]. This affects only the paramagnetic spin–orbit term. The generalization can be disabled by `$curswitchdisengage`.

Relativistic effects can be treated in a one-component (scalar X2C) and a two-component (spin-orbit X2C) formalism. For the scalar-relativistic approach [192], the keyword `$rx2c` is required. Further, `$rlocal` is strongly recommended. Note that the FC and SD terms are coupled and consequently the FC/SD cross contribution is not explicitly evaluated. In contrast, the PSO and DSO terms can be calculated separately. The restricted kinetic balance (RKB) condition and a finite nucleus model (if `$finnuc` is set) for both the scalar and the vector potential are employed. In a finite nucleus model, steep s functions are not

that important and the use of x2c-TZVPall [155] or Dyall's uncontracted CVTZ [203–205] basis sets is usually sufficient. Note that the one-component approach leads to comparably large errors when the DSO term becomes pronounced, e.g., for H–H couplings. Here, complex algebra can be used to consider all X2C contributions. This is done by using the pseudo-two-component algorithm (dsop2c). This does not require a two-component SCF solution [192]. Simply replace 'dso' in the `$ncoupling` group of the control file by 'dsop2c'.'

In a relativistic two-component calculation [193], the desired 2c keywords (e.g. `$soghf`, `$rx2c`, `$rlocal`, `$snso`, and `$snsopara`) have to be set in addition to `$ncoupling`. The calculation of spin–spin coupling constants is currently restricted to closed-shell systems. Thus, the keyword `$kramers` should be set for the ground-state calculation. The implementation supports all density functional approximations of the first four rungs of Jacob's ladder [193]. The current-dependent generalization of the response of the kinetic energy density is used by default to ensure gauge invariance [125, 193, 194]. As done for the scalar X2C approach, the restricted kinetic balance (RKB) condition and a finite nucleus model (if `$finnuc` is set) for both the scalar and the vector potential are employed. The FC, SD, and PSO terms of Ramsey's theory are coupled and can no longer be calculated individually. Due to the X2C response, DSO-like term is recovered (despite using RKB), but which is expensive and can therefore be turned off. Using the picture-change correction (`$pcc`) in a manner similar to Refs. 37 and 206 is a cheap alternative [207]. The use of x2c-TZVPall-2c [155] or Dyall's uncontracted CVTZ [203, 204] basis sets is recommended.

The GW-BSE formalism is available for both non-relativistic, scalar-relativistic, and relativistic two-component calculations [194]. This only requires GW quasiparticle energies and the keyword `$bse` in addition to the SSCC settings. Note that the quasiparticle energies can be obtained as discussed in Sec. 14 and have to be stored on disk.

The gyromagnetic ratios of the coupling nuclei can easily be changed in the atomic attributes menu of the `define` module. The same holds for the isotopes, see also Sec. 23.2.1.

The output will include the isotropic part of the coupling (cf. Ref. 123),

$$J_{KL}^{\mathrm{iso}} = \frac{1}{3} \sum_{\alpha=x,y,z} (J_{KL})_{\alpha\alpha} \,, \tag{8.27}$$

the anisotropic contribution,

$$J_{KL}^{\mathrm{anis}} = h \frac{\gamma_K}{2\pi} \frac{\gamma_L}{2\pi} \sqrt{\frac{3}{2} \left(\frac{1}{4} \sum_{\alpha\beta} \left((K_{KL})_{\alpha\beta} + (K_{KL})_{\beta\alpha} \right)^2 - 3(K_{KL}^{\mathrm{iso}})^2 \right)} \,, \tag{8.28}$$

and the complete tensor J_{KL} according to 8.25. You can also choose to obtain the corresponding quantities of the reduced coupling tensor K.

8.4.13 Magnetic fields

TD-DFT and GW-BSE calculations in magnetic fields can be performed since V7.7. A two-component formalism is necessary for this, so `$soghf` needs to be set for magnetic fields.

Note that for metaGGAs the full current-dependent kernel is used, effectively leading to current-dependent cTD-DFT.

8.4.14 Prediciting colors using the Color Prediction Tool cpt

To obtain the calculated emission and absorption colors of a compound after a TD-DFT/GW-BSE calculation simply execute `cpt` in the same directory (basically only the exspectrum or spectrum file needs to be present). In case of response calculations from `ricc2` or `pnoccsd` the `$spectrum` keywords needs to be present for the spectrum file to be generated. `escf`, starting from version 7.4, will always generate exspectrum files that can automatically be read in from `cpt`. Note: if more than one spectrum is present in the exspectrum/spectrum file `cpt` will generate the accumulated color. For some examples see test case TURBOTEST/escf/short/CPT and the included files and CRIT file.

8.4.15 Approximations for Coulomb and Exchange integrals

In hybrid functional TD-DFT (gradient) calculations most of the time is spent in evaluating the exchange-integrals. Therefore two ways to speed these up have been implemented:

`$rick`
: use the RI-K approximation (recommended if enough RAM available)

`$senex`
: use seminumerical exchange. The default settings of `$esenex` are recommended and lead to negligible errors in the excitation energies.

Usage of RI-K is highly recommended if enough RAM and a fast disk can be provided. RI-K is available for all types of calculations in the modules `escf` and `egrad` up to D2h symmetry. The keyword `$rick` is exclusive to `escf`, `egrad` and `aoforce`, other programs will not trigger usage of RI-K even if those parts will. The keyword `$rik` triggers RI-K in all modules where it is available, also (but not only) in `escf`, `egrad` and `aoforce`. RI-K is not available for range-separated hybrids. Seminumerical exchange is supported for all types of calculations in `escf` and `egrad`, also two-component calculations are fully supported. We recommend to use the default settings as they will yield reliable results in virtually all cases.

```
$esenex
```

Further, if one also wants to compute the Coulomb contribution using seminumerical techniques the `$pseudospectral` keyword may be added to the control file.

```
$esenex
$pseudospectral
```

Seminumerical exchange and fully pseudospectral approaches can be used with all point groups and functionals (global and range-separated hybrids). For a more detailed list of options please refer to the general keyword section.

8.4.16 Minimal auxiliary basis set

The calculation of TDDFT response properties can be speeded-up by about two-orders of magnitude using a minimal auxiliary basis set composed by just one s-type basis function per atom [208, 209] and switching off the calculation of the XC kernel on the grid. The resulting approach is named TDDFT-as [208], for semilocal XC functionals using the RI-J approach with a minimal auxiliary basis set for the Coulomb interaction, or TDDFT-ris [209], for hybrid XC functionals using the RI-K approach with the same minimal auxiliary basis set for both the Coulomb and exchange interaction. The resulting average deviation from conventional TDDFT excitation energies is less than 0.1 eV for organic molecules [209] and less than 0.02 eV metal clusters [208]. Note that we are considering only singlet excitation energies for closed-shell systems: open-shell systems and triplet excitations are inaccurate.

The TDDFT-as/TDDFT-ris approaches are based on the exact KS ground-states and approximate the linear-response. Thus these methods are similar to other tight-binding TDDFT methods [210–212] and less empirical. The exponents of the minimal auxiliary basis set need to be properly selected as they also approximate the XC kernel effects.

The keyword

```
$escfnoxc
```

disables the grid and the calculation of the XC kernel.

The script `escfrisprep` can be use to add this keyword to the `control` file as well as to include the minimal auxiliary basis set.

How to perform a TDDFT-as/TDDFT-ris:

- Run a conventional ground-state KS with RI-J or RI-JK approximation
- Use `define` to add all the required TDDFT keywords in the control file.
- Run `escfrisprep`
- Run `escf`

`escfrisprep`, with the default option (`-m auto`) will detect the XC type (reading the `$xctype` group) and define the appropriate basis set. Otherwise, for hybrid/RSH functionals, the TDDFT-ris method can be forced with:

```
escfrisprep -m ris
```

and the minimal basis set will be set in the `$cbas` group. For semilocal functionals, the TDDFT-as method can be forced with:

```
escfrisprep -m as
```

and the minimal basis set will be set in the `$jbas` group.

For transition metal-complexes, the minimal auxiliary basis set is not sufficient for the description of low-lying excited states [209]. For those systems run:

```
escfrisprep -x element
```

where `element` is the element (e.g. the transition metal atom) to exclude. Use -x multiple times if you want to exclude more than one element (e.g `-x Au -x Ag`). For those atoms the full (default) auxiliary basis set will be used.

TDDFT-as/TDDFT-ris can also be used to generate initial guesses for further runs with standard TDDFT, which can reduce the number of iterations in some cases by a factor of 1.5. To do this, set up your calculation as previously described. Then run:

```
escfrisprep
escf > escf-ris.out
escfrisprep -r
escf > escf.out
```

The command `escfrisprep -r` will restore the standard TDDFT files.

Chapter 9

Second-order Møller–Plesset Perturbation Theory

Preliminary note

TURBOMOLE offers three programs for MP2 energy and gradient calculations. A "conventional" implementation [213], mpgrad, based on the calculation of four-center integrals, an implementation which uses the resolution-of-the-identity (RI) approximation [214] as part the RI-CC2 program [11], ricc2, and an implementation which is meant for very large ($>$ 100 atoms and $>$ 3000 basis functions) single point MP2 energy calculations. The latter program uses in addition to a local RI also a hybrid OSV-PNO approximation to reduce the scaling of the computational costs with the system size to $\approx \mathcal{O}(\mathcal{N}^2)$ but has for small and medium-sized systems a larger prefactor than ricc2.

9.1 Functionalities of MPGRAD, and RICC2, and PNOCCSD

Functionalities of mpgrad:

- Calculation of MP2 energies and/or MP2 gradients for RHF and UHF wave functions.
- The frozen core approximation (possibility to exclude low-lying orbitals from the MP2 treatment) is implemented only for MP2 energies.
- Exploitation of symmetry of all point groups.
- Can be used sequentially or MPI-parallel.
- Can be combined with the COSMO solvation model (see section 9.7 and chapter 19.2 for details). (Presently restricted to sequential calculations.)

Functionalities of ricc2 at the MP2 level:

- Calculation of MP2 energies and/or gradients for RHF and UHF wave functions within the RI approximation (RI-MP2). In geometry optimizations and vibrational frequency calculations (with NumForce) it can be combined with RI-JK-SCF for the Hartree-Fock reference calculation.
- The frozen core approximation is implemented for both energies and gradients.
- RI-MP2 needs optimised auxiliary basis sets, which are available for most standard basis sets as e.g. SVP, TZVP, TZVPP, QZVPP as well as for the (aug-)cc-p(wC)VXZ (X = D, T, Q, 5) basis set series (for Al–Ar also for the (aug-)cc-p(wC)V(X+d)Z series and for *p*-block elements Ga–Rn also the respective ECP basis set series (-pp)).
- Exploitation of symmetry for all point groups for MP2 energies and gradients.
- Can be combined with the COSMO solvation model (see chapter 19.2 for details).
- Runs sequentially and parallel (with MPI, OpenMP and hybrid MPI/OpenMP)
- Contains an implementation of explicitly correlated MP2-F12 methods (presently restricted to energies and the C_1 point group).
- Can for open-shell calculations be used with UHF and single-determinant high-spin ROHF reference wavefunctions. (ROHF-MP2 presently limited to energies.)
- Energies and gradients for the spin-component scaled SCS- and SOS-MP2 approaches, including a Laplace-transformed implementation of SOS-MP2 with $\mathcal{O}(\mathcal{N}^4)$ scaling computational costs.
- Static polarizabilities (currently restricted to closed-shell reference wavefunctions and the sequential and SMP versions; cannot yet be combined with spin-component scaling), see Chapter 10.5 for a description of the input
- See Chapter 10 for further details.

Functionalities of `pnoccsd`:

- Currently restricted to CCSD, CCSD(T0), CCSD(T), MP2 and DFT double hybrid (e.g. B2PLYP) single point energy calculations with a RHF or UHF reference determinant and the C_1 point group.
- Runs sequentially and parallel (MP2 with MPI, OpenMP and hybrid MPI/OpenMP, CCSD and beyond with OpenMP).
- Contains an implementation of explicitly correlated PNO-MP2-F12 methods.
- See Section 12 for further details.

9.1.1 How to quote

- For calculations with `mpgrad`:
  Semi-direct MP2 Gradient Evaluation on Workstation Computers: The MPGRAD Program. F. Haase and R. Ahlrichs; J. Comp. Chem. **14**, 907 (1993).

- For calculations with `ricc2`:
  CC2 excitation energy calculations on large molecules using the resolution of the identity approximation. C. Hättig and F. Weigend;
- for MPI parallel calculations with `ricc2` in addition:
  Distributed memory parallel implementation of energies and gradients for second-order Møller-Plesset perturbation theory with the resolution-of-the-identity approximation. Christof Hättig, Arnim Hellweg, Andreas Köhn, Phys. Chem. Chem. Phys. **8**, 1159-1169, (2006).
- for MP2-F12 calculations in addition:
  The MP2-F12 Method in the TURBOMOLE Programm Package. Rafal A. Bachorz, Florian A. Bischoff, Andreas Glöß, Christof Hättig, Sebastian Höfener, Wim Klopper, David P. Tew, J. Comput. Chem. **32**, 2492–2513 (2011).
- for $\mathcal{O}(\mathcal{N}^4)$-scaling LT-SOS-MP2 calculations:
  Scaled opposite-spin CC2 for ground and excited states with fourth order scaling computational costs. Nina O. C. Winter, Christof Hättig, J. Chem. Phys., **134**, 184101 (2011) and Scaled opposite-spin second order Møller–Plesset correlation energy: An economical electronic structure method. Y., Jung, R.C. Lochan, A.D. Dutoi, and M. Head-Gordon, J. Chem. Phys., **121**, 9793 (2004).
- for SCS-MP2 calculations:
  S. Grimme, J. Chem. Phys. **118**, 9095 (2003).
- for RI-MP2 polarizabilities:
  Large scale polarizability calculations using the approximate coupled cluster model CC2 and MP2 combined with the resolution-of-the identity approximation. Daniel H. Friese, Nina O. C. Winter, Patrick Balzerowski, Raffael Schwan, Christof Hättig, J. Chem. Phys., **136**, 174106 (2012).
- for PNO-MP2 calculations: A $\mathcal{O}(\mathcal{N}^3)$-scaling PNO-MP2 method using a hybrid OSV-PNO approach with an iterative direct generation of OSVs. Gunnar Schmitz, Benjamin Helmich, Christof Hättig, Mol. Phys. **111**, 2463–2476, (2013). and Principal Domains in Local Correlation Theory. David. P. Tew, J. Chem. Theory Comput. **15**, 6597 (2019)
- for explicitly correlated PNO-MP2-F12 calculations: Explicitly correlated PNO-MP2 and PNO-CCSD and its application to the S66 set and large molecular systems. Gunnar Schmitz, Christof Hättig, David Tew, Phys. Chem. Chem. Phys. **16**, 22167–22178 (2014).

9.2 Some Theory

Second-order Møller–Plesset Perturbation Theory (MP2) corrects errors introduced by the mean-field ansatz of the Hartree–Fock (HF) theory. The perturbation operator is the difference the full electronic Hamiltonian and the Fock operator for occupied/occupied and

virtual/virtual block. The MP2 energy is (in spin orbitals) given by:

$$E_{\mathrm{MP2}} = \frac{1}{4}\sum_{iajb} t_{ab}^{ij}\langle ij||ab\rangle, \tag{9.1}$$

with the doubles amplitudes

$$t_{ab}^{ij} = \frac{\langle ij||ab\rangle}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}, \tag{9.2}$$

i and j denote occupied, a and b virtual orbitals, ϵ_p are the corresponding orbital energies, and $\langle ij||ab\rangle = \langle ij|ab\rangle - \langle ij|ba\rangle$ are four-center two-electron repulsion integrals.

MP2 gradients (necessary for optimisation of structure parameters) are calculated as analytical derivatives of the MP2 energy with respect to nuclear coordinates. Calulation of these derivatives also yields the first order perturbed wave function, expressed as "MP2 density matrix", in analogy to the HF density matrix. MP2 corrections to first-order properties like electric moments or atomic populations are obtained from the density matrix in the same way as for Hartree-Fock.

The "resolution of the identity (RI) approximation" means expansion of products of virtual and occupied orbitals in a basis of auxiliary functions. The calculation and transformation of the four-center two-electron integrals (see above) is replaced by that of three-center integrals, which leads to computational savings of RI-MP2 compared to a conventional MP2 calculations by a factor of ca. 5 (small basis sets like SVP) to ca. 10 (large basis sets like TZVPP) and more (for quadruple-ζ and larger basis sets). The RI errors—i.e. the errors due to the RI approximation—are with optimised auxiliary basis sets small and well documented [215, 216]. The use of the `mpgrad` program is therefore only recommended for reference calculations or if suitable auxiliary basis sets are not available.

The PNO-MP2 implementation in the `pnoccsd` program is meant for large systems with $\gtrapprox$ 100 atoms where the costs for RI-MP2 calculations become unreasonably large. The PNO-MP2 implementation uses five additional approximations to reduce the scaling of the computational costs with the system size to below $\mathcal{O}(\mathcal{N}^2)$:

- The truncation of the pair natural orbital (PNO) basis, $\phi_{\bar{a}_{ij}} = \sum_a \phi_a d_{a\bar{a}}^{ij}$ for each pair ij of occupied orbitals to PNOs with occupation numbers $n_{\bar{a}}^{ij} \geq T_{\mathrm{PNO}}$:

$$\sum_b D_{ab}^{ij} d_{b\bar{b}}^{ij} = n_{\bar{b}}^{ij} d_{a\bar{b}}^{ij} \tag{9.3}$$

- The truncation of the orbital specific virtual (OSV) basis, $\phi_{\tilde{a}_i} = \sum_a \phi_a d_{a\tilde{a}}^{i}$ for each occupied orbital i to OSVs with occupation numbers $n_{\tilde{a}}^{i} \geq T_{\mathrm{OSV}}$:

$$\sum_b D_{ab}^{ii} d_{b\tilde{b}}^{i} = n_{\tilde{b}}^{i} d_{a\tilde{b}}^{i} \tag{9.4}$$

  The OSV basis is used to expand the PNOs. For a pair ij the PNOs are expanded in the union of the OSVs for the occupied orbitals i and j.

- The distant pair approximation: only such pairs ij are included in the correlation treatment for which the contribution to the correlation energy is expected (based on initially computed estimates) to be $\geq T_{\mathrm{pair}}$.

- The local RI approximation: two-electron integrals for a pair ij are expanded in a pair-specific subset of the full auxiliary basis which is determined based on an overlap criterium and the selection threshold T_{RI}.
- The principal domain PAO approximation: the OSVs and PNOs are represented using a domain of projected atomic orbitals which are selected to best represent the density.

9.3 How to Prepare and Perform MP2 Calculations

Prerequisites

Calculations with `mpgrad`, `ricc2` or `pnoccsd` require

- a converged SCF calculation with the one-electron density convergence threshold set to `$denconv 1.d-7` or less
- the maximum core memory the program is allowed to allocate should be defined in the data group `$maxcor` (in MB); the recommended value is ca. 3/4 of the available (physical) core memory at most.
- orbitals to be excluded from the correlation treatment have to be specified in the data group `$freeze`
- for `mpgrad` the calculation of gradients is omitted by adding the flag `$mp2energy` to the `control` file; in this case only the MP2 energy is calculated.

Calculations with `ricc2` and `pnoccsd` moreover use

- an auxiliary basis defined in the data groups `$atoms` and `$cbas`. If not specified before the start of `ricc2` or `pnoccsd` the programs will assign the default auxiliary basis sets which have the same names as the one-electron basis sets and will try to extract them from the TURBOMOLE basis set library. If this succeeds the assignment and auxiliary basis sets are stored in the `control` file else the programs will stop.

This is not needed for `mpgrad`, but here one needs

- a specification for scratch files and their size in data group `$mointunit` (see Section 23.2.21)
- and the number of passes for integral evaluations and transformations in data group `$traloop`

For explicitly-correlated MP2-F12 calculations one needs—depending the details of the applied approximations—additionally a so-called complementary auxiliary basis set (CABS, defined in `$cabs`) and a RI-SCF auxiliary basis set defined in `$jkbas`.

Calculations with ricc2 and pnoccsd

1. RI-MP2 calculations require the specification of auxiliary basis sets (`$cbas`) and a converged SCF calculation with the one-electron density convergence threshold set to `$denconv 1.d-7` or less. In addition, the options `$freeze` (frozen core approximation) and `$maxcor` (maximum core memory usage) should be set. All these settings can be done during the input generation with the program `define` under the entries `mp2`/`cc` or `pnocc` of the last main menu.

2. The `ricc2` program requires the data group:
   ```
   $ricc2
     mp2
     geoopt model=mp2
   ```
   (Where the last line should only be included if the calculation of gradients is needed e.g. in geometry optimizations.) This can be prepared with `define` in the menu `cc`.

3. The `pnoccsd` program does not require any special input for a MP2 calculation with default thresholds but it is recommended to specify the PNO truncation threshold explicitly in the data group:
   ```
   $pnoccsd
     mp2
     tolpno= 1.0E-7
   ```
   This can be prepared with `define` in the menu `pnocc`.

4. For explicitly-correlated MP2-F12 calculations with `ricc2` or `pnoccsd` also the data groups `$rir12` and `$lcg` are needed.

5. Start a single `ricc2` and `pnoccsd` calculations respectively with the commands `ricc2` and `pnoccsd`.

6. For optimisation of structure parameters at the RI-MP2 level with `ricc2` use the command `jobex -level cc2`. For geometry optimizations with RI-JK-SCF as reference for RI-MP2 with the `ridft` and `ricc2` binaries the additional option `-rijk` has to be given.

7. The combination of RI-MP2 with RI-JK-SCF can lead to significant computational savings in particular for geometry optimizations for small and medium sized molecules with large basis sets (quadruple-ζ and beyond) or basis sets with diffuse functions (e.g. the aug-cc-pVXZ basis set families). For large molecules with TZVPP or similar basis sets, conventional direct SCF calculations are usually more efficient.

8. With `ricc2` and `pnoccsd` spin-component scaled SCS- or SOS-RI-MP2 calculations can be carried out by adding in the `$ricc2` (for `ricc2`) or `$pnoccsd` (for `pnoccsd` data group the line

   ```
   scs  cos=1.2d0 css=0.3333d0
   ```

   where the two parameters are the scaling factors for, respectively, the opposite- and same-spin contribution. The specification of the scaling factors is optional; the default

values are cos=6/5 and css=1/3 as recommended by S. Grimme in *J. Chem. Phys.* **118** (2003) 9095. The abbreviation `sos` can be used for SOS-MP2 calculations with cos=1.3 and css=0.0 (Y., Jung, R.C. Lochan, A.D. Dutoi, and M. Head-Gordon, *J. Chem. Phys.* **121** (2004) 9793.). SOS-MP2 is in `ricc2` implemented with $\mathcal{O}(\mathcal{N}^4)$ scaling costs. For such calculations the data group `$laplace` has to be added.

9. For technical recommendations and additional options for parallel RI-MP2 and PNO-MP2 calculations with the `ricc2` and `pnoccsd` programs see Secs. 3.4 and 10.7 and 12.

MP2 calculations with a ROHF reference state

With the program `ricc2` it is possible to compute MP2 (and spin-component scaled) MP2 energies with single-determinant restricted open-shell reference wavefunctions. No additional input is required apart from the usual ROHF input for the `dscf` and `ridft` programs and a standard MP2 input for `ricc2`.

TURBOMOLEs Hartree-Fock codes can handle within the ROHF framework many cases, which include beside common high- and low-spin configuration state functions also weighted averages of high-spin CSFs (see Sec. 6.3 for further details). The Møller-Plesset perturbation theory and coupled cluster functionalities implemented in `ricc2` require a single-determinant reference state and can thus only deal with high-spin open-shell cases (not averaged):

- The spins of all n_{open} unpaired electrons are parallel (α spin will be assumed) so that the ROHF reference state has the spin multiplicity $n_{open} + 1$.
- There must be only one type of open shells and all orbitals in this shell must have the occupation number 1.
- For the single electron case (i.e. doublets) the Roothaan parameters are $a = b = 0$, for high-spin cases with more than one unpaired electron the Roothaan parameters must be set to $a = 1$ and $b = 2$.

For non-Abelian points groups this implies that shells with degenerate orbitals (as e.g. t_1 in point group I) must be half-filled. An average over the different (symmetry-equivalent or inequivalent) high-spin determinants that are obtained when a shell of degenerate orbitals is less than or more than half-filled is not possible with single-point `ricc2` calculations.

For states with less than or more than half-filled shells of degenerate orbitals the calculations half to be done in a point group that lifts the degeneracy such that it becomes possible to assign integer occupation numbers. A symmetry-breaking of the orbitals can be avoided by doing the Hartree-Fock calculation in the full point group. The input (and MO coefficients) can then be transformed to a lower point group using define only for the `ricc2` calculation.

When using an ROHF reference, two definitions of the frozen core orbitals are possible: The frozen core orbitals can either be spin restricted, or spin unrestricted. In the first case the alpha and beta core orbitals are those on the mos file generated by the ROHF dscf calculation and are the same, but the Fock matrix elements between core and valence orbitals

are non-zero. In the second case, the alpha and beta orbitals semi-canonicalised prior to selecting the frozen orbitals with lowest orbital eigenvalues. Here the frozen alpha and beta orbitals differ (slightly), but the core-valence Fock matrix elements are zero. Both options are available in `ricc2` calculations by specified restricted `res` or (semi-)canonical `can` `$ricc2 core res/can`

Calculations with mpgrad

1. Add `$denconv 1.d-7` to the `control` file and perform a `dscf` run.
2. If any orbitals are decided to be excluded from MP2 treatment, add data group `$freeze` manually to the `control` file, see also Section 23.2.21.
3. For preparation of an `mpgrad` run use the script `Mp2prep`:
   `mp2prep -e/g -m` *memory* `-p` *discspace [scratch file directory]*
   As an example, with the command
   `mp2prep -e -m 100 -p 1000 /work`
   an MP2-energy calculation is prepared, the amount of available core memory is restricted to 100 MB, the MOs are blocked, so that integral scratch files—located in the directory /work—do not need more than 1000 Mb. The number of blocks, i.e. the number of passes with repeated integral evaluations, is written to the `control` file (`$traloop`) as well as the specification of scratch files (`$mointunit`, see Section 23.2.21). Note: less disc space means more passes and thus lower efficiency of `mpgrad`, but due the technical limitations *discspace* should be limited to values $<$ 16Gb to avoid integer overflow errors. Settings obtained by `mp2prep` may be changed manually. You may change the number of passes in `$traloop` by editing the `control` file (e.g. if the originally intended disc space is not available). To adapt the size of scratch files add `$statistics mpgrad` to `control` file and start an `mpgrad` statistics run with the command `mpgrad`.
4. Start a single `mpgrad` calculation with the command `mpgrad`.
5. For optimisation of structure parameters at the (non-RI-) MP2 level use the command `jobex -level mp2`. Note, that the frozen core approximation is ignored in this case.

9.4 General Comments on MP2 Calculations, Practical Hints

Recommendations

- It is well-known, that perturbation theory yields reliable results only, if the perturbation is small. This is also valid for MP2, which means, that MP2 improves HF results only, if HF already provides a fairly good solution to the problem. If HF fails, e.g. in

case of partially filled d-shells, MP2 usually will also fail and should not be used in this case.

- MP2 results are known to converge very slowly with increasing basis sets, in particular slowly with increasing l-quantum number of the basis set expansion. Thus for reliable results the use of TZVPP basis sets (or higher) is recommended. When using SVP basis sets at most a qualitative trend can be expected. Basis sets much larger than TZVPP usually do not significantly improve geometries of bonded systems, but still can improve the energetic description. For non–bonded systems larger basis sets (especially, with more diffuse functions) are needed.
- It is recommended to exclude all non-valence orbitals from MP2 calculations, as neither the `TURBOMOLE` standard basis sets SVP, TZVPP, and QZVPP nor the cc-pVXZ basis set families (with X=D,T,Q,5,6) are designed for correlation treatment of inner shells (for this purpose polarisation functions for the inner shells are needed). The default selection for frozen core orbitals in `define` (orbitals below -3 a.u. are frozen) provides a reasonable guess. If core orbitals are included in the correlation treatment, it is recommended to use basis sets with additional tight correlation functions as e.g. the cc-pwCVXZ and cc-pCVXZ basis set families.
- RI-MP2: We strongly recommend the use of auxiliary basis sets optimized for the corresponding orbital basis sets.

RI-MP2 calculations with the `ricc2` program: All what is needed for a RI-MP2 gradient calculation with the `ricc2` program is a `$ricc2` data group with the entry `geoopt model=mp2`. If you want only the RI-MP2 energy for a single point use as option just `mp2`. To activate in MP2 energy calculations the evaluation of the D_1 diagnostic (for details see Sec. 10.1). use instead `mp2 d1diag`. (Note that the calculation of the D_1 diagnostic increases the costs compared to a MP2 energy evaluation by about a factor of three.)

Comments on the Output

- Most important output for `ricc2`, `pnoccsd`, and `mpgrad` are of course the MP2(+HF) energies (written to standard output and additionally to the file `energy`) and MP2(+HF) gradients (written to the file `gradient`).
- In case of MP2 gradient calculations the modules also calculate the MP2 dipole moment from the MP2 density matrix (note, that in case of `mpgrad` a frozen core orbital specification is ignored for gradient calculations and thus for MP2 dipole moments).

Further output contains indications of the suitability of the (HF+MP2) treatment.

- As discussed above, MP2 results are only reliable if the MP2 corrections to the Hartree-Fock results are small. One measure for size of MP2 corrections to the wavefunction are the doubles amplitudes, t_{ab}^{ij}, as is evident from the above equations. `mpgrad` by default prints the five largest amplitudes as well as the five largest norms

of amplitude matrices $\mathbf{t}^{ij}$ for fixed i and j. The number of printed amplitudes can be changed by setting the data group `$tplot` n where n denotes the number of largest amplitudes to be plotted. It is up to the user to decide from these quantities, whether the HF+MP2 treatment is suited for the present problem or not. Unfortunately, it is not possible to define a threshold, which distinguishes a "good" and a "bad" MP2-case, since the value of individual amplitudes or amplitudes matrices $\mathbf{t}^{ij}$ are not orbital-invariant, but depend on the orbital basis and thereby under certain circumstances on the orientation, the point group, or the start guess for the MOs. Example: the largest norm of t-amplitudes for the Cu-atom ($d^{10}s^1$, "good" MP2-case) amounts to ca. 0.06, that of the Ni-atom (d^8s^2, "bad" MP2 case) is ca. 0.14.

- A more reliable criterion is obtained from the MP2 density matrix. Its eigenvalues reflect the changes in occupation numbers resulting from the MP2 treatment, compared to the Hartree-Fock level, where occupation numbers are either one (two for RHF) or zero. Small changes mean small corrections to HF and thus suitability of the MP2 method for the given problem. In case of gradient calculations `ricc2` displays by default the largest eigenvalue of the MP2 density matrix, i.e. the largest change in occupation numbers (in %). If `$cc2_natocc` is set the full set of natural occupation numbers and orbitals will be save to the `control` file. For main group compounds largest changes in occupation numbers of ca. 5 % or less are typical, for d^{10} metal compounds somewhat higher values are tolerable.
- A similar idea is pursued by the D_2 and D_1 diagnostics [217,218] which is implemented in `ricc2`. D_2 is a diagnostic for strong interactions of the HF reference state with doubly excited determinants, while D_1 is a diagnostic for strong interactions with singly excited determinants.

9.5 RI-MP2-F12 Calculations

To obtain the F12 correction to the MP2 energy, the data group `$rir12` must be added to the `control` file. A typical run will include the input:

```
$ricc2
  mp2 energy only
$rir12
```

The MP2-F12 ground-state energy is

$$E_{\text{MP2-F12}} = E_{\text{MP2}} + E_{\text{F12}}, \tag{9.5}$$

where E_{MP2} is the conventional MP2 energy and E_{F12} the correction from explicitly-correlated theory. The second term contains contributions from explicitly-correlated geminal basis functions of the form

$$\hat{Q}_{12} f_{12} |ij\rangle, \tag{9.6}$$

where $|ij\rangle$ is a two-electron determinant of occupied (semi-)canonical Hartree–Fock spin orbitals, f_{12} is a correlation factor, which can be either linear r_{12} (in this case, the approach is denoted MP2-R12 instead of MP2-F12) or a function of r_{12}, and $\hat{Q}_{12}$ defines the doubles excitation space covered by the geminals (it also ensures strong orthogonality to the occupied orbitals). Usually $\hat{Q}_{12}$ is chosen to be $\hat{Q}_{12} = (1 - \hat{O}_1)(1 - \hat{O}_2) - \hat{V}_1\hat{V}_2$, where $\hat{O}_\mu = \sum_k |\varphi_k(\mu)\rangle\langle\varphi_k(\mu)|$ is the projection operator onto the space spanned by the occupied spin orbitals φ_k and $\hat{V}_\mu = \sum_a |\varphi_a(\mu)\rangle\langle\varphi_a(\mu)|$ is the projector onto the virtual spin orbitals.

The F12 correction is obtained by minimizing the functional

$$F_{\mathrm{F12}} = \sum_{i<j} \left\{ \mathbf{c}_{ij}^T \mathbf{B}_{ij} \mathbf{c}_{ij} + 2\mathbf{c}_{ij}^T \mathbf{v}_{ij} \right\} \tag{9.7}$$

with respect to the amplitudes collected in the vector $\mathbf{c}_{ij}$. The vectors $\mathbf{v}_{ij}$ and the matrices $\mathbf{B}_{ij}$ are defined as

$$\mathbf{v}_{ij}(kl) = \langle kl | f_{12}\hat{Q}_{12} r_{12}^{-1} | ij \rangle, \tag{9.8}$$

$$\mathbf{B}_{ij}(kl, mn) = \langle kl | f_{12}\hat{Q}_{12}(\hat{f}_1 + \hat{f}_2 - \varepsilon_i - \varepsilon_j)\hat{Q}_{12} f_{12} | mn \rangle, \tag{9.9}$$

in the spin-orbital formalism (m, n denote spin orbitals and $|mn\rangle$ is a two-electron determinant). $\hat{f}_\mu$ is the Fock operator for electron μ and ε_k is a (semi-)canonical Hartree–Fock orbital energy.

The F12 implementation is compatible with ECP, DKH and X2C scalar relativistic treatments. The additional terms that arise in the Fock matrix are included in the evaluation of the F12 contributions. [219, 220]

A MP2-F12 calculation is defined through a number of choices concerning the nature of the geminals (f_{12} and $\hat{Q}_{12}$), the geminal excitation space (ijkl or ijij) and approximations in computing the B matrix (GBC, EBC, $[\hat{T}, f_{12}]$). These choices correspond to keywords in the `$rir12` data group, explained below.

To run a MP2-F12 calculation, one has to select the auxiliary basis sets `cbas`, `cabs` and optionally `jkbas`. The `ricc2` program uses the robust fitting techniques of Ref. [221] for the F12 integrals and the `cbas` basis is used for both the F12 and the usual MP2 Coulomb integrals. For the density fitting of the Coulomb and exchange matrices of the Fock matrix, the `jkbas` will be used instead of the `cbas` basis if it is included in the control file (this is recommended and is achieved using the `rijk` menu in `define`). For the RI approximation of the 3- and 4-electron integrals as sums of products of 2-electron integrals, intrinsic to the F12 method, the complementary auxiliary basis (CABS) approach is used [222]. If `define` is used to set up the `cabs` basis, the library `cabasen` is searched. This library contains the optimised `cabs` basis sets [223] for the cc-pVXZ-F12 basis sets of Peterson *et al.* [224]. For other basis sets, the auxilliary basis in the library `cabasen` is identical with the auxilliary basis in the library `cbas`.

The `$rir12` data group may be set by choosing the `f12` option in the `cc` menu when running `define`. This command activates the `f12` menu, where the default options may be changed if desired:

```
 Orbital basis       : cc-pVTZ-F12
 Cardinal number     : T
 Recommended exponent:   1.0000
 Actual      exponent:   1.0000

INPUT MENU FOR MP2-F12 CALCULATIONS

 ansatz    : CHOOSE ANSATZ               2      [1,2*,2]
 r12model  : CHOOSE MODEL                B      [A,B]
 comaprox  : COMMUTATOR APPROXIMATION    F+K    [F+K,T+V]
 cabs      : CABS ORTHOGONALIZATION      svd  1.0D-08   [cho,svd]
 examp     : CHOOSE FORMULATION          fixed noflip [inv,fixed,noinv, flip,noflip]
 r12orb    : CHOOSE GEMINAL ORBITALS     hf     [hf,rohf,boys,pipek]
 corrfac   : CHOOSE CORRELATION FACTOR   LCG    [R12,LCG]
 cabsingles: CABS SINGLE EXCITATIONS     on     [on,off]
 pairenergy: PRINT OUT PAIRENERGIES      off    [on,off]
 slater    : SLATER EXPONENT             1.0000

 * / end    : write $rir12 to file and leave the menu
 &          : go back - leaving $rir12 unchanged...
```

`ansatz` corresponds to the choice of $\hat{Q}_{12}$. Almost all modern MP2-F12 calculations use ansatz `2` (default), which gives much improved energies over ansatz `1` (see Ref. [225] for details). The principal additional cost of using ansatz `2` over ansatz `1` is concerned with the coupling between the F12 and conventional amplitudes. This is avoided by choosing `2*`, which corresponds to neglecting EBC (Extended Brillouin Condition) terms in the Fock matrix elements.

`r12model` is the method of computing the matrices $\mathbf{B}_{ij}$ (see Ref. [225] for details). The cost and accuracy increases from `A` to `B`. It is recommended to use `B` (default). The energies computed using `A` are then also printed out in the output.

`comaprox` is the method for approximately computing the integrals for the operator $[\hat{T}, f_{12}]$, where the matrix representations of `F+K` or `T+V` are used. `F+K` (the core Hamiltonian plus Coulomb term) is recommended and is the default.

`cabs` refers to the method of orthogonalising the orbitals in the complementary auxiliary basis. Singular-value decomposition (`svd`) or Cholesky decomposition (`cho`) are available. `svd` is recommended and is the default, with a threshold of `1.0d-08`. The basis set used for CABS is set from the `cc` menu.

`examp` refers to the choice of excitation space. `inv` is the orbital-invariant merhod of Ref. [226], with amplitudes $c_{ij}(kl)$. `noinv` is the original orbital-dependent diagonal "ijij" method of Ref. [226], with amplitudes $c_{ij}(ij)$ (not recommended, unless in combination with localised orbitals). `fixed` is the (diagonal and orbital-invariant) rational generator approach of Ref. [227], where the F12 amplitudes are not optimised but predetermined using the coalescence conditions (default). An additional keyword `noflip` suppresses the use of spin-flipped geminals in open-shell calculations; by default spin-flipped geminals are used as described in Ref. [22].

`r12orb` controls which orbitals are used in the F12 energy contribution. `hf` means that (semi-)canonical Hartree–Fock orbitals are used (default). `rohf` means that ROHF orbitals are used (any frozen orbitals will then also implicitly be ROHF). For calculations on closed-shell systems, localised orbitals may be used. Both the Boys [228] and Pipek–Mezey [229] methods are available for localisation of the orbitals.

`corrfac` corresponds to the choice of correlation factor f_{12} in the geminal basis functions. `R12` results in a calculation using linear-r_{12} and `LCG` results in a calculation using the Slater-type correlation factor with exponent 1.4 a_0^{-1}, represented as a linear combination of six Gaussians (see Ref. [230]). Note that the exponents 0.9, 1.0 and 1.1 a_0^{-1} are recommended for use with the cc-pVXZ-F12 basis sets [224].

`cabsingles` switches on/off the calculation of a second-order correction to the Hartree–Fock energy by accounting for single excitations into the complementary auxiliary basis set (CABS). The single excitations into the CABS basis can be computed without extra costs if the CABS Fock matrix elements are required anyway for the F12 calculation (*i.e.*, for ansatz `2`, approximation `B` or comaprox `F+K`). The computation of CABS singles cannot be switched off if it is free of costs.

`pairenergy` controls whether or not the F12 contribution to the MP2 pair energies appear in the output (default `off`).

Further options:

`corrfac LCG` refers to a further data group for the definition of the correlation factor. When `define` is used, the default is

```
$lcg
  nlcg    6
  slater  1.4000
```

The nature of the LCG correlation factor may be changed by editing this data group in the control file. For example, to use a Slater-type correlation factor with exponent 1.0 a_0^{-1}, represented as a linear combination of three Gaussians, use

```
$lcg
  nlcg    3
  slater  1.0000
```

Alternatively, the exponents and coefficients of the fit may be given explicitly:

```
$lcg
  nlcg    3
  expo1  coef1
  expo2  coef2
  expo3  coef3
```

MP2-F12 calculations may be combined with Grimme's SCS approach (S. Grimme, *J. Chem. Phys.* **118** (2003) 9095) by inserting `scs` in `$ricc2`,

```
$ricc2
  mp2 energy only
  scs
```

In this case, the SCS parameters cos=6/5 and css=1/3 are used. Also individual scaling factors for the same-spin and opposite-spin contributions may be defined, see Section 10.8.

For open-shell calculations, two choices of the `examp fixed noflip` method are available. These are controled by a keyword in the `$rir12` data group

```
  ump2fixed  full      [diag,full]
```

These differ in the treatment of the $\alpha\beta$ block, where either only the diagonal excitations enter (with amplitude 0.5) `diag`, or the equivalent of the spin-adapted singlet and triplet pair excitations enter (as far as possible) `full`. Note that the `diag` method with UMP2-F12 yields a result different to that of `fixed` MP2-F12, even for identical RHF and UHF determinants. However, the `diag` method is somewhat less expensive than the `full` method.

Recommendations for orbital and auxiliary basis sets:

The best orbital basis sets to use for MP2-F12 calculations are probably the cc-pVXZ-F12 basis sets, specially optimised for MP2-F12 calculations [224] for the atoms H, He, B–Ne and Al–Ar. In conjunction with these cc-pVXZ-F12 basis sets, we recommend to use the optimised cc-pVXZ-F12 sets of Yousaf and Peterson [223] as `cabs`. Furthermore, `cbas` and `jkbas` basis sets can be selected from the `cbasen` and `jkbasen` libraries, respectively, using the alias cc-pVXZ-F12 (a `jkbas` is currently not available for He, Ne and Ar). This alias points to the corresponding aug-cc-pwCV(X+1)Z `cbas` and aug-cc-pV(X+1)Z `jkbas`. These recommendations are on the side of caution and are likely to be refined as more experience is gained [219, 231, 232].

For atoms other than H, He, B–Ne and Al–Ar, optimised F12 basis sets are not yet available. In this case, basis sets must be selected and/or optimised carefully. It is advised to contact the Theoretical Chemistry Group in Karlsruhe for support (e-mail to: `klopper@kit.edue`).

9.6 Laplace-transformed SOS-RI-MP2 with $\mathcal{O}(\mathcal{N}^4)$ scaling costs

The `ricc2` module contains an implementation of SOS-MP2 which exploits the RI approximation and a Laplace transformation of the orbital energy denominators

$$\frac{1}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} = \int_0^\infty e^{-(\epsilon_a+\epsilon_b-\epsilon_i-\epsilon_j)t} dt \approx \sum_{\alpha=1}^{n_L} w_\alpha e^{-(\epsilon_a+\epsilon_b-\epsilon_i-\epsilon_j)t_\alpha} \quad , \tag{9.10}$$

to achieve an implementation with $\mathcal{O}(\mathcal{N}^4)$ scaling costs, opposed to the conventional $\mathcal{O}(\mathcal{N}^5)$ scaling implementation. In particular for large molecules the Laplace-transformed implementation can reduce a lot the computational costs of SOS-MP2 calculations without loss in accuracy.

The Laplace-transformed implementation for SOS-MP2 calculations is activated with the input

```
$laplace
   conv=5
```

where the parameter `conv` is a convergence threshold for the numerical integration in Eq. (9.10). A value of `conv=5` means that the numerical integration will be converged to a root mean squared error of $\approx 10^{-5}$ a.u.

Whether the conventional or the Laplace-transformed implementation will be more efficient depends firstly on the system size (the number of occupied orbitals) and secondly on the required accuracy (the number of grid points for the numerical integration in Eq. (9.10)) and can be understood and estimated from the following considerations:

- The computational costs for the most expensive step in (canonical) RI-MP2 energy calculations for large molecules requires $\frac{1}{2}O^2V^2N_x$ floating point multiplications, where O and V are, respectively, the number occupied and virtual orbitals and N_x is the number of auxiliary functions for the RI approximation. For the LT-SOS-RI-MP2 implementation the most expensive step involves $n_L OVN_x^2$ floating point multiplications, where n_L is the number of grid points for the numerical integration. Thus, the ratio of the computational costs is approximately

  $$conv : LT \approx \frac{\frac{1}{2}O^2V^2N_x}{n_L OVN_x^2} = \frac{OV}{2n_L N_x} \approx O : 6n_L \quad ,$$

  where for the last step $N_x \approx 3V$ (typical for valence TZ basis sets) has been assumed. Thus, the Laplace-transformed implementation will be faster than the conventional implementation if $O > 6n_L$.

The number of grid points n_L depends on the requested accuracy and the spread of the orbital energy denominators in Eq. (9.10). The efficiency of Laplace-transformed SOS-RI-MP2 calculations can therefore (in difference to conventional RI-MP2 calculations) be

enhanced significantly by a carefull choice of the thresholds, the basis set, and the orbitals included in the correlation treatment:

- The threshold `conv` for the numerical integration is by default set to the value of `conv` specified for the ground state energy in the data group `$ricc2` (see Sec. 23.2.22), which is initialized using the threshold `$denconv`, which by default is set conservatively to the tight value of 10^{-7}.
  - For single point energy calculations `conv` in `$laplace` can savely be set to 4, which gives SOS-MP2 energies converged within $\approx 10^{-4}$ a.u. with computational costs reduced by one third or more compared to calculations with the default settings for these thresholds.
  - For geometry optimizations with SOS-MP2 we recommend to set `conv` in `$laplace` to 5.
- The spread of the orbital energy denominators depends on the basis sets and the orbitals included in the correlation treatment. Most segmented contracted basis sets of triple-ζ or higher accuracy (as e.g. the TZVPP and QZVPP basis sets) lead to rather high lying "anti core" orbitals with orbital energies of 10 a.u. and more.
  - For the calculation of SOS-MP2 valence correlation energies it is recommended to exclude such orbitals from the correlation treatment (see input for `$freeze` in Sec. 23).
  - Alternatively one can use general contracted basis sets, as e.g. the correlation consistent cc-pVXZ basis sets. But note that general contracted basis sets increase the computational costs for the integral evaluation in the Hartree-Fock and, for gradient calculations, also the CPHF equations and related 4-index integral derivatives.
  - Also for the calculation of all-electron correlation energies with core-valence basis sets which include uncontracted steep functions it is recommended to check if extremely high-lying anti core orbitals can be excluded.

Note that for large molecules it is recommended to disable for geometry optimizations (or for gradient or property calculations in general) the preoptimization for the Z vector equations with the `nozpreopt` option in the `$response` data group (see Sec. 23.2.22).

Restrictions:

- It is presently not compatible with the calculation of the D_1 and D_2 diagnostics. The respective options will be ignored by program if the Laplace-transformed implementation is used.

9.7 COSMO-MP2

In TURBOMOLE MP2 can be combined in two ways with COSMO solvation model, which are known in the literature as "Perturbation Theory on Energy" (PTE) and "Perturbation Theory on Energy and Density" (PTED) approaches. The PTE approach is obtained by truncating the free energy of the solute strictly at second-order in the difference between the Hartree-Fock mean field and the actual electron-electron interaction, where the solvent-mediated electron-electron interaction at the same footing direct electron-electron interaction. In the PTED approach the polarizable environment is self-consistently equilibrated with the correlated MP2 density. Thereby some higher-order contributions are included. These higher-order terms should be small since supposition underlying MP2 is that the difference between the Hartree-Fock and the MP2 wavefunction should be small — if this is not fulfilled, MP2 is not adequate.

The PTE-COSMO-MP2 energy is obtained by adding to the COSMO-HF free energy the MP2 correlation energy evaluated with the MOs and the Fock matrix from a COSMO-HF calculation. The PTE-COSMO-MP2 approach is implemented in `mpgrad` for energies and in `ricc2` for energies, first-order properties and gradients with closed-shell RHF or UHF reference wavefunctions. In `ricc2` PTE-COSMO-MP2 calculations can be combined with spin-component scaling (SCS or SOS) and for SOS with the $\mathcal{O}(\mathcal{N}^4)$-scaling LT-based implementation.

Current restrictions for PTE-COSMO-MP2 are:

- Not available for ROHF reference wavefunctions
- Not available for second-order properties (polarizabilities)
- Not available for vibrational frequencies (see Sec. 19.2 for the complications that arise for the calculation of vibrational frequencies in solution)

The PTED approach in implemented in both `mpgrad` and `ricc2` only for energies. No gradients and no other properties are available in the two programs.

9.8 Low-scaling MP2 and MP2-F12 calculations with a hybrid OSV-PNO approximation

For a PNO-MP2 calculation with default thresholds and standard basis sets the `pnoccsd` program does not require any special input apart from `$freeze` and `$maxcor` and the input needed for the Hartree-Fock calculation. It is, however, recommended to specify the PNO truncation threshold in the data group `$pnoccsd`. For further details see Secs. 12 and 23.2.24.

Running `pnoccsd` parallel The MP2 part of the `pnoccsd` program is parallelized with OMP for shared-memory and with MPI for distributed memory architectures. Important

for the performance of the parallel `pnoccsd` calculations are the settings for the core memory usage and for the directories where large integral and scratch files are stored.

The keyword `$maxcor` defines for `pnoccsd` (as for `ricc2` and `ccsdf12`) the core memory usage. Note, however, that `$maxcor` defines only the memory controlled by the electronic structure code. Additional memory can be allocated by the math and MPI libraries linked into the program and by the operating and I/O systems. It is therefore recommended to set `$maxcor` not higher than to 75% of the physical core memory that is available for the calculation.

For MPI parallel `pnoccsd` calculations it is very important to set with the `$tmpdir` keyword the path to directories in fast local file systems to avoid that large integral and scratch files are stored in the NFS file system where the calculation is started.

Chapter 10

Second-Order Approximate Coupled-Cluster (CC2) Calculations

`ricc2` is a module for the calculation of excitation energies and response properties at a correlated second-order *ab initio* level, in particular the second-order approximate coupled-cluster model CC2 [233], but also the MP2, CIS(D), CIS(D_∞), and ADC(2) levels. All calculations employ the resolution-of-the-identity (RI) approximation for the electron repulsion integrals used in the correlation treatment and the description of excitation processes. At present the following functionalities are implemented:

ground state energies for MP2 and CC2 and spin-component scaled variants thereof; the MP2 results are identical with those obtained with `rimp2` (but usually the calculations are somewhat faster).

excitation energies for the models CIS/CCS, CIS(D), CIS(D_∞), ADC(2), and CC2 including spin-component scaled SCS and SOS version of of the latter four methods

transition moments for ground state—excited and excited—excited state transitions for the models CCS and CC2; for ADC(2) only moments for ground state—excited state transitions are available

two-photon transition moments for ground state—excited state transitions for the models CCS and CC2

induced transition moments for ground state—excited state transitions for the models CCS and CC2 for the computation of spin-orbit induced oscillator strengths for transitions from the ground state to excited triplet states and phosphorescence lifetimes with SOC-PT

first-order properties for the ground state with SCF (CCS), MP2, and CC2 and for excited states with CCS, CC2, ADC(2) and CIS(D_∞)

geometric gradients for the electronic ground state at the MP2 and the CC2 level; for electronically excited states at the CIS(D_∞), ADC(2), and CC2 level

second-order properties (linear response function): for the ground state with MP2 and CC2 and a closed-shell RHF reference wavefunction, frequency-dependent properties are restricted to CC2

third-order properties (quadratic response function): for the ground state with CC2 and a closed-shell RHF reference wavefunction

gradients for auxiliary basis sets for RI-MP2, -CC2, etc. calculations based on the RI-MP2 error functional

F12 corrections to RI-MP2; MP2 ground-state energies can be computed (in C_1 symmetry) using explicitly-correlated two-electron basis functions in the framework of the MP2-F12 model [231, 234].

solvent effects for the methods and states for which (orbital–relaxed) densities are available equilibrium solvent effects can be included in the framework of the `cosmo`mode (for details see Chapter 19.2).

damped response also known as complex polarization propagator for the CC2 linear response function (currently restricted for CC2, no spin component scaling, and can not be combined with COSMO, PE, or FDE, only parallelized with OMP, not yet with MPI)

All functionalities at the MP2 and CC2 level are implemented for closed-shell RHF and open-shell UHF reference wavefunctions (with the exception of induced transition moments using SOC-PT, which are only available for a closed-shell RHF reference). Ground state energies for MP2, MP2-F12 and CC2 and excited state energies for CC2 are also implemented for single determinant restricted open-shell Hartree-Fock (ROHF) reference wavefunctions (cmp. Sec. 9.3). (Note, that no gradients are available for MP2 and CC2 with ROHF reference wavefunctions.) For a two-component GHF reference wavefunction energies for the CCS, MP2/ADC(2), CIS(D_∞) and CC2 methods as well as ground state—excited state transtition moments for ADC(2) and CC2 are available.

The second-order models MP2, CIS(D), CIS(D_∞), ADC(2) and CC2 can be combined with a spin-component scaling (SCS or SOS). (Not yet available for second-order properties, two-photon and induced transition moments.) For the SOS variants one can switch to an implementation with $\mathcal{O}(\mathcal{N}^4)$-scaling costs by setting the keyword for the numerical Laplace transformation (LT) (`$laplace`) .

As listed above, some functionalities are, as a side-produce, in `ricc2` also implemented at the uncorrelated HF-SCF, CIS, and CCS levels. There are only made available in `ricc2` for easier test calculations and comparisons, without that the code in `ricc2` has optimized for them.

For calculations with CCSD, CCSD(T) and other higher-order models beyond CC2 see Chapter 11.

Prerequisites

Calculations with the `ricc2` module require (almost) the same prerequisites as RI-MP2 calculations:

1. a converged SCF calculation with the one-electron density convergence threshold set to `$denconv 1.d-5` or less
2. if non-standard basis sets used: an auxiliary basis defined in the data group `$cbas` (for standard basis sets, where a corresponding auxiliary basis set is found in the basis set library, the program will automatically use this if `$cbas` is not set)
3. if orbitals should be excluded from the correlation treatment (and excitation processes) the data group `$freeze` has to be set
4. the maximum core memory which the program is allowed to allocate should be defined in the data group `$maxcor`; the recommended value is 66–75% of the available (physical) core memory.
5. depending on the type of calculations that should be carried out, additionally the data groups `$ricc2`, `$excitations`, `$response`, `$laplace`, `$rir12` and `$lcg` have to be set (see below and Section 23.2.22).

For calculations with the `ricc2` program it is recommended to use the `cc2` submenu of the `define` program to set the data groups `$denconv`, `$freeze`, `$cbas`, `$maxcor`. MP2-F12 calculations require in addition the data groups `$rir12`, `$cabs`, `$jkbas` and `$lcg`. The exponent of the Slater function in the interelectronic distance r_{12}, which appears in the geminals used MP2-F12 is defined in the data group `$lcg` and should be adapted to the one-electron basis set which is used.

Note, that the implementation of non-Abelian point groups in `ricc2` is limited to the electronic ground state (but comprises most of the RI-MP2 functionality included in `ricc2`). In the present version `ricc2` can for excited states only deal with real Abelian point groups (C_1, C_s, C_2, C_i, C_{2h}, C_{2v}, D_2, D_{2h}). The F12 correction can only be calculated in the C_1 point group.

How To Perform a Calculation

Single point calculations:

Call the `ricc2` program after a converged SCF calculation, which can be carried either with the `dscf` or the `ridft` program.

Geometry optimizations and molecular dynamics:

Invoke `jobex` with the `-level cc2` option; see Section 5.1 for additional options and parameters of the `jobex` script that might be needed or useful for geometry optimizations and *ab initio* molecular dynamics calculations.

Force constants and vibrational frequencies:

Force constants can be calculated by numerical differentiation of the gradients. Invoke for this `NumForce` with the `-level cc2` option; see Chapter 15 for details about `NumForce`. The usage of the `NumForce` interface for excited states is restricted to C_1 symmetry.

Note: using `ricc2` in connection with `jobex` or `NumForce` requires that the method and the electronic state, for which the gradient should be calculated and written to the interface files, is specified in the option `geoopt` (see Section 10.3.1) in datagroup `$ricc2` (see Section 23.2.22). For calculations on excited states this state has in addition to be included in the input for excitation energies in datagroup `$excitations`.

RI-SCF reference wavefunctions: The `ricc2` program can be used in combination with conventional SCF or with the RI-J and RI-JK approximations for SCF, with the exception that the calculation of gradients for reference wavefunctions which employ only the RI-J approximation for the Coulomb matrix but 4-index integrals for the exchange matrix is presently not supported. The implementation of gradients in `ricc2` assumes that the reference wavefunction has either been calculated without RI-J approximation (using `dscf`) or with the RI-JK approximation (using `ridft`).

See Chapter 6 for a discussion of the RI approximations in SCF calculations and 23.2.10 for the required input. In geometry optimizations with `jobex` and for the calculation of force constants and vibrational spectra with `NumForce`, the `ricc2` program is used in combination with the RI-JK approximation for the Hatree-Fock calculation (using `ridft`) if `jobex` and `NumForce` are invoked with the `-rijk` option.

How to quote

If results obtained with the `ricc2` program are used in publications, the following citations should be included if you have used the methods, program parts, auxiliary basis sets, or results reported in therein:

Methods:

- for the approximate coupled-cluster singles-and-doubles model CC2:
  O. Christiansen, H. Koch, P. Jørgensen, *Chem. Phys. Lett.*, **243** (1995) 409–418.
- for CI singles with a perturb. correct. for connected double excitations, CIS(D):
  M. Head-Gordon, R. J. Rico, M. Oumi and T. J. Lee, *Chem. Phys. Lett.*, **219** (1994) 21.
  and for the iterative CIS(D_∞) variant:
  M. Head-Gordon, M. Oumi and D. Maurice, *Mol. Phys.* **96** (1999) 593.
- for the algebraic diagrammatic construction through second order ADC(2):
  J. Schirmer, *Phys. Rev. A.* **26** (1981) 2395. A. B. Trofimov and J. Schirmer, *J. Phys. B.* **28** (1995) 2299.

- for MP2-F12:
  W. Klopper and C. C. M. Samson, *J. Chem. Phys.* **116** (2002) 6397–6410.
  D. P. Tew and W. Klopper, *J. Chem. Phys.* **123** (2005) 074101.
- for the SCS and SOS variants of MP2:
  S. Grimme, *J. Chem. Phys.* **118** (2003) 9095 (SCS) or Y., Jung, R.C. Lochan, A.D. Dutoi, M. Head-Gordon, *J. Chem. Phys.* **121** (2004) 9793 (SOS).
- for the SCS and SOS variants of CC2 and ADC(2):
  A. Hellweg, S. Grün, C. Hättig, *Phys. Chem. Chem. Phys.* **10** (2008) 4119-4127.
- for the two-component CCS, ADC(2), CIS(D_∞) and CC2 methods:
  K. Krause and W. Klopper, *J. Chem. Phys.* **142** (2015) 104109.

Implementation:

- please, include always a reference to the publication reporting the implementation of the core part of the `ricc2` program:
  C. Hättig and F. Weigend, *J. Chem. Phys.* **113** (2000) 5154.
- for transition moments and excited state first order properties:
  C. Hättig and A. Köhn, *J. Chem. Phys.* **117** (2002) 6939.
- for triplet excited states include:
  C. Hättig and K. Hald, *Phys. Chem. Chem. Phys.* **4** (2002) 2111. C. Hättig, A. Köhn and K. Hald, *J. Chem. Phys.* **116** (2002) 5401.
- for ground state geometry optimizations include:
  C. Hättig, *J. Chem. Phys.* **118** (2003) 7751.
- for geometry optimizations for excited states include:
  A. Köhn and C. Hättig, *J. Chem. Phys.* **119** (2003) 5021.
- for calculations with RI-ADC(2), RI-CIS(D), RI-CIS(D_∞) include:
  C. Hättig, *Adv. Quant. Chem.* **50** (2005) 37.
- if the parallel version of `ricc2` is used include a reference to:
  C. Hättig, A. Hellweg, A. Köhn, *Phys. Chem. Chem. Phys.* **8** (2006) 1159.
- for transition moments between excited states:
  M. Pabst and A. Köhn, *J. Chem. Phys.* **129** (2008) 214101.
- for RI-MP2-F12 calculations:
  R. A. Bachorz, F. A. Bischoff, A. Glöß, C. Hättig, S. Höfener, W. Klopper, D. P. Tew, *J. Comput. Chem.* **32** (2011) 2492.
- for $\mathcal{O}(\mathcal{N}^4)$-scaling calculations using the Laplace transformation:
  - ground-state and excitation energies:
    N. O. C. Winter, C. Hättig, *J. Chem. Phys.* **134** (2011) 184101.
  - transition moments, first-order properties and gradients:
    N. O. C. Winter, C. Hättig, *Chem. Phys.* **401** (2012) 217.

- for second-order properties (relaxed or unrelaxed):
  D. H. Friese, N. O. C. Winter, P. Balzerowski, R. Schwan, C. Hättig, *J. Chem. Phys.* **136** (2012) 174106.
- for two-photon transition moments:
  D. H. Friese, C. Hättig, K. Rudd, *Phys. Chem. Chem. Phys.* **14** (2012) 1175–1184.
- for phosphorescence live times with SOC-PT-CC2:
  B. Helmich-Paris, C. Hättig, C. van Wüllen, *J. Chem. Theory Comput.* **12** (2016) 1892–1904.
- for damped response in the linear response function:
  - default symmetric form:
    D. Fedotov, S. Coriani, C. Hättig, *J. Chem. Phys.* **154** (2021) 124110.
  - asymmetric form:
    Y. J. Franzke, et al., *J. Chem. Theory Comp.* **???** (2023) ?????.
- for RI-CC2 the quadratic response function:
  J. H. Andersen, S. Coriani, C. Hättig, *ChemRxiv* https://doi.org/10.26434/chemrxiv-2023-dgbdz.
- for the polarizable embedding, PERI-CC2:
  - ground-state and excitation energies and one-photon transition moments:
    T. Schwabe, K. Sneskov, J. M. H. Olsen, J. Kongsted, O. Christiansen, C. Hättig, *J. Chem. Theory Comput.* **8** (2012) 3274–3283.
  - two-photon transition moments:
    D. Hršak, A. M. Khah, O. Christiansen, C. Hättig,
    J. Chem. Theory Comput. **11** (2015) 3669–3678.
  - ground and excited-state gradients:
    A. M. Khan, S. K. Khani, C. Hättig,
    J. Chem. Theory Comput. 14 (2018) 4640–4650.
- for COSMO in `ricc2`:
  S. K. Khani, A. M. Khah, C. Hättig,
  Phys. Chem. Chem. Phys. **20** (2018) 16354-16363.

Auxiliary basis sets:

- the appropriate reference for the auxiliary SVP, TZVP and TZVPP basis sets (for calculations with RI-MP2, RI-CC2 and related methods) is:
  F. Weigend, M. Häser, H. Patzelt, R. Ahlrichs, *Chem. Phys. Lett.* **294** (1998) 143.
- for the auxiliary cc-pVXZ (cc-pV(X+d)Z), aug-cc-pVXZ (aug-cc-pV(X+d)Z) basis sets with X = D, T, or Q cite:
  F. Weigend, A. Köhn, C. Hättig, *J. Chem. Phys.* **116** (2001) 3175.

- for the auxiliary cc-pV5Z (cc-pV(5+d)Z), aug-cc-pV5Z (aug-cc-pV(5+d)Z), cc-pwCVXZ with X = D, T, Q, 5 and QZVPP basis sets the reference is:
  C. Hättig, *Phys. Chem. Chem. Phys.* **7** (2005) 59–66.
  This reference should also be included if you employ the analytic basis set gradients implemented in the `ricc2` program for the optimization of your own auxiliary basis set(s).
- for the auxiliary def2-basis sets from Rb to Rn the reference is:
  A. Hellweg, C. Hättig, S. Höfener, and W. Klopper, *Theor. Chem. Acc.* **117** (2007) 587–597.

- for the auxiliary cc-pVXZ-PP, aug-cc-pVXZ-PP, cc-pwCVXZ-PP, and aug-cc-pwCVXZ-PP basis sets for Ga–Kr, In–Xe, and Tl–Rn:
  C. Hättig, G. Schmitz, J. Koßmann, *Phys. Chem. Chem. Phys.* 14 (2012) 6549–6555.

(For more details on the references for the basis sets included in the basis set libraries of the `TURBOMOLE` distribution see Sec. 1.3 and the library files.)

10.1 CC2 Ground-State Energy Calculations

The CC2 ground-state energy is—similarly to other coupled-cluster energies—obtained from the expression

$$E_{\rm CC} = \langle {\rm HF}|H|{\rm CC}\rangle = \langle {\rm HF}|H\exp(T)|{\rm HF}\rangle \quad , \tag{10.1}$$

$$= E_{\rm SCF} + \sum_{iajb}\left[t_{ab}^{ij} + t_a^i t_b^j\right]\,\Big[2(ia|jb) - (ja|ib)\Big], \tag{10.2}$$

where the cluster operator T is expanded as $T = T_1 + T_2$ with

$$T_1 = \sum_{ai} t_a^i \tau_{ai} \tag{10.3}$$

$$T_2 = \frac{1}{2}\sum_{aibj} t_{ab}^{ij}\tau_{aibj} \tag{10.4}$$

(for a closed-shell case; in an open-shell case an additional spin summation has to be included). The cluster amplitudes t_a^i and t_{ab}^{ij} are obtained as solution of the CC2 cluster equations [233]:

$$\Omega_{\mu_1} = \langle\mu_1|\hat{H} + [\hat{H}, T_2]|{\rm HF}\rangle = 0 \quad , \tag{10.5}$$

$$\Omega_{\mu_2} = \langle\mu_2|\hat{H} + [F, T_2]|{\rm HF}\rangle = 0 \quad , \tag{10.6}$$

with

$$\hat{H} = \exp(-T_1)H\exp(T_1),$$

where μ_1 and μ_2 denote, respectively, the sets of all singly and doubly excited determinants.

The residual of the cluster equations $\Omega(t_{ai}, t_{aibj})$ is the so-called vector function. The recommended reference for the CC2 model is ref. [233], the implementation with the resolution-of-the-identity approximation, RI-CC2, was first described in ref. [11].

Advantages of the RI approximation: For RI-CC2 calculations, the operation count and thereby the CPU and the wall time increases—as for RI-MP2 calculations—approximately with $\mathcal{O}(O^2V^2N_x)$, where O is the number of occupied and V the number of virtual orbitals and N_x the dimension of the auxiliary basis set for the resolution of the identity. Since RI-CC2 calculations require the (iterative) solution of the cluster equations (10.5) and (10.6), they are about 10–20 times more expensive than MP2 calculations. The disk space requirements are approximately $O(2V + N)N_x + N_x^2$ double precision words. The details of the algorithms are described in ref. [11], for the error introduced by the RI approximation see refs. [216, 235].

Required input data: In addition to the above mentioned prerequisites ground-state energy calculations with the `ricc2` module require only the data group `$ricc2` (see Section 23.2.22), which defines the methods, convergence thresholds and limits for the number of iterations etc. If this data group is not set, the program will carry out a CC2 calculation. With the input

```
$ricc2
  mp2
  cc2
  conv=6
```

the `ricc2` program will calculate the MP2 and CC2 ground-state energies, the latter converged to approximately 10^{-6} a.u. The solution for the single-substitution cluster amplitudes is saved in the file `CCR0--1--1---0`, which can be kept for a later restart.

Ground-State calculations for other methods than CC2: The MP2 equations and the energy are obtained by restricting in the CC2 equations the single-substitution amplitudes t_{ai} to zero. In this sense MP2 can be derived as a simplification of CC2. But it should be noted that CC2 energies and geometries are usually not more accurate than MP2.

For CCS and CIS the double-substitution amplitudes are excluded from the cluster expansion and the single-substitution amplitudes for the ground state wavefunction are zero for closed–shell RHF and open–shell UHF reference wavefunctions and thus energy is identical to the SCF energy.

For the Methods CIS(D), CIS(D_∞) and ADC(2) the ground state is identified with the MP2 ground state to define is total energy of the excited state, which is needed for the definition of gradients and (relaxed) first-order properties which are obtained as (analytic) derivatives the total energy.

Diagnostics: Together with the MP2 and/or CC2 ground state energy the program evaluates the D_1 diagnostic proposed by Janssen and Nielsen [217], which is defined as:

$$D_1 = \sqrt{\max\left(\lambda_{\max}\Big[\sum_i t_{ai}t_{bi}\Big], \lambda_{\max}\Big[\sum_a t_{ai}t_{aj}\Big]\right)} \tag{10.7}$$

where $\lambda_{\max}[\mathbf{M}]$ is the largest eigenvalue of a positive definite matrix $\mathbf{M}$. (For CC2 the D_1 diagnostic will be computed automatically. For MP2 is must explictly be requested with the `d1diag` option in the `$ricc2` data group, since for RI-MP2 the calculation of D_1 will contribute significantly to the computational costs.) Large values of D_1 indicate a multireference character of the ground-state introduced by strong orbital relaxation effects. In difference to the T_1 and S_2 diagnostics proposed earlier by Lee and coworkers, the D_1 diagnostic is strictly size-intensive and can thus be used also for large systems and to compare results for molecules of different size. MP2 and CC2 results for geometries and vibrational frequencies are, in general, in excellent agreement with those of higher-order correlation methods if, respectively, $D_1(\text{MP2}) \leq 0.015$ and $D_1(\text{CC2}) \leq 0.030$ [14,217]. For $D_1(\text{MP2}) \leq 0.040$ and $D_1(\text{CC2}) \leq 0.050$ MP2 and/or CC2 usually still perform well, but results should be carefully checked. *Larger values of D_1 indicate that MP2 and CC2 are inadequate to describe the ground state of the system correctly!*

The D_2 diagnostic proposed by Nielsen and Janssen [218] can also be evaluated. This analysis can be triggered, whenever a response property is calculated, e.g. dipole moment, with the keyword `$D2-diagnostic`. *Note that the calculation of D_2 requires an additional $O(N^5)$ step!* $D_2(\text{MP2/CC2}) \leq 0.15$ are in excellent agreement with those of higher-order correlation methods, for $D_2(\text{MP2/CC2}) \geq 0.18$ the results should be carefully checked.

10.2 Calculation of Excitation Energies

With the `ricc2` program excitation energies can presently be calculated with the RI variants of the methods CCS/CIS, CIS(D), CIS(D_∞), ADC(2) and CC2. The CC2 excitation energies are obtained by standard coupled-cluster linear response theory as eigenvalues of the Jacobian, defined as derivative of the vector function with respect to the cluster amplitudes.

$$\mathbf{A}^{\text{CC2}}_{\mu\nu} = \frac{d\Omega_\mu}{dt_\nu} = \begin{pmatrix} \langle\mu_1|[[\hat{H}+[\hat{H},T_2],\tau_{\nu_1}]|\text{HF}\rangle & \langle\mu_1|[\hat{H},\tau_{\nu_2}]|\text{HF}\rangle \\ \langle\mu_2|[\hat{H},\tau_{\nu_1}]|\text{HF}\rangle & \langle\mu_2|[F,\tau_{\nu_2}]|\text{HF}\rangle \end{pmatrix} \tag{10.8}$$

Since the CC2 Jacobian is a non-symmetric matrix, left and right eigenvectors are different and the right (left) eigenvectors E^i_ν ($\bar{E}^i_\mu$) are **not** orthogonal among themselves, but form a biorthonormal basis (if properly normalized):

$$\bar{E}^i E^j = \bar{E}^i_{\mu_1} E^j_{\nu_1} + \bar{E}^i_{\mu_2} E^j_{\nu_2} = \delta_{ij} \quad . \tag{10.9}$$

To obtain excitation energies only the right or the left eigenvalue problem needs to be solved, but for the calculation of transition strengths and first-order properties both, left and right, eigenvectors are needed (see below). A second complication that arises from the non-symmetric eigenvalue problem is that in the case of close degeneracies within the

same irreducible representation (symmetry) it can happen that instead of two close lying real roots a degenerate complex conjugated pair of excitation energies and eigenvectors is obtained. CC2 (and also other standard coupled-cluster response methods) are thus not suited for the description of conical intersections etc. For the general theory behind coupled cluster response calculations see e.g. ref. [236, 237] or other reviews.

The `ricc2` program exploits that the doubles/doubles block of the CC2 Jacobian is diagonal and the (linear) eigenvalue problem in the singles and doubles space can be reformulated as a (non-linear) eigenvalue problem in single-substitution space only:

$$\begin{aligned} \mathbf{A}^{eff}_{\mu_1\nu_1}(t,\omega) &= \mathbf{A}^{\mathrm{CC2}}_{\mu_1\nu_1}(t) - \mathbf{A}^{\mathrm{CC2}}_{\mu_1\gamma_2}(t)(\mathbf{A}_{\gamma_2\gamma_2} - \omega)\mathbf{A}^{\mathrm{CC2}}_{\gamma 2\nu_1}(t) \\ \mathbf{A}^{eff}_{\mu_1\nu_1}(t^{\mathrm{CC2}},\omega^{\mathrm{CC2}})E_{\nu_1} &= \omega^{\mathrm{CC2}}E_{\nu_1} \end{aligned}$$

This allows to avoid the storage of the double-substitution part of the eigen- or excitation vectors E_{ν_2}, $\bar{E}_{\nu_2}$. The algorithms are described in refs. [11, 12], about the RI error see ref. [235].

The solution of the CC2 eigenvalue problem can be started from the solutions of the CCS eigenvalue problem (see below) or the trial vectors or solutions of a previous CC2 excitation energy calculation. The operation count per transformed trial vector for one iteration for the CC2 eigenvalue problem is about 1.3–1.7 times the operation count for one iteration for the cluster equations in the ground-state calculation—depending on the number of vectors transformed simultaneously. The disk space requirements are about $O(V+N)N_x$ double precision words per vector in addition to the disk space required for the ground state calculation.

CCS excitation energies are obtained by the same approach, but here double-substitutions are excluded from the expansion of the excitation or eigenvectors and the ground-state amplitudes are zero. Therefore the CCS Jacobian,

$$\mathbf{A}^{\mathrm{CCS}}_{\mu\nu} = \frac{d\Omega_\mu}{dt_\nu} = \langle \mu_1|[H,\tau_{\nu_1}]|\mathrm{HF}\rangle \quad , \tag{10.10}$$

is a symmetric matrix and left and right eigenvectors are identical and form an orthonormal basis. The configuration interaction singles (CIS) excitation energies are identical to the CCS excitation energies. The operation count for a RI-CIS calculation is $\mathcal{O}(ON^2N_x)$ per iteration and transformed trial vector.

The second-order perturbative correction CIS(D) to the CIS excitation energies is calculated from the expression

$$\omega^{\mathrm{CIS(D)}} = \omega^{\mathrm{CIS}} + \omega^{(\mathrm{D})} = \mathbf{E}^{\mathrm{CIS}}\mathbf{A}^{eff}(t^{\mathrm{MP1}},\omega^{\mathrm{CIS}})\mathbf{E}^{\mathrm{CIS}} \tag{10.11}$$

(Note that t^{MP1} are the first-order double-substitution amplitudes from which also the MP2 ground-state energy is calculated; the first-order single-substitution amplitudes vanish for a Hartree–Fock reference due to the Brillouin theorem.) The operation count for a RI-CIS(D) calculation is similar to that of a single iteration for the CC2 eigenvalue problem. Also disk space requirements are similar.

Running excitation energy calculations: The calculation of excitation energies is initiated by the data group `$excitations` in which at least the symmetries (irreducible representations) and the number of the excited states must be given (for other options see Section 23.2.22). With the following input the `ricc2` program will calculate the lowest two roots (states) for the symmetries A_1 and B_1 of singlet multiplicity * at the CIS, CIS(D) and CC2 level with default convergence thresholds. Ground-state calculations will be carried out for MP2 (needed for the CIS(D) model and used as start guess for CC2) and CC2.

```
$ricc2
  cis
  cis(d)
  cc2
$excitations
  irrep=a1 nexc=2
  irrep=b1 nexc=2
```

The single-substitution parts of the right eigenvectors are stored in files named `CCRE0-`*s*`--`*m*`-`*xxx*, where *s* is the number of the symmetry class (irreducible representation), *m* is the multiplicity, and *xxx* the number of the excitation within the symmetry class. For the left eigenvectors the single-substitution parts are stored in files named `CCLE0-`*s*`--`*m*`-`*xxx*. These files can be kept for later restarts.

Trouble shooting: For the iterative second-order methods CIS(D_∞), ADC(2), and CC2 the solution of the nonlinear partitioned eigenvalue problem proceeds usually in three steps:

1. solution of the CCS/CIS eigenvalue problem to generate reasonable start vectors; the eigenvectors are converged in this step only to a remaining residual norm $<$ `preopt`
2. pre-optimization of the eigenvectors by a robust modified Davidson algorithm (see ref. [11]) using the `LINEAR CC RESPONSE SOLVER` until the norm of all residuals are below `preopt`, combined with a DIIS extrapolation for roots assumed to be converged below the threshold `thrdiis`.
3. solution of the nonlinear eigenvalue problem with a DIIS algorithm using the `DIIS CC RESPONSE SOLVER` until the norm of the residuals are below the required threshold `conv`

This procedure is usually fairly stable and efficient with the default values for the thresholds. But for difficult cases it can be necessary to select tighter thresholds. In case of convergence problems the first thing do is to verify that the ground state is not a multireference case by checking the D1 diagnostic. If this is not the case the following situations can cause problems in the calculation of excitation energies:

*Provided that it is not an unrestricted open shell run. In this case the wavefunctions will not be spin eigenfunctions and multiplicities are not well defined.

- almost degenerate roots in the same symmetry class
- complex roots (break down of the CC approximation close to conical intersections)
- large contributions from double excitations

The first two reasons can be identified by running the program with a print level ≤ 3. It will then print in each iteration the actual estimates for the eigenvalues. If some of these are very close or if complex roots appear, you should make sure that the DIIS procedure is not switched on before the residuals of the eigenvectors are small compared to the differences in the eigenvalues. For this, `thrdiis` (controlling the DIIS extrapolation in the linear solver) should be set about one order of magnitude smaller than the smallest difference between two eigenvalues and `preopt` (controlling the switch to the DIIS solver) again about one order of magnitude smaller then `thrdiis`.

Tighter thresholds or difficult situations can make it necessary to increase the limit for the number of iterations `maxiter`.

In rare cases complex roots might persist even with tight convergence thresholds. This can happen for CC2 and CIS(D_∞) close to conical intersections between two states of the same symmetry, where CC response can fail due to its non-symmetric Jacobian. In this case one can try to use instead the ADC(2) model. But the nonlinear partitioned form of the eigenvalue problem used in the `ricc2` program is not well suited to deal with such situations.

Diagnostics for double excitations: As pointed out in ref. [13], the $\%T_1$ diagnostic (or $\%T_2 = 100 - \%T_1$) which is evaluated directly from the squared norm of the single and double excitation part of the eigenvectors $\%T_1 = 100 \cdot T_1/(T_1+T_2)$ with $T_i = \sum_{\mu_i} E_{\mu_i}^2$ where the excitation amplitudes are for spin-free calculations in a corresponding spin-adapted basis (which is not necessarily normalized) has the disadvantage that the results depend on the parameterization of the (spin-adapted) excitation operators. This prevents in particular a simple comparison of the results for singlet and triplet excited states if the calculations are carried out in a spin-free basis. With the biorthogonal representation for singlet spin-coupled double excitations [236] results for $\%T_1$ also differ largely between the left and right eigenvectors and are not invariant with respect to unitary transformations of the occupied or the virtual orbitals.

The `ricc2` module therefore uses since release 6.5 an alternative double excitation diagnostic, which is defined by $\%\mathcal{T}_1 = 100 * \mathcal{T}_1/(\mathcal{T}_1 + \mathcal{T}_2)$ with $\mathcal{T}_1 = \sum_{ai} E_{ai}^2$ and $\mathcal{T}_2 = \sum_{i>j} \sum_{a>b} E_{aibj}^2$ with E_{ai} and E_{aibj} in the spin-orbital basis. They are printed in the summaries for excitation energies under the headings `%t1` and `%t2`. For spin-adapted excitation amplitudes $\mathcal{T}_1$ and $\mathcal{T}_2$ have to be computed from respective linear combinations for the amplitudes which reproduce the values in the spin-orbital basis. For ADC(2), which has a symmetric secular matrix with identical left and right normalized eigenvectors $\mathcal{T}_1$ and $\mathcal{T}_2$ are identical with the contributions from the singles and doubles parts for the eigenvectors to the trace of the occupied or virtual block of the (orbital unrelaxed) difference density between the ground and the excited state, i.e. the criterium proposed in ref. [13]. Compared to the suggestion from ref. [13] $\mathcal{T}_1$ and $\mathcal{T}_2$ have the additional advantage of that they are for all methods guaranteed to be positive and

can be evaluated with the same insignificantly low costs as T_1 and T_2. They are invariant with respect to unitary transformations of the occupied or the virtual orbitals and give by construction identical results in spin-orbital and spin-free calculations. For CC2 and CIS(D_∞) the diagnostics $\mathcal{T}_1$ and $\mathcal{T}_2$ agree for left and right eigenvectors usually with a few 0.01%, for CIS(D) and ADC(2) they are exactly identical. For singlet excitations in spin-free calculations, $\%\mathcal{T}_2$ is typically by a factors of 1.5–2 larger than $\%T_2$. The second-order methods CC2, ADC(2), CIS(D_∞) and CIS(D) can usually be trusted for $\%\mathcal{T}_2 \leq 15\%$.

For compatibility, the program can be switched to use of the old $\%\mathcal{T}_1$ and $\%\mathcal{T}_2$ diagnostics (printed with the headers `||T1||` and `||T2||`) by setting the flag `oldnorm` in the data group `$excitations`. Note that the choice of the norm effects the individual results left and right one- and two-photon transition moments, while transition strengths and all other observable properties independent of the individual normalization of the right and left eigenvectors.

The $\%\mathcal{T}_2$ and $\%T_2$ diagnostics can not be monitored in the output of the (quasi-) linear solver. But it is possible to do in advance a CIS(D) calculation. The CIS(D) results for the $\%\mathcal{T}_2$ and $\%T_2$ correlate usually well with the results for this diagnostic from the iterativ second-order models, as long as there is clear correspondence between the singles parts of the eigenvectors. Else the DIIS solver will print the doubles diagnostics in each iteration if the print level is set > 3. States with large double excitation contributions converge notoriously slow (a consequence of the partitioned formulation used in the `ricc2` program). However, the results obtained with second-order methods for doubly excited states will anyway be poor. It is strongly recommended to use in such situations a higher-level method.

Visualization of excitations: An easy way to visualize single excitations is to plot the natural transition orbitals that can be obtained from a singular value decomposition of the excitation amplitudes. See Sec. 20.1.7 for further details.

Another, but computational more involved possibility is plot the difference density between the ground and the respective excited state. This requires, however, a first-order property or gradient calculation for the excited state to obtain the difference density. For further details see Sec. 10.3.3.

10.2.1 Core-Valence Separation (CVS) Approximation for Core Spectra

Core excited states are high in energy and, at the CC2 or ADC(2) level, embedded in a dense spectrum of doubly excited states for which the eigenvectors can not easily be converged in the doubles-direct implementation that is used in the `ricc2` program. Two alternatives implemented in `ricc2` for core spectra are damped response (see Secs. 10.5.1 and 10.6.1) and the core-valence separation (CVS) approximation [2, 238].

The CVS approximation decouples the core excited from the valence states by neglecting in the second-quantized Hamiltonian the coupling terms that change the number of electrons in the core orbitals. For the excitation energies for valence states and transition moments

between valence states, the CVS approximation is equivalent to the frozen-core approximation. The excitation energies and amplitudes for singly core excited states are then obtained as eigenpairs of the block of the Jacobian matrix $\mathbf{A}$ or, for ADC(2), the secular matrix with one core hole index ($1ch$):

$$\begin{pmatrix} \mathbf{A}_{val,val} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{1ch,1ch} \end{pmatrix} \tag{10.12}$$

The off-diagonal blocks $\mathbf{A}_{val,1ch}$ and $\mathbf{A}_{1ch,val}$ vanish in the CVS approximation. The calculation of core excited states needs as prerequisite an additional data group `$core_excitations` that specifies the subset of MOs from which core excitations are allowed. These need to be a subset of the core orbitals specified in the data group `$freeze`, e.g. for indole (C_8NH_7) in C_S symmetry at the carbon K-edge:

```
$freeze
  a' 1-9
$core_excitations
  a' 2-9
```

In addition the number of core holes has to be specified in the `irrep` option in data group `$excitations`. Possible values are 0 for valence states, which is the default, and 1 for states with one core hole:

```
$excitations
  irrep= a' multiplicity=1 nexc=4 npre=8 nstart=16 ncore=1
  irrep= a' multiplicity=1 nexc=2 npre=4 nstart=6  ncore=0
  spectrum states=all operators=diplen
```

The CVS approximation is available in the `ricc2` program for excitation energies and transition strengths for between core-excited states and the ground-state or other core- or valence-excited states with CIS, ADC(2), and CC2. First-order properties, gradients, or non-linear spectra (two-photon, phosphorescence, MCD, etc.) for core excited states are not yet available.

10.3 First-Order Properties and Gradients

For the ground state first-order properties (expectation values) are implemented at the SCF, MP2 and CC2 level. Note that for the ground state CCS and CIS are equivalent to SCF. For excited states first-order properties are implemented only at the CCS and CC2 level. Gradients are presently only available for the ground state at the MP2 and the CC2 and for excited states only at the CC2 level.

10.3.1 Ground State Properties, Gradients and Geometries

For CC2, one distinguishes between orbital-relaxed and unrelaxed properties. Both are calculated as first derivatives of the respective energy with respect to an external field corresponding to the calculated property. They differ in the treatment of the SCF orbitals. In the orbital-relaxed case the external field is (formally) already included at the SCF stage and the orbitals are allowed to relax in the external field; in the orbital-unrelaxed case the external field is first applied after the SCF calculation and the orbitals do not respond to the external field. *Orbital-unrelaxed* CC2 properties are calculated as first derivatives of the real part of the unrelaxed Lagrangian [233]

$$\begin{aligned} L^{\text{ur CC2}}(t,\bar{t},\beta) &= \langle \text{HF}|H|\text{CC}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{H} + [\hat{H},T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|\hat{H} + [F_0 + \beta\hat{V},T_2]|\text{HF}\rangle \end{aligned} \tag{10.13}$$

with $H = H_0 + \beta V$—where V is the (one-electron) operator describing the external field, β the field strength, and H_0 and F_0 are the Hamiltonian and Fock operators of the unperturbed system—by the expression:

$$\langle V\rangle^{\text{ur CC2}} = \Re\left(\frac{\partial L^{\text{ur CC2}}(t,\bar{t},\beta)}{\partial\beta}\right)_0 = \sum_{pq} D^{\text{ur}}_{pq}\, V_{pq} \quad , \tag{10.14}$$

$$\begin{aligned} &= \Re\Bigg(\langle \text{HF}|\hat{V}|\text{HF}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{V} + [V,T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|[\hat{V},T_2]|\text{HF}\rangle\Bigg) \quad , \end{aligned} \tag{10.15}$$

where $\Re$ indicates that the real part is taken. *Relaxed* CC2 properties (and gradients) are calculated from the the full variational density including the contributions from the orbital response to the external perturbation, which are derived from the Lagrangian [14, 237]

$$\begin{aligned} L^{\text{rel CC2}}(t,\bar{t}) &= \langle \text{HF}|H|\text{CC}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{H} + [\hat{H},T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|\hat{H} + [F,T_2]|\text{HF}\rangle + \sum_{\mu_0} \bar{\kappa}_{\mu_0} F_{\mu_0} \quad , \end{aligned} \tag{10.16}$$

where F is the Fock operator corresponding to the Hamiltonian of the perturbed system $H = H_0 + \beta V$. One-electron properties are then obtained as:

$$\begin{aligned} \langle V\rangle^{\text{rel CC2}} &= \Re\Bigg(\langle \text{HF}|\hat{V}|\text{HF}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{V} + [V,T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|[V,T_2]|\text{HF}\rangle + \sum_{\mu_0} \bar{\kappa}_{\mu_0} V_{\mu_0}\Bigg) \quad , \end{aligned} \tag{10.17}$$

$$= \sum_{pq} D^{\text{rel}}_{pq}\, V_{pq} \quad . \tag{10.18}$$

The calculation of one-electron first-order properties requires that in addition to the cluster equations also the linear equations for the Lagrangian multipliers $\bar{t}_\mu$ are solved, which

requires similar resources (CPU, disk space, and memory) as the calculation of a single excitation energy. For orbital-relaxed properties also a CPHF-like linear equation for the Lagrangian multipliers $\bar{\kappa}_{\mu_0}$ needs to be solved and the two-electron density has to be build, since it is needed to set up the inhomogeneity (right-hand side). The calculation of relaxed properties is therefore somewhat more expensive—the operation count for solving the so-called Z-vector equations is similar to what is needed for an SCF calculation—and requires also more disk space to keep intermediates for the two-electron density—about $O(2V+2N)N_x+N_x^2$ in addition to what is needed for the solution of the cluster equations. For ground states, orbital-relaxed first-order properties are standard in the literature.

The calculation of the gradient implies the calculation of the same variational densities as needed for relaxed one-electron properties and the solution of the same equations. The construction of the gradient contributions from the densities and derivative integrals takes about the same CPU time as 3–4 SCF iterations and only minor extra disk space. For details of the implementation of CC2 relaxed first-order properties and gradients and a discussion of applicability and trends of CC2 ground-state equilibrium geometries see ref. [14]. The following is in example input for a MP2 and CC2 single point calculation of first-order properties and gradients:

```
$ricc2
  mp2
  cc2
$response
  static relaxed operators=diplen,qudlen
  gradient
```

A different input is required for geometry optimizations: in this case the model for which the geometry should be optimized must be specified in the data group `$ricc2` by the keyword `geoopt`:

```
$ricc2
  mp2
  cc2
  geoopt model=cc2
```

For CC2 calculations, the single-substitution part of the Lagrangian multipliers $\bar{t}_\mu$ are saved in the file `CCL0--1--1---0` and can be kept for a restart (for MP2 and CCS, the single-substitution part $\bar{t}_\mu$ vanishes).

For MP2 only relaxed first-order properties and gradients are implemented (unrelaxed MP2 properties are defined differently than in CC response theory and are not implemented). For MP2, only the CPHF-like Z-vector equations for $\bar{\kappa}_{\mu_0}$ need to be solved, no equations have to be solved for the Lagrangian multipliers $\bar{t}_\mu$. CPU time and disk space requirements are thus somewhat smaller than for CC2 properties or gradients.

For SCF/CIS/CCS it is recommended to use the modules `grad` and `rdgrad` for the calculation of, ground state gradients and first-order properties.

10.3.2 Excited State Properties, Gradients and Geometries

Also for excited states presently unrelaxed and relaxed first-order properties are available in the `ricc2` program. These are implemented for CCS and CC2. Note, that in the unrelaxed case CIS and CCS are *not* equivalent for excited-states first-order properties and no first-order properties are implemented for CIS in the `ricc2` program.

Orbital-unrelaxed first-order properties

The unrelaxed first-order properties are calculated from the variational excited states Lagrangian [239], which for the calculation of unrelaxed properties is composed of the unrelaxed ground state Lagrangian, Eq. (10.13), and the expression for the excitation energy:

$$\begin{aligned} L^{\text{ur CC2, ex}}(E,\bar{E},t,\bar{t}^{(ex)},\beta) &= \langle \text{HF}|H|\text{CC}\rangle + \sum_{\mu\nu}\bar{E}_\mu \mathbf{A}_{\mu\nu}(t,\beta)E_\nu \\ &+ \sum_{\mu_1}\bar{t}^{(ex)}_{\mu_1}\langle\mu_1|\hat{H}+[\hat{H},T_2]|\text{HF}\rangle \\ &+ \sum_{\mu_2}\bar{t}^{(ex)}_{\mu_2}\langle\mu_2|\hat{H}+[F_0+\beta\hat{V},T_2]|\text{HF}\rangle \end{aligned} \tag{10.19}$$

where it is assumed that the left and right eigenvectors are normalized such that $\sum_{\mu\nu}\bar{E}_\mu\langle\mu|\tau_\nu\rangle E_\nu = 1$ and $H = H_0 + \beta V$. The first-order properties are calculated as first derivatives of $L^{\text{ur CC2, ex}}(E,\bar{E},t,\bar{t}^{(ex)},\beta)$ with respect to the field strength β and are evaluated via a density formalism:

$$\langle V\rangle^{\text{ur,ex}} = \mathcal{R}\left(\frac{\partial L^{ur,ex}(E,\bar{E},t,\bar{t}^{(ex)},\beta)}{\partial\beta}\right)_0 = \sum_{pq} D^{\text{ur,ex}}_{pq}V_{pq} \quad , \tag{10.20}$$

(Again $\Re$ indicates that the real part is taken.) The unrelaxed excited-state properties obtained thereby are related in the same way to the total energy of the excited states as the unrelaxed ground-state properties to the energy of the ground state and the differences between excited- and ground-state unrelaxed properties are identical to those identified from the second residues of the quadratic response function. For a detailed description of the theory see refs. [237, 239]; the algorithms for the RI-CC2 implementation are described in refs. [13, 235]. ref. [235] also contains a discussion of the basis set effects and the errors introduced by the RI approximation.

The calculation of excited-state first-order properties thus requires the calculation of both the right (E_μ) and left ($\bar{E}_\mu$) eigenvectors and of the excited state Lagrangian multipliers $\bar{t}^{(ex)}_\mu$. The disk space and CPU requirements for solving the equations for $\bar{E}_\mu$ and $\bar{t}^{(ex)}_\mu$ are about the same as those for the calculation of the excitation energies. For the construction of the density matrices in addition some files with $\mathcal{O}(n_{root}N^2)$ size are written, where n_{root} is the number of excited states.

The single-substitution parts of the excited-states Lagrangian multipliers $\bar{t}^{(ex)}_\mu$ are saved in files named `CCNL0-`*s*`--`*m*`-`*xxx*.

For the calculation of first-order properties for excited states, the keyword `exprop` must be added with appropriate options to the data group `$excitations`; else the input is same as for the calculation of excitation energies:

```
$ricc2
  cc2
$response
  fop unrelaxed_only operators=diplen,qudlen
$excitations
  irrep=a1 nexc=2
  exprop states=all operators=diplen,qudlen
```

Orbital-relaxed first-order properties and gradients

To obtain orbital-relaxed first-order properties or analytic derivatives (gradients) the Lagrange functional for the excited state in Eq. (10.19) is—analogously to the treatment of ground states—augmented by the equations for the SCF orbitals and the perturbations is also included in the Fock operator:

$$\begin{aligned} L^{\text{rel CC2, ex}}(E,\bar{E},t,\bar{t}^{(ex)},\beta) &= \langle \text{HF}|H|\text{CC}\rangle + \sum_{\mu\nu}\bar{E}_\mu \mathbf{A}_{\mu\nu}(t,\beta)E_\nu \\ &\quad + \sum_{\mu_1}\bar{t}^{(ex)}_{\mu_1}\langle\mu_1|\hat{H}+[\hat{H},T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2}\bar{t}^{(ex)}_{\mu_2}\langle\mu_2|\hat{H}+[F,T_2]|\text{HF}\rangle + \sum_{\mu_0}\bar{\kappa}^{(ex)}_{\mu_0}F_{\mu_0} \quad . \end{aligned} \tag{10.21}$$

Compared to unrelaxed properties, the calculation of relaxed properties needs in addition for each excited state the solution of a CPHF equations for the Lagrangian multipliers $\bar{\kappa}^{(ex)}_{\mu_0}$, for which the computational costs are similar to those of a Hartree-Fock calculation.

Orbital-relaxed properties are requested by adding the flag `relaxed` to the input line for the `exprop` option. The following is an example for a CC2 single point calculation for orbital-relaxed excited state properties:

```
$ricc2
  cc2
$excitations
  irrep=a1 nexc=2
  exprop states=all relaxed operators=diplen,qudlen
```

Note that during the calculation of orbital-relaxed excited-state properties the corresponding unrelaxed properties are also automatically evaluated at essentially no additional costs. Therefore, the calculation of unrelaxed properties can not be switched off when relaxed properties have been requested.

Again the construction of gradients requires the same variational densities as needed for relaxed one-electron properties and the solution of the same equations. The construction of

the gradient contributions from one- and two-electron densities and derivative integrals takes approximately the same time as for ground states gradients (approx. 3–4 SCF iterations) and only minor extra disk space. The implementation of the excited state gradients for the RI-CC2 approach is described in detail in Ref. [15]. There one can also find some information about the performance of CC2 for structures and vibrational frequencies of excited states.

For the calculation of an excited state gradient with CC2 at a single point (without geometry optimization and if it is not a calculation with `NumForce`) one can use the input:

```
$ricc2
  cc2
$excitations
  irrep=a1 nexc=2
  xgrad states=(a1 1-2)
```

For geometry optimizations or a numerical calculation of the Hessian with `NumForce` the wavefunction model and the excited state for which the geometry should be optimized have to be specified in the data group `$ricc2` with the keyword `geoopt`:

```
$ricc2
  geoopt model=cc2 state=(a1 2)
$excitations
  irrep=a1 nexc=2
```

If the geometry optimization should carried out for the lowest excited state (of those for which an excitation energy is requested in `$excitation`), one can use alternatively `state=(s1)`.

Since the calculation of unrelaxed and relaxed first-order properties can be combined gradient calculations without significant extra costs, a request for excited state gradients will automatically enforce the calculation of the relaxed and unrelaxed dipole moments. If the keyword `geoopt` is used, the relaxed dipole moment for the specified excited state and wavefunction model will be written to the `control` file and used in calculations with `NumForce` for the evaluation of the IR intensities.

10.3.3 Visualization of densities and Density analysis

As most other programs which allow for the calculation of wavefunctions and densities also the `ricc2` module is interfaced to wavefunction analysis and visualization toolbox described in chapter 20. From `ricc2` module this interface can used in two different ways

1. If through the `geoopt` keyword in `$ricc2` a unique method and state has been specified for which the density, gradient and properties are evaluated, the density analysis and visualization routines will called by default with the (orbital-relaxed) density for this state and method similar as in `dscf`, `ridft`, `mpgrad`, etc.

2. The `ricc2` program can be called in a special analysis mode which allows to analyse densities and combination (e.g. differences) of densities evaluated in preceeding `ricc2` calculations.

Default density analysis and visualization:

As in a single calculations with the `ricc2` program one–electron densities can be calculated for more than one method and/or electronic state, the interface to the analysis and visualization routines require the specification of a unique level of calculation and a unique state. This is presently done through the `geoopt` flag which determines the method/state for which results are written to interface files (e.g. `control`, `gradient`, or `xxx.map`).

In ground state calculations `ricc2` will pass to the density analysis routines the correlated total (and for UHF based calculations also the spin) density and the canonical SCF orbitals from which the SCF (spin) density is constructed. All options described in chapter 20 are available from within the `ricc2` program apart from the evaluation of electrostatic moments, which would interfere with the calculation of expectation values requested through the `fop` option in `$response`.

In excited state calculation `ricc2` will pass the excited state total (and for UHF based calculation in addition the spin) density. But no ground state densities and/or uncorrelated densities or orbitals. Thus, for excited states the `ricc2` program does, in difference to `egrad` not print out a comparison with the ground state SCF density. Also, all some options which require orbitals (as e.g. the generation and visualization of localized orbitals or some population analysis options) and not available for excited states in `ricc2`.

As other modules, also `ricc2` provides the `-proper` flag to bypass a re-calculation of the density and gradient to enter immediately the density analysis routines with a previously calculated density. The `ricc2` program will then pass the densities found on the interface file for the density analysis routines without further check on the method and state for which they have been evaluated. If both, ground and excited state densities are found on file, both will be passed to the density analysis, thereby providing a shortcut to the `-fanal` and the `$anadens` keyword for the analysis of differences between ground and excited state densities.

The general density analysis option:

In general `ricc2` saves by default all *relaxed* densities generated during a calculation in files named `<method>-<type>-<mult><irrep>-<number>-total.cao` `<method>` denotes the level of theory, like `mp2`, `cc2`, or `adcp2`. `<type>` is one of `gsdn` (ground state) or `xsdn` (excited state) the other entries specify multiplicity, irreducible representation and the number of the state. Having specified the calculation of relaxed densities—e.g. by requesting relaxed one-electron properties or as a by-product of a gradient calculation—you will end up with two files named like

```
cc2-gsdn-1a1-001-total.cao
cc2-xsdn-3a2-001-total.cao
```

In case of open shell molecules, additional files with names `...-spndn.cao` (for one-electron spin-densities) will be generated.

These files are (currently) in a binary format, similar as the files `dens`, `mdens` and `edens`. Therefore be aware that a transfer between different computer architectures may result in trouble.

The densities on these files can be analysed with the tools and interfaces provided by Moloch (see Section 20.2). This can be done by calling `ricc2` with the option `-fanal` which bypasses the usual wavefunction calculation and triggers the program into an analysis mode for densities. In this mode the program interpretes `$anadens` and the keywords described in Section 20.2. To plot, for example, the difference density of the two above mentioned total densities you have to add the following lines in your `control` file

```
$anadens
 calc my_favourite_diffden from
 1d0 cc1td-cc2-xs-3a2-001
-1d0 cc1td-cc2-gs-1a1-001
$pointval
```

and invoke

```
  ricc2 -fanal
```

This will generate the files `my_favourite_diffden` and `my_favourite_diffden.map`. The latter can be converted into gOpenMol format as described in Section 20.2.

10.3.4 Fast geometry optimizations with RI-SCF based gradients

If geometry optimizations on MP2 or CC2 level are performed with large basis set, especially with diffuse basis functions, the N^4–steps might become the dominant part of the overall timings. In these cases, the integral screening in the Hartree–Fock part often becomes inefficient. The resolution–of–the–identity can be applied here to speed up the calculation of the HF reference wavefunction, as well as the solution of the coupled–perturbed Hartree–Fock (CPHF) equations in the MP2 or CC2 gradient calculation.

An additional auxiliary basis (denoted `jkbas`) set has to be assigned via the General Options Menu in the `define` program. In the submenu `rijk` choose `on` and select your auxiliary basis set. Then, run the `jobex` script the additional rijk-flag:

```
> jobex -level cc2 -rijk
```

10.4 Transition Moments

Transition moments (for one-photon transitions) are presently implemented for excitations out of the ground state and for excitations between excited states for the coupled cluster

models CCS and CC2. Transition moments for excitations from the ground to an excited state are also available for ADC(2), but use an additional approximation (see below). Note, that for transition moments (as for excited-state first-order properties) CCS is *not* equivalent to CIS and CIS transition moments are not implemented in the `ricc2` program.

Two-photon transition moments and induced transition moments, e.g. for phosphorescence or magnetic circular dichroism (MCD), are only available for CC2.

10.4.1 Ground to excited state transition moments

In response theory, transition strengths (and moments) for transitions from the ground to excited state are identified from the first residues of the response functions. Due to the non-variational structure of coupled cluster different expressions are obtained for the CCS and CC2 "left" and "right" transitions moments $M^V_{0\leftarrow f}$ and $M^V_{f\leftarrow 0}$. The transition strengths $S^{0f}_{V_1V_2}$ are obtained as a symmetrized combinations of both [240]:

$$S^{0f}_{V_1V_2} = \frac{1}{2}\left\{M^{V_1}_{0\leftarrow f}M^{V_2}_{f\leftarrow 0} + \left(M^{V_2}_{0\leftarrow f}M^{V_1}_{f\leftarrow 0}\right)^*\right\} \tag{10.22}$$

Note, that only the transition strengths $S^{0f}_{V_1V_2}$ are a well-defined observables but not the transition moments $M^V_{0\leftarrow f}$ and $M^V_{f\leftarrow 0}$. For a review of the theory see refs. [237, 240]. The transition strengths calculated by coupled-cluster response theory according to Eq. (10.22) have the same symmetry with respect to an interchange of the operators V_1 and V_2 and with respect to complex conjugation as the exact transition moments. In difference to SCF (RPA), (TD)DFT, or FCI, transition strengths calculated by the coupled-cluster response models CCS, CC2, etc. do not become gauge-independent in the limit of a complete basis set, i.e., for example the dipole oscillator strength calculated in the length, velocity or acceleration gauge remain different until also the full coupled-cluster (equivalent to the full CI) limit is reached.

For a description of the implementation in the `ricc2` program see refs. [14, 235]. The calculation of transition moments for excitations out of the ground state resembles the calculation of first-order properties for excited states: In addition to the left and right eigenvectors, a set of transition Lagrangian multipliers $\bar{M}_\mu$ has to be determined and some transition density matrices have to be constructed. Disk space, core memory and CPU time requirements are thus also similar.

The single-substitution parts of the transition Lagrangian multipliers $\bar{N}_\mu$ are saved in files named `CCME0-`*s*`--`*m*`-`*xxx*.

To obtain the transition strengths for excitations out of the ground state the keyword `spectrum` must be added with appropriate options (see Section 23.2.22) to the data group `$excitations`; else the input is same as for the calculation of excitation energies and first-order properties:

```
$ricc2
  cc2
```

```
$excitations
  irrep=a1 nexc=2
  spectrum states=all operators=diplen,qudlen
```

For the ADC(2) model, which is derived by a perturbation expansion of the expressions for exact states, the calculation of transition moments for excitations from the ground to an excited state would require the second-order double excitation amplitudes for the ground state wavefunction, which would lead to operation counts scaling as $\mathcal{O}(\mathcal{N}^6)$, if no further approximations are introduced. On the other hand the second-order contributions to the transition moments are usually not expected to be important. Therefore, the implementation in the `ricc2` program neglects in the calculation of the ground to excited state transition moments the contributions which are second order in ground state amplitudes (i.e. contain second-order amplitudes or products of first-order amplitudes). With this approximation the ADC(2) transition moments are only correct to first-order, i.e. to the same order to which also the CC2 transition moments are correct, and are typically similar to the CC2 results. The computational costs for the ADC(2) transition moments are (within this approximation) much lower than for CC2 since the left and right eigenvectors are identical and no Lagrangian multipliers need to be determined. The extra costs (i.e. CPU and wall time) for the calculations of the transitions moments are similar to the those for two or three iterations of the eigenvalue problem, which reduces the total CPU and wall time for the calculation of a spectrum (i.e. excitation energies and transition moments) by almost a factor of three.

10.4.2 Transition moments between excited states

For the calculation of transition moments between excited states a set of Lagrangian multipliers $\bar{N}_\mu$ has to be determined instead of the $\bar{M}_\mu$ for the ground state transition moments. From these Lagrangian multipliers and the left and right eigenvectors one obtaines the "right" transition moment between two excited states i and f as

$$M^V_{f\leftarrow i} = \sum_{pq} \left\{ D^\xi_{pq}(\bar{N}^{fi}) + D^A_{pq}(\bar{E}^f, E^i) \right\} \hat{V}_{pq}. \tag{10.23}$$

where $\hat{V}$ are the matrix elements of the perturbing operator. A similar expression is obtained for the "left" transition moments. The "left" and "right" transition moments are then combined to yield the transition strength

$$S^{if}_{V_1V_2} = \frac{1}{2} \left\{ M^{V_1}_{i\leftarrow f} M^{V_2}_{f\leftarrow i} + \left(M^{V_2}_{i\leftarrow f} M^{V_1}_{f\leftarrow i} \right)^* \right\} \tag{10.24}$$

As for the ground state transitions, only the transition strengths $S^{if}_{V_1V_2}$ are a well-defined observables but not the transition moments $M^V_{i\leftarrow f}$ and $M^V_{f\leftarrow i}$.

The single-substitution parts of the transition Lagrangian multipliers $\bar{N}_\mu$ are saved in files named `CCNE0-`*s*`--`*m*`-`*xxx*.

To obtain the transition strengths for excitations between excited states the keyword `tmexc` must be added to the data group `$excitations`. Additionally, the initial and final states

must be given in the same line; else the input is same as for the calculation of excitation energies and first-order properties:

```
$ricc2
  cc2
$excitations
  irrep=a1 nexc=2
  irrep=a2 nexc=2
  tmexc istates=all fstates=(a1 1) operators=diplen
```

10.4.3 Ground to excited state two-photon transition moments

For closed-shell restricted and high-spin open-shell unrestricted Hartree-Fock reference states two-photon transition moments for one-electron operators can be computed at the CCS and the CC2 level. The implementation is restricted to real Abelian point groups (D_{2h} and its subgroups) and for CC2 without spin-component scaling. The response theory for the calculation of two-photon transition moments with coupled cluster methods has been described in [241, 242], the RI-CC2 implementation in [243].

Similar as for one-photon transitions also for two-photon transitions the non-variational structure of coupled cluster theory leads to different expressions for "left" and "right" transition moments $M^{V_1V_2}_{0\leftarrow f}$ and $M^{V_3V_4}_{f\leftarrow 0}$. Only the transition strengths $S^{0f}_{V_1V_2,V_3V_4}$ which are obtained as symmetrized combination of both [240, 242],

$$S^{0f}_{V_1V2,V_3V_4}(\omega) = \frac{1}{2}\left\{M^{V_1V_2}_{0\leftarrow f}(-\omega)M^{V_3V_4}_{f\leftarrow 0}(\omega) + \left(M^{V_3V_4}_{0\leftarrow f}(-\omega)M^{V_1V_2}_{f\leftarrow 0}(\omega)\right)^*\right\} \tag{10.25}$$

is an observable quantity.

The `ricc2` program prints in the output in addition to the transition moments in atomic units also the rotationally averaged transition strengths in atomic units and transition rates in Göppert-Maier (GM) units for linear, perpendicular and circular polarized beams.

In addition to the input for the excitation energies, the computation of two-photon transition moments requires the keyword `twophoton` in the data group `$excitations` and the data group for the numerical Laplace transformation `$laplace`.

```
ricc2
  cc2
$laplace
  conv=6
$excitations
  irrep=a1 nexc=1
  irrep=b2 nexc=1
  twophoton states=all operators=(diplen,diplen) freq=0.1d0
```

The syntax for the input for `states` is the same as for one-photon transition moments (option `spectrum`). Frequencies ω_2 for the photon associated in the "right" transition moment,

$M^{V_1V_2}_{f\leftarrow 0}(\omega_2)$, with the second operator in each operator pair can be given in atomic units with the suboption `freq`. It corresponds to the frequency ω in Eq. 10.25. The frequency associated with the first operator is automatically determined from the sum rule $\omega_1 = \omega_{0f} - \omega_2$, where ω_{0f} is the frequency for the transition $f \leftarrow 0$. The "left" transition moments $M^{V_1V_2}_{0\leftarrow f}$ are computed for the frequencies with the opposite sign. If not specified, the two-photon transition moments are computed for the case that both photons have the same frequency, i.e. their energies are set for each state to 1/2 of the transition energy, which is the most common case.

10.4.4 Induced ground–to–excited state transition moments

The keyword `momdrv` in the `$excitation` data group

```
momdrv states=(b2{3} 1) operators=(diplen,soc)
```

initiates the calculation of first derivatives of the one-photon transition moments for the first operator (here the length form of electric dipole operator `diplen`) with respect to a time-independent perturbation by the second operator (here spin-orbit coupling). Note, that the order of the operators in the pair `(diplen,soc)` matters and that the `momdrv` keyword requires that the data group `$laplace` is set in the input.

The implementation in the `ricc2` program is based on the fully phase-isolated formulation parameterization of the eigenvectors [239] and its implementation for doubles-direct RI-CC2 described in Ref. [244, 245].

Phosphorescence lifetimes using SOC-PT-CC2

For the calculation of oscillator strengths for triplet excited states and phosphorescence lifetimes for molecules without heavy atoms the `momdrv` keyword provides as alternative to the relativistic two-component variant a perturbative SOC-PT-CC2 approach which works with one-component wavefunctions. In SOC-PT-CC2 the oscillator strengths for triplet excited states are calculated as first derivatives of the (spin-forbidden) oscillator strengths for one-component wavefunctions with respect to the strength of the spin-orbit coupling (SOC) in the limit of zero spin-orbit coupling. The spin-orbit coupling is treated within the effective spin-orbit mean field approximation, where the mean field two-electron contribution is computed from the Hartree-Fock density. The SOC-PT-CC2 approach is about an order of magnitude faster than the computation of oscillator strengths with the two-component RI-CC2 variant. Scalar relativistic effects from the spin-free X2C Hamiltonian can be included by using the spin-free X2C Hamiltonian for the Hartree-Fock reference wavefunction. The theory and the implementation are described in [245].

The implementation is restricted to closed-shell Hartree-Fock reference wavefunctions and CC2 without spin-component scaling. For carrying out such calculations one needs in addition to the input for the excitation energies for triplet states the keyword `momdrv` and the data group for the numerical Laplace transformation `$laplace`.

```
$ricc2
  cc2
$laplace
  conv=6
$excitations
  irrep=b2 multiplicity=3 nexc=1
  momdrv states=(b2{3} 1) operators=(diplen,soc)
```

For phosphorescence lifetimes the operator pair has to be set to `(diplen,soc)` for the dipole operator and the SOC operator in the SOMF approximation, so that a frequency of `0.0d0` is associated with the SOC operator and the ground to excited state transition frequency ω_{0f} with the dipole operator.

The program prints in the output in addition to the first-order induced transition moments for the operator pair `(diplen,soc)` the induced oscillator strengths and the life times.

Perturbation-induced circular dischroism

It is possible to calculate the Faraday $\mathcal{A}$ and $\mathcal{B}$ terms for the magnetic circular dichroism (MCD) of a molecule as well as the $\mathcal{B}_K$ term for nuclear spin-induced circular dichroism (NSCD). The $\mathcal{A}$ term is only present for degenerate states, the presence of which is automatically detected by the `ricc2` program, as long as both states are included in the `momdrv` keyword:

```
$ricc2
  cc2
$laplace
  conv=6
$excitations
   irrep=b2u nexc=1
   irrep=b3u nexc=1
   momdrv states=(b2u 1;b3u 1) operators(diplen,angmom)
```

For the computation of the NSCD $\mathcal{B}_K$ term, the operator combination is `(diplen, psoXXXX)` where XXXX is the nucleus' number as it appears in the `coord` file, i.e., for the first atom it is pso0001:

```
$ricc2
  cc2
$laplace
  conv=6
$excitations
   irrep=a   nexc=2
   momdrv states=all operators(diplen,pso0001)
```

`pso` denotes the paramagnetic (nuclear) spin-(electron) orbit (PSO) operators. Similar as for other rank-1 tensor operators it comprises the set of all three cartesian components (x, y, z) if not an individual component is specified. **Note:** The current implmenentation uses the cartesian compoments of the PSO operators for individual atoms directly without any symmetry-adaption and is, thus, not compatible with the use of point group symmetry.

10.5 Ground State Second-order Properties with MP2 and CC2

Second-order properties for one-electron perturbations can be computed at the MP2 and the CC2 level. For MP2, second-order properties are computed as derivatives of the SCF+MP2 total energy. This approach includes the relaxation of the SCF orbitals in the presence of the perturbation and is restricted to the static (i.e. frequency-independent) limit.

For the coupled-cluster model CC2, second-order properties can, similar as the first-order properties, be computed in the orbital-unrelaxed or the orbital-relaxed approach as derivatives of the of the Lagrange functions in Eqs. 10.13 and 10.16. As for MP2, the orbital-relaxed calculations are restricted to the static limit. Frequency-dependent second-order properties as e.g. dipole polarizabilities can be computed with the orbital-unrelaxed approach.

Second-order properties are available in the OMP and MPI parallel versions and for closed-shell and unrestricted high-spin open-shell Hartree-Fock refrences. Note, that second-order properties are not available for spin-component scaled variants of MP2 and CC2 or for restricted open-shell references. Furthermore, non-Abelian point groups and point groups with complex irreducible representations are not implemented for second-order properties.

In addition to the standard input, second-order properties require that the data group for the numerical Laplace transformation `$laplace` and that the `sops` option in the data group `$response` is set. Frequency-dependent dipole polarizabilities with the CC2 model are obtained with the input:

```
$ricc2
  cc2
$laplace
  conv=4
$response
  sop operators=(diplen,diplen) freq=0.077d0
```

The frequency has to be given in atomic units. Static orbital-relaxed polarizabilities are obtained with

```
$response
  sop operators=(diplen,diplen) relaxed
```

10.5.1 Damped Second-order Properties with CC2

Damped linear response is available at the CC2 level. In addition to the second-order property input, the flag `cpp=on` must be set along with a damping factor `gamma`, in atomic units. The following input will compute the damped polarizability with $\gamma = 4.5563{\cdot}10^{-3}$ a.u.:

```
$response
  cpp=on gamma=4.5563d-3
  sop operators=(diplen,diplen) freq=0.077d0
```

using the symmetric expression for the linear response function. To compute a second-order property from the asymmetric expression, the keyword `asym` must be added to the `sops` option:

```
$response
  cpp=on gamma=4.5563d-3
  sop asym operators=(diplen,diplen) freq=0.077d0
```

Since damped response is often used to compute spectra in a frequency interval on an even-spaced grid. This can be done with the following simplified input for the interval $[0.4, 0.5]$ with a step width of 0.005:

```
  cpp=on gamma=4.5563d-3
  sop operators=(diplen,diplen) frqscan=0.3d0, 0.4d0, 0.005d0
```

The options `abscpp` and `ecdcpp` can be used with the `sop` keyword as shortcuts to request the operator combinations that are needed to compute the isotropic avarages required for, respectively, absorption spectra in the length and ECD spectra in the velocity gauge.

```
  cpp=on gamma=4.5563d-3
  sop abscpp ecpcpp frqscan=0.3d0, 0.4d0, 0.005d0
```

10.6 Ground State third-order Properties with CC2

Third-order properties or quadratic response functions for one-electron perturbations are only implemented for CC2 using the orbital-unrelaxed response formalism. The implementation is currently limited to a closed-shell Hartree-Fock reference and the sequential and OMP-parallel versions of the program. Note that second-order properties are also not yet available for the spin-component scaled variants of CC2. Furthermore, non-Abelian points groups and point groups with complex irreducible representations are not implemented for third-order properties.

The input for third-order properties is similar as for second-order properties: In addition to the standard input for the CC2 ground-state calculation, they require that the data group for the numerical Laplace transformation `$laplace` is set. Third-order ground-state properties are requested with the `top` option in the data group `$response`:

```
$ricc2
  cc2
$laplace
  conv=4
$response
  top operators=(diplen,diplen,diplen) freq1=0.077d0 freq2=-0.077d0
```

The above input will request the full dipole-dipole-dipole tensor for the frequency-dependent hyperpolarizability $\beta_{ijk}(0.077, -0.077, 0.0)$. The frequencies specified with `freq1` are associated to the first and those specified with `freq2` with the second operator in the operator triple specified with `operators`. The frequencies have to be given in atomic units. If not given, a value of 0.0 is used. The frequencies for the third operator are set to the negative of the sum of the frequencies for the first two operators. In general, the `operators` option can be followed by a comma-separated list of operator triples and `freq1` and `freq2` by comma-separated lists of frequencies, e.g.

```
  top operators=(xdiplen,xdiplen,xdiplen),(zdiplen,zdiplen,zdiplen)
                    freq1=0.077d0,0.088d0 freq2=-0.077d0,0.088d0
```

(without linebreak!) will request the four hyperpolarizability vaues $\beta_{xxx}(0.077, -0.077, 0.0)$, $\beta_{zzz}(0.077, -0.077, 0.0)$, $\beta_{xxx}(0.088, 0.088, 0.176)$, and $\beta_{zzz}(0.088, 0.088, 0.176)$. Note that the i-th frequency in the list `freq1` is (only) combined with the i-th frequency in the list `freq2`, while each operator triple is combined with each frequency combination.

10.6.1 Damped Third-order Properties with CC2

Damped quadratic response is available at the CC2 level. In addition to the third-order property input, the flag `cpp=on` must be set along with a damping factor `gamma`, in atomic units. The following input will compute the damped hyperpolarizability with $\gamma = 4.5563 \cdot 10^{-3}$ a.u.:

```
$response
  cpp=on gamma=4.5563d-3
  top operators=(diplen,diplen,diplen) freq1=0.077d0 freq2=-0.077d0
```

As for damped second-order properties, it is possible to set a frequency interval in the input

```
$response
  cpp=on gamma=4.5563d-3
  top operators=(diplen,diplen,diplen) frqscan=0.3d0, 0.4d0, 0.005d0
```

To build an MCD spectrum using damped response over a grid of frequencies, use the `mcdcpp` option along with `frqscan`

```
$response
  cpp=on gamma=4.5563d-3
  top mcdcpp frqscan=0.3d0, 0.4d0, 0.005d0
```

The operator combination for the MCD property is `(diplen,diplen,angmom)`.

It is also possible to compute nuclear spin-induced CD (NSCD) from damped quadratic response. For this, the operator combination is `(diplen,diplen,psoXXXX)` where `XXXX` is the nucleus' number as it appears in the coord file, i.e., the PSO operator of the first atom is `pso0001`. Its damped NSCD spectrum is build from the input

```
$response
  cpp=on gamma=4.5563d-3
  top operators=(diplen,diplen,pso0001) frqscan=0.3d0, 0.4d0, 0.005d0
```

10.7 Parallel RI-MP2 and RI-CC2 Calculations

All functionalities of the `ricc2` program are available in the OpenMP-based parallel version for shared memory (SMP) architectures. Most functionalities of the `ricc2` program are also parallelized for distributed memory architectures (e.g. clusters of Linux boxes) based on the *message passing interface* (MPI) standard.

While in general the parallel execution of `ricc2` works similar to that of other parallelized TURBOMOLE modules as e.g. `dscf` and `grad`, there are some important difference concerning in particular the handling of the large scratch files needed for RI-CC2 (or RI-MP2). As the parallel version `dscf` also the parallel version of `ricc2` assumes that the program is started in a directory which is readable (and writeable) on all compute nodes under the same path (e.g. a NFS directory). The directory must contain all input files and will at the end of a calculation contain all output files. Large scratch files (e.g. for integral intermediates) will be placed under the path specified in the `control` file with `$tmpdir` (see Section 23.2.22) which should point to a directory in a file system with a good performance. All large files will be placed on the nodes in these file systems. (The local file system must have the same name on all nodes.) Note that at the end of a `ricc2` run the scratch directories specified with `$tmpdir` are not guaranteed to be empty. To avoid that they will fill your file system you should remove them after the `ricc2` calculation is finished.

Another difference to the parallel HF and DFT (gradient) programs is that `ricc2` will communicate much larger amounts of data between the compute nodes. With a fast network interconnection (Gigabit or better) this should not cause any problems, but with slow networks the communication might become the limiting factor for performance or overloading the system. If this happens the program can be put into an alternative mode where the communication of integral intermediates is replaced by a reevaluation of the intermediates (at the expense of a larger operation count) wherever this is feasible. Add for this in the `control` the following data group:

```
$mpi_param
```

```
min_comm
```

10.8 Spin-component scaling approaches (SCS/SOS)

By introducing individual scaling factors for the same–spin and opposite–spin contributions to the correlation energy most second–order methods can be modified to achieve a (hopefully) better performance. SCS-MP2 has first been proposed by S. Grimme and SOS-MP2 by Y. Jung *et al.* (see below). The generalization of SCS and SOS to CC2 and ADC(2) for ground and excited states is described in [17]. It uses the same scaling factors as proposed for the original SCS- and SOS-MP2 approaches (see below). In the `ricc2` program we have also implemented SCS and SOS variants of CIS(D) for excitation energies and of CIS(D_∞) for excitation energies and gradients, which are derived from SCS-CC2 and SOS-CC2 in exactly the same manner as the unmodified methods can be derived as approximations to CC2 (see Sec. 10.2 and Ref. [246]). Please note, that the SCS-CIS(D) and SOS-CIS(D) approximations obtained in this way and implemented in `ricc2` differ from the spin-component scaled SCS- and SOS-CIS(D) methods proposed by, respectively, S. Grimme and E. I. Ugorodina in [247] and Y. M. Rhee and M. Head–Gordon in [248].

A line with scaling factors has to be added in the `$ricc2` data group:

```
$ricc2
scs  cos=1.2d0  css=0.3333d0
```

`cos` denotes the scaling factor for the opposite–spin component, `css` the same–spin component.

As an abbreviation
`scs`
can be inserted in `$ricc2`. In this case, the SCS parameters cos=6/5 and css=1/3 proposed S. Grimme (S. Grimme, *J. Chem. Phys.* **118** (2003) 9095.) are used. These parameters are also recommended in [17] for the SCS variants of CC2, CIS(D), CIS(D_∞), and ADC(2) for ground and excited states.

Also, just
`sos`
can be used as a keyword, to switch to the SOS approach proposed by the Head-Gordon group for MP2 with scaling factors of cos=1.3 and css=0.0 (Y., Jung, R.C. Lochan, A.D. Dutoi, and M. Head-Gordon, *J. Chem. Phys.* **121** (2004) 9793.), which are also recommended for the SOS variants of CC2, CIS(D), CIS(D_∞), and ADC(2). The Laplace-transformed algorithm for the SOS variants are activated by the additional data group `$laplace`:

```
$laplace
   conv=4
```

For further details on the Laplace-transformed implementation and how one can estimated

whether the $\mathcal{O}(\mathcal{N}^4)$-scaling Laplace-transformed or $\mathcal{O}(\mathcal{N}^5)$-scaling conventional RI implementation is efficient see Sec. 9.6.

Since Version 6.6 the $\mathcal{O}(\mathcal{N}^4)$-scaling Laplace-transformed implementation is available for ground and excited state gradients with CC2 and ADC(2).

Restrictions:

- the spin (S^2) expectation value for open-shell calculation can not be evaluated in the SCS or SOS approaches
- for LT-SOS-CC2 (and the related CIS(D) and ADC(2) versions) the following further limitations apply:
  - incompatible with the calculation of the D_1 and D_2 diagnostics
  - second-order properties, two-photon transitions moments and induced transition moments are not implemented for the spin-component scaled variants.

Chapter 11

CCSD, CCSD(F12*) and CCSD(T) calculations

Since release V7.0 the coupled-cluster singles-and-doubles method CCSD and its explicitly-correlated variants CCSD(F12) and CCSD(F12*) are implemented in the `ccsdf12` program. CCSD and the F12 variants can be combined with a perturbative correction for connected triple excitations, CCSD(T).* The Brueckner variants BCCD and BCCD(T) with and without explicit correlation are also available. As perturbative approximations beyond MP2, also the approximations MP3, MP3(F12), MP4, and MP4(F12*) are available. Presently the implementation of the F12 variants and of connected triple excitations is restricted to ground state energies. The CCSD implementation without F12 is limited to ground–state and excitation energies. Closed-shell (RHF), unrestricted (UHF) or single determinant restricted (ROHF) open-shell reference wavefunctions can be used for CCSD and CCSD(T), but no gradients or properties are (yet) available for these wavefunction models. The same is true for BCCD and BCCD(T). The MP3 and MP4 approximations can currently not be combined with ROHF reference wavefunctions.

Further limitations:

no MPI parallelization: calculations at these levels can presently only carried out on a single compute node, only the OpenMP (see Sec. 3.4.2) parallelization is available

use of symmetry restricted: only D_{2h} and its subgroups can be used for conventional (i.e. not F12) calculations; no symmetry can be used for the F12 methods

Please note that calculations with MP3, MP4, CCSD, BCCD and methods beyond CCSD require considerably more disc space and core memory than MP2 or CC2 calculations. (See section below for more details and recommendations.)

*Note that for the explicitly correlated CCSD variants the explicitly-correlated double excitations are neglected for the calculation of the triples corrections as described in Ref. [249].

Prerequisites

MP3, MP4, CCSD and CCSD(T) calculations with the `ccsdf12` module require the same prerequisites as RI-CC2 calculations:

1. a converged SCF calculation with the one-electron density threshold set to `$denconv 1.d-5` or less
2. an auxiliary basis defined in the data group `$cbas`
3. if orbitals should be excluded from the correlation treatment the data group `$freeze` has to be set
4. the maximum core memory which the program is allowed to allocate should be defined in the data group `$maxcor`; the recommended value is 66–75% of the available (physical) core memory. (see Sec. 23.2.3 for details)
5. the data group `$ricc2` with a specification of the coupled-cluster model

The same is true for Brueckner methods. Calculations with the CCSD(F12*), CCSD(F12) and $\text{BCCD}_{(\text{F12}*)}$ methods require in addition:

- the data group `$rir12` with the definition of the standard approximations for the explicitly-correlated contributions (see Sec. 9.5 for details)
- the data group `$lcg`, which define the correlation function (here it is in particular important to choose for F12 calculations the exponent; recommended values are 0.9 for cc-pVDZ-F12, 1.0 for cc-pVTZ-F12 and 1.1 for cc-pVQZ-F12 basis sets)
- a complementary auxiliary (CABS) basis set

Furthermore, it is recommended to select in addition an auxiliary JK basis set for the evaluation of the Fock matrix elements. (The `rijk` menu of define can be used for this.)

How To Perform a Calculation

As presently no gradients are available, only single–point calculations are possible:

1. Select in `define` within the menu `cc`

   the wavefunction model (submenu `ricc2`),

   frozen core options (submenu `freeze`),

   an auxiliary basis for `$cbas` (submenue `cbas`),

   the amount of main memory (option `memory`),

   and for F12 calculations in addition

   the F12 options (submenu `f12`), and

a CABS basis (submenu `cabs`).

By default a CCSD(F12*) with ansatz 2 and geminal amplitudes fixed by the cusp conditions is performed.[†]

```
ccsdapprox  ccsd(f12*)
```

The auxiliary JK basis must be chosen in menu `rijk` and the exponent for the correlation function must set by editing the `$lcg` data group of the control file.

2. Do an SCF calculation using either the `dscf` or the `ridft` module.
3. Invoke the `ccsdf12` program on the command line or with a batch script.

How to quote:

- for all F12 calculations cite the implementation of RI-MP2-F12 in TURBOMOLE:
  The MP2-F12 Method in the TURBOMOLE Programm Package. Rafal A. Bachorz, Florian A. Bischoff, Andreas Glöß, Christof Hättig, Sebastian Höfener, Wim Klopper, David P. Tew, J. Comput. Chem. **32**, 2492–2513 (2011).
- for MP3(F12) and CCSD(F12):
  Quintuple-ζ quality coupled-cluster correlation energies with triple-ζ basis sets. David P. Tew, Wim Klopper, Christian Neiss, Christof Hättig, Phys. Chem. Chem. Phys. **9** 921–1930 (2007).
- for MP4(F12*), CCSD(F12*) and CCSD(F12*)(T):
  Accurate and efficient approximations to explicitly correlated coupled-cluster singles and doubles, CCSD-F12. Christof Hättig, David P. Tew, Andreas Köhn, J. Chem. Phys. **132** 231102 (2010).
- for $\mathrm{BCCD}_{(\mathrm{F12}*)}$ and $\mathrm{BCCD(T)}_{(\mathrm{F12}*)}$:
  Explicitly correlated coupled-cluster theory with Brueckner orbitals. David P. Tew, J. Chem. Phys., **145**, 074103 (2016).

11.1 Characteristics of the Implementation and Computational Demands

In CCSD the ground–state energy is (as for CC2) evaluated as

$$E_{\mathrm{CC}} = \langle \mathrm{HF}|H|\mathrm{CC}\rangle = \langle \mathrm{HF}|H\exp(T)|\mathrm{HF}\rangle \quad , \tag{11.1}$$

[†]For other avaible approximation and the corresponding input options see Sec. 23.2.22.

where the cluster operator $T = T_1 + T_2$ consist of linear combination of single and double excitations:

$$T_1 = \sum_{ai} t_{ai}\tau_{ai} \ , \tag{11.2}$$

$$T_2 = \frac{1}{2}\sum_{aibj} t_{ab}^{ij}\tau_{aibj} \ . \tag{11.3}$$

In difference to CC2, the cluster amplitudes t_{ai} and t_{aibj} are determined from equations which contain no further approximations apart from the restriction of T to single and double excitations:

$$\Omega_{\mu_1} = \langle\mu_1|\hat{\bar{H}} + [\hat{\bar{H}}, T_2]|\mathrm{HF}\rangle = 0 \ \ , \tag{11.4}$$

$$\Omega_{\mu_2} = \langle\mu_2|\hat{\bar{H}} + [\hat{\bar{H}}, T_2] + [[\hat{H}, T_2], T_2]|\mathrm{HF}\rangle = 0 \ \ , \tag{11.5}$$

where again

$$\hat{\bar{H}} = \exp(-T_1)\hat{H}\exp(T_1),$$

and μ_1 and μ_2 are, respectively, the sets of all singly and doubly excited determinants. For MP3 the energy is computed from the first-order amplitudes ($t_\mu^{(1)}$) as

$$E_{\mathrm{MP3},tot} = E_{\mathrm{HF}} + E_{\mathrm{MP2}} + E_{\mathrm{MP3}} \tag{11.6}$$

$$= \langle\mathrm{HF}|\hat{H} + [\hat{H}, T_2^{(1)}]|\mathrm{HF}\rangle + \sum_{\mu_2} t_{\mu_2}^{(1)}\langle\mu_2|[\hat{W}, T_2^{(1)}]|\mathrm{HF}\rangle \tag{11.7}$$

with $\hat{W} = \hat{H} - \hat{F}$. To evaluate the fourth-order energy one needs in addition to the first-order also the second-order amplitudes, which are obtained from the solution of the equations

$$\langle\mu_1|[\hat{F}, T_1^{(2)}] + [\hat{W}, T_2^{(1)}]|\mathrm{HF}\rangle = 0 \tag{11.8}$$

$$\langle\mu_2|[\hat{F}, T_2^{(2)}] + [\hat{W}, T_2^{(1)}]|\mathrm{HF}\rangle = 0 \tag{11.9}$$

$$\langle\mu_3|[\hat{F}, T_3^{(2)}] + [\hat{W}, T_2^{(1)}]|\mathrm{HF}\rangle = 0 \tag{11.10}$$

From these the fourth-order energy correction is computed as:

$$E_{\mathrm{MP4}} = \sum_{\mu_2} t_{\mu_2}^{(1)}\langle\mu_2|[\hat{W}, T_1^{(2)} + T_2^{(2)} + T_3^{(2)}] + [[\hat{W}, T_2^{(1)}], T_2^{(1)}]|\mathrm{HF}\rangle \ . \tag{11.11}$$

Eqs. (11.5) and (11.7) – (11.11) are computational much more complex and demanding than the corresponding doubles equations for the CC2 model. If $\mathcal{N}$ is a measure for the system size (e.g. the number of atoms), the computational costs (in terms of floating point operations) for CCSD calculations scale as $\mathcal{O}(\mathcal{N}^6)$. If for the same molecule the number of one-electron basis functions N is increased the costs scale with $\mathcal{O}(\mathcal{N}^4)$. (For RI-MP2 and RI-CC2 the costs scale with the system size as $\mathcal{O}(\mathcal{N}^5)$ and with the number of basis functions as $\mathcal{O}(\mathcal{N}^3)$.) The computational costs for an MP3 calculations are about the same as for one CCSD iteration. For MP4 the computational costs are comparable to those for two CCSD iteration plus the costs for the perturbation triples correction (see below).

Explicitly-correlated CCSD(F12) methods: In explicitly-correlated CCSD calculations the double excitations into products of virtual orbitals, described by the operator $T_2 = \frac{1}{2}\sum_{aibj} t_{ab}^{ij}\tau_{aibj}$, are augmented with double excitations into the explicitly-correlated pairfunctions (geminals) which are described in Sec. 9.5:

$$T = T_1 + T_2 + T_{2'} \tag{11.12}$$

$$T_{2'} = \frac{1}{2}\sum_{ijkl} c_{ij}^{kl}\tau_{kilj} \tag{11.13}$$

where $\tau_{kilj}|ij\rangle = \hat{Q}_{12}f_{12}|kl\rangle$ (for the definition $\hat{Q}_{12}$ and f_{12} see Sec. 9.5). This enhances dramatically the basis set convergence of CCSD calculations ([20]). Without any further approximations than those needed for evaluating the necessary matrix elements, this extension of the cluster operator T leads to the CCSD-F12 method. CCSD(F12) is an approximation ([20, 249]) to CCSD-F12 which neglects certain computationally demanding higher-order contributions of $\hat{T}_{2'}$. This reduces the computational costs dramatically, while the accuracy of CCSD(F12) is essentially identical to that of CCSD-F12 [250, 251]. In the CCSD(F12) approximation the amplitudes are determined from the equations:

$$\Omega_{\mu_1} = \langle\mu_1|\hat{\tilde{H}} + [\hat{\tilde{H}}, T_2 + T_{2'}]|\mathrm{HF}\rangle = 0 \quad , \tag{11.14}$$

$$\Omega_{\mu_2} = \langle\mu_2|\hat{\tilde{H}} + [\hat{\tilde{H}}, T_2 + T_{2'}] + [[\hat{H}, T_2 + 2T_{2'}], T_2]|\mathrm{HF}\rangle = 0 \quad , \tag{11.15}$$

$$\Omega_{\mu_{2'}} = \langle\mu_{2'}|[\hat{F}, T_{2'}] + \hat{\tilde{H}} + [\hat{\tilde{H}}, T_2]|\mathrm{HF}\rangle = 0 \quad . \tag{11.16}$$

Similar as for MP2-F12, also for CCSD(F12) the coefficients for the doubles excitations into the geminals, c_{ij}^{kl} can be determined from the electronic cusp conditions using the rational generator (also known as SP or fixed amplitude) approach. In this case Eq. (11.16) is not solved. To account for this, the energy is then computed from a Lagrange function as:

$$E_{\mathrm{CCSD(F12)-SP}} = L_{\mathrm{CCSD(F12)}} = \langle\mathrm{HF}|H|\mathrm{CC}\rangle + \sum_{\mu_{2'}} c_{\mu_{2'}}\Omega_{\mu_{2'}} \tag{11.17}$$

This is the recommended approach which is used by default if not any other approach has been chosen with the `examp` option in `$rir12` (see Sec. 9.5 for further details on the options for F12 calculations; note that the `examp noinv` option should not be combined with CCSD calculations). CCSD(F12)-SP calculations are computationally somewhat less expensive that CCSD(F12) calculations which solve Eq. (11.16), while both approaches are approximately similar accurate for energy differences.

The SP approach becomes in particular very efficient if combined with the neglect of certain higher-order explicitly-correlated contributions which have a negligible effect on the energies but increase the costs during the CC iterations. The most accurate and **recommended variant is the CCSD(F12*) approximation** [21], which gives essentially identical energies as CCSD(F12)-SP. Also available are the CCSD[F12] (Ref. [21]), CCSD-F12a (Ref. [252]) and CCSD-F12b (Ref. [253]) approximations as well as the perturbative corrections CCSD(2)$_{\overline{\mathrm{F12}}}$ and CCSD(2*)$_{\overline{\mathrm{F12}}}$ (see Refs. [21, 254, 255]) and CCSD$_{(\mathrm{F12}*)}$ (Ref. [220]). These approximations should only be used with ansatz 2 and the SP approach (i.e. fixed geminal amplitudes). Note that the CCSD-F12c method which is available in the `molpro`

package is the same as CCSD(F12*). Since CCSD(F12*) is the original name that was introduced when the method was proposed for the first time in Ref. [21] users are asked to use in publications this abbreviation and cite the original reference [21].

For MP3 the approximations (F12*), and (F12) to a full F12 implementation become identical: they include all contributions linear in the coefficients c_{ij}^{kl}. The explicitly-correlated MP4 method MP4(F12*) is defined as fourth-order approximation to CCSD(F12*)(T). Note that MP4(F12*) has to be used with the SP or fixed amplitude approach for the geminal coefficients c_{ij}^{kl}. MP3(F12*) and MP4(F12*) are currently only available for closed-shell or unrestricted Hartree-Fock reference wavefunctions.

The CPU time for a CCSD(F12) calculation is approximately the sum of the CPU time for an MP2-F12 calculation with the same basis sets plus that of a conventional CCSD calculation multiplied by $(1 + N_{CABS}/N)$, where N is the number of basis and N_{CABS} the number of complementary auxiliary basis (CABS) functions (typically $N_{CABS} \approx 2{-}3N$). If the geminal coefficients are determined by solving Eq. (11.16) instead of using fixed amplitudes, the costs per CCSD(F12) iteration increase to $\approx (1 + 2N_{CABS}/N)$ times the costs for a conventional CCSD iteration. Irrespective how the geminal coefficients are determined, the disc space for CCSD(F12) calculations are approximately a factor of $\approx (1 + 2N_{CABS}/N)$ larger than the disc space required for a conventional CCSD calculation. Note that this increase in the computational costs is by far outweighted by the enhanced basis set convergence.

In combination with the CCSD(F12*) approximation (and also CCSD[F12], CCSD-F12a, CCSD-F12b, CCSD(2)$_{\overline{\text{F12}}}$, CCSD(2*)$_{\overline{\text{F12}}}$ and CCSD$_{(\text{F12}*)}$) the CPU time for the SP approach is only about 20% or less longer than for a conventional CCSD calculation within the same basis set.

CC calculations with restricted open-shell (ROHF) references: The MP2 and all CC calculations for ROHF reference wavefunctions are done by first transforming to a semi-canonical orbital basis which are defined by the eigenvectors of the occupied/occupied and virtual/virtual blocks of the Fock matrices of alpha and beta spin. No spin restrictions are applied in the cluster equations. This approach is sometimes also denoted as ROHF-UCCSD.

Note that if a frozen-core approximation is used, the semicanonical orbitals depend on whether the block-diagonalization of the Fock matrices is done in space of all orbitals or only in the space of the correlated valence orbitals. The two approaches lead thus to slightly different energies, but neither one is more valid or more accurate than the other. Both schemes are available through the specification of `core res/can`. The default is `core can` where the full occupied block is diagonalised. The same scheme is used e.g. in the CFOUR program suite, but other codes as e.g. the implementation in MOLPRO use a block-diagonalization restricted to the active valence space. The semi-canonical orbitals are written to the `$sc_alph` and `$sc_bet` data groups.

Brueckner Calculations and KS-CCSD(T): Brueckner orbital optimisation in place of single excitations is available, which is useful when the HF reference relaxes significantly in the presence of correlation (for example for calculations on heavy elements). The default way a calculation proceeds is by first building the Fock matrix for the input starting orbitals, semi-canonicalising these orbitals and then optimising the (non-frozen) orbitals at the same time as solving for the doubles amplitudes, such that the singles residual vanishes at convergence. The default is therefore UBCCD for open-shell references. The final orbitals are written in the `$bcc_mos` or `$bcc_alph`, `$bcc_bet` data groups. It is possible through options in the `$ricc2` data group to skip the initial semi-canonicalisation and to control the nature of the frozen orbitals. It is also possible to request restricted open-shell BCCD where the Brueckner optimisation applies only to $t_1^\alpha + t_1^\beta$ such that the alpha and beta doubly occupied orbitals of a starting ROHF reference remain identical to each other during the optimisation, and $t_1^\alpha - t_1^\beta$ is no longer zero at convergence. Since it is necessary to semi-canonicalise the converged orbitals in order to perform a (T) calculation, the final orbitals are always in semi-canonical form if (T) is requested. Since the F12 integrals depend on the orbitals, the iterative BCCD(F12*) approximation is very expensive. Instead, it is recommended to use the perturbative equivalent BCCD(T)$_{(\mathrm{F12}*)}$.

As an alternative to BCCD theory, it is possible (but not recommended) to used Kohn-Sham orbitals to define the reference. When using orbitals other than converged HF orbitals, it is necessary to add the `$non-canonical MOs` data group to the control file. When performing an F12 calculation, the option `r12orb arb` should be set in the `$rir12` data group.

Perturbative triples corrections: To achieve ground state energies with a high accuracy that systematically surpasses the accuracy MP2 and DFT calculations for reaction and binding energies, the CCSD model has to be combined with a perturbative correction for connected triples. The recommended approach for the correction is the CCSD(T) model

$$E_{\mathrm{CCSD(T)}} = E_{\mathrm{CCSD}} + E_{DT}^{(4)} + E_{ST}^{(5)} \tag{11.18}$$

which includes the following two terms:

$$E_{DT}^{(4)} = \sum_{\mu_2} t_{\mu_2}^{CCSD} \langle \mu_2 | [H, T_3^{(2)}] | \mathrm{HF} \rangle \tag{11.19}$$

$$E_{ST}^{(5)} = \sum_{\mu_1} t_{\mu_1}^{CCSD} \langle \mu_2 | [H, T_3^{(2)}] | \mathrm{HF} \rangle \tag{11.20}$$

where the approximate triples amplitudes are evaluated as:

$$t_{aibjck}^{(2)} = -\frac{\langle {}_{abc}^{ijk} | [\hat{H}, T_2] | \mathrm{HF} \rangle}{\epsilon_a - \epsilon_i + \epsilon_b - \epsilon_j + \epsilon_c - \epsilon_k} \tag{11.21}$$

In the literature one also finds sometimes the approximate triples model CCSD[T] (also denoted as CCSD+T(CCSD)), which is obtained by adding only $E_{DT}^{(4)}$ to the CCSD energy. Usually CCSD(T) is slightly more accurate than CCSD[T], although for closed-shell or spin-unrestricted open-shell reference wavefunctions the energies of both models, CCSD(T) and CCSD[T] model, are correct through 4.th order perturbation theory. For a ROHF reference,

however, $E_{ST}^{(5)}$ contributes already in 4.th order and only the CCSD(T) model is correct through 4.th order perturbation theory.

Integral-direct implementation and resolution-of-the-identity approximation: The computationally most demanding (in terms floating point operations) steps of a CCSD calculation are related to two kinds of terms. One of the most costly steps is the contraction

$$\Omega^B_{aibj} = \sum_{cd} t^{ij}_{cd}(ac|bd) \tag{11.22}$$

where a, b, c, and d are virtual orbitals. For small molecules with large basis sets or basis sets with diffuse functions, where integral screening is not effective, it is the time-determing step and can most efficiently be evaluated with a minimal operation count of $\frac{1}{4}O^2V^2$ (where O and V are number of, respectively occupied and virtual orbitals), if the 4-index integrals $(ac|bd)$ in the MO are precalculated and stored on file before the iterative solution of the coupled-cluster equation, 11.4 and 11.5, and the full permutational symmetries of t^{ij}_{cd}, $(ac|bd)$, and Ω_{aibj} are exploited. For larger systems, however, the storage and I/O of the integrals $(ac|bd)$ leads to bottlenecks. Alternatively, this contribution can be evaluated in an integral-direct way as

$$t^{ij}_{\kappa\lambda} = \sum_{cd} t^{ij}_{cd} C_{\kappa c} C_{\lambda d}, \qquad \Omega^B_{\mu i \nu j} = \sum_{\kappa\lambda} t^{ij}_{\kappa\lambda}(\mu\kappa|\nu\lambda), \qquad \Omega^B_{aibj} = \sum_{\mu\nu} \Omega^B_{\mu i \nu j} C_{\mu a} C_{\nu b} \tag{11.23}$$

which, depending on the implementation and system, has formally a 2–3 times larger operation count, but allows to avoid the storage and I/O bottlenecks by processing the 4-index integrals on-the-fly without storing them. Furthermore, integral screening techniques can be applied to reduce the operation count for large systems to an asymptotic scaling with $\mathcal{O}(\mathcal{N}^4)$.

In TURBOMOLE only the latter algorithm is presently implemented. (For small systems other codes will therefore be faster.)

The other class of expensive contributions are so-called ring terms (in some publications denoted as C and D terms) which involve contractions of the doubles amplitudes t_{aibj} with several 4-index MO integrals with two occupied and two virtual indices, partially evaluated with T_1-dependent MO coefficients. For these terms the implementation in TURBOMOLE employs the resolution-of-the-identity (or density-fitting) approximation (with the cbas auxiliary basis set) to reduce the overhead from integral transformation steps. Due this approximation CCSD energies obtained with TURBOMOLE will deviate from those obtained with other coupled-cluster programs by a small RI error. This error is usually in the same order or smaller than the RI error for a RI-MP2 calculation for the same system and basis sets.

The RI approximation is also used to evaluate the 4-index integrals in the MO basis needed for the perturbative triples corrections.

Disc space requirements: In difference to CC2 and MP2, the CCSD model does no longer allow to avoid the storage of double excitation amplitudes (t^{ij}_{ab}) and intermediates of

with a similar size. Thus, also the disc space requirements for CCSD calculations are larger than for RI-MP2 and RI-CC2 calculations for the same system. For a (closed-shell) CCSD ground state energy calculation the amount of disc space needed can be estimated roughly as

$$N_{disc} \approx \Big(4N^3 + (4 + m_{DIIS})O^2N^2\Big)/(128 \times 1000) \text{ MBytes} , \tag{11.24}$$

where N is the number of basis functions, O the number of occupied orbitals and m_{DIIS} the number of vectors used in the DIIS procedure (by default 10, see Sec. 23.2.22 for details).

For (closed-shell) CCSD(T) calculations the required disc space is with

$$N_{disc} \approx \Big(5N^3 + 5O^2N^2 + ON^3\Big)/(128 \times 1000) \text{ MBytes} , \tag{11.25}$$

somewhat larger.

For calculations with an open-shell (UHF or ROHF) reference wavefunction the above estimates should be multiplied by factor of 4.

Memory requirements: The CCSD and CCSD(T) implementation in TURBOMOLE uses multi-pass algorithms to avoid strictly the need to store arrays with a size of $\mathcal{O}(N^3)$ or $\mathcal{O}(O^2N^2)$ or larger as complete array in main memory. Therefore, the minimum memory requirements are relatively low—although it is difficult to give accurate estimates for them.

On should, however, be aware that, if the amount of memory provided to the program in the data group `$maxcor` becomes too small compared to $O^2N^2/(128 \times 1000)$ MBytes, loops will be broken in many small batches at the cost of increased I/O and a decrease in performance. As mentioned above, it is recommended to set `$maxcor` to 66–75% of the physical core memory available for the calculation.

Important options: The options to define the orbital and the auxiliary basis sets, the maximum amount of allocatable core memory (`$maxcor`), and the frozen-core approximation (`$maxcor`) have been mentioned above and described in the chapters on MP2 and CC2 calculations. Apart from this, CCSD and CCSD(T) calculations require very little additional input.

Relevant are in particular some options in the `$ricc2` data group:

```
$ricc2
  ccsd
  ccsd(t)
  conv=7
  oconv=6
  mxdiis=10
  maxiter=25
```

The options `ccsd` and `ccsd(t)` request, respectively, CCSD and CCSD(T) calculations. Since CCSD(T) requires the cluster amplitudes from a converged CCSD calculation, the

option `ccsd(t)` also implies `ccsd`. `bccd(t)` requests the Brueckner coupled cluster variant BCCD(T).

The number given for `mxdiis` defines the maximum number of vectors included in the DIIS procedure for the solution of the cluster equations. As mentioned above, it has some impact on the amount of disc space used by a CCSD calculation. Unless disc space becomes a bottleneck, it is not recommended to change the default value.

With `maxiter` one defines the maximum number of iterations for the solution of the cluster equations. If convergence is not reached within this limit, the calculation is stopped. Usually 25 iterations should be sufficient for convergence. Only in difficult cases with strong correlation effects more iterations are needed. It is recommended to increase this limit only if the reason for the strong correlation effects is known. (Since one reason could also be an input error as e.g. unreasonable geometries or orbital occupations or a wrong basis set assignment. With diffuse basis functions it is sometimes neccessary to tighten the integral screening threshold in `$scftol`.) If oscillatory convergence is observed, adding a level shift can help convergence.

The two parameters `conv` and `oconv` define the convergence thresholds for the iterative solution of the cluster equations. Convergence is assumed if the change in the energy (with respect to the previous iteration) is smaller than $10^{-\mathrm{conv}}$ and the euclidian norm of the residual (the so-called vector function) smaller than $10^{-\mathrm{oconv}}$. If `conv` is not given in the data group `$ricc2` the threshold for changes in the energy is set to the value given in `$denconv` (by default 10^{-7}). If `oconv` is not given in the data group `$ricc2` the threshold for the residual norm is by default set to $10 \times$ `conv`. With the default settings for these thresholds, the energy will thus be converged until changes drop below 10^{-7} Hartree, which typically ensures an accuracy of about 1 μH. These setting are thus rather tight and conservative even for the calculation of highly accurate reaction energies. If for your application larger uncertainties for the energy are tolerable, it is recommended to use less tight thresholds, e.g. `conv=6` or `conv=5` for an accuracy of, respectively, at least 0.01 mH (0.03 kJ/mol) or 0.1 mH (0.3 kJ/mol). The settings for `conv` (and `oconv`) have not only an impact on the number of iterations for the solution of the cluster equations, but as they determine the thresholds for integral screening also on the costs for the individual iterations.

CCSD(T) energy with a second-order correction from the interference-corrected MP2-F12: The error introduced from a CCSD(T) calculation with a finite basis set can be corrected from second-order corrections of the the interference-corrected MP2-F12 (INT-MP2-F12) (Ref. [256]). The approximate CCSD(T)-INT-F12 at the basis set limit is given from

$$E_{\mathrm{CCSD(T)/CBS}} \approx E_{\mathrm{CCSD(T)}} + \sum_{i<j} F^{ij}[e_{ij}^{\mathrm{MP2-F12}} - e_{ij}^{\mathrm{MP2}}]\,. \tag{11.26}$$

From `define`, in the submenu `$ricc2` select the `ccsd(t)` method and add the keyword `intcorr`

```
$ricc2
```

```
ccsd(t)
intcorr
```

Then, switch on the `f12` method (approximation `A` or `B`, `inv` or `fixed`). The corrected CCSD(T)-INT-F12 energy will be printed in the end of the calculation. It is highly recommended to start the CCSD(T)-INT-F12 calculation from a converged SCF calculation with symmetry, which is transformed to C_1. It is furthermore recommended to use Boys localized orbitals in the `$rir12` submenu. A table with the corrected second-order pair-electron energies and the corresponding interference factors can also be printed in the output by using the keyword `intcorr all` instead of `intcorr`.

Excitation energies with CCSD: At the (conventional) CCSD level also electronic excitation energies can be computed. For this the data group `$excitations` has to be added (the same keyword as for CC2 applies). The implementation is currently restricted to vertical excitation energies (no transition moments or properties available) and in the closed–shell case to singlet excited states.

Note that for single-excitation dominated transitions CCSD is as CC2 correct through second-order and is not necessarily more accurate than CC2. It is, however, for double excitations still correct through first-order, while CC2 describes double excitations only in a zero-order approximation. Therefore, CCSD results are more robust with respect to double excitation contributions to transitions and are thus usefull to check if CC2 is suitable for a certain problem.

Chapter 12

PNO-based CCSD and CCSD(T) calculations

The `pnoccsd` program provides a PNO-based implementations of CCSD including the perturbative triples corrections (T0) and (T) using RHF and UHF references. Presently the implementation is restricted to ground state energies.

Further limitations:

no MPI parallelization: calculations beyond PNO-MP2 can presently only carried out on a single compute node, only the OpenMP (see Sec. 3.4.2) parallelization is available.

no use of symmetry: no point group symmetry can be used; the calculations can only be done in C_1

Please note that one of the main strategies in PNO-CCSD calculations is to precompute most of the required two-electron integrals in the local (LMO, PNO, OSV, etc.) basis sets prior to the CCSD iterations. Therefore PNO-CCSD calculations on small molecules or with very tight thresholds can require considerably more disc space and core memory than the canonical implementation in `ccsdf12`. In general, the performance of PNO-CCSD calculations depend crucially on the speed of the disc I/O. It is therefore recommended to machines with fast RAID systems. There also the option for non-F12 calculations to activate a direct mode where the disk requirements are reduced and some quantities are evaluted on the fly.

Prerequisites

CCSD and CCSD(T) calculations with the `pnoccsd` module require the same prerequisites as calculations with `ccsdf12`:

1. a converged SCF calculation with the one-electron density threshold set to `$denconv`

`1.d-5` or less

2. an auxiliary basis defined in the data group `$cbas`
3. if orbitals should be excluded from the correlation treatment the data group `$freeze` has to be set
4. the maximum core memory which the program is allowed to allocate should be defined in the data group `$maxcor`; the recommended value is 66–75% of the available (physical) core memory. (see Sec. 23.2.3 for details)
5. the data group `$pnocsd` with a specification of the method and the PNO threshold.

In principle the the data group `$pnoccsd` does not require more input than a specification of the wavefunction model (available are currently: `mp2`, `mp3`, `ccsd`, `ccsd(t0)`, and `ccsd(t)`). The program will then use by default a PNO threshold `tolpno` of 10^{-7}. All other thresholds are, if not explicitly specified, coupled to `tolpno` (see below).

For CCSD(T) calculations one should in addition include the Laplace data group and specify the threshold for the Laplace transformation used in the PNO-based implementation of the perturbative triples correction (T). By default the program will use a threshold of `conv=2`. In many cases the PNO-CCSD(T0) approximation provides an economic alternative to PNO-CCSD(T).

Calculations at the MP3(F12) or CCSD(F12*) level require in addition:

- the data group `$rir12` with the definition of the standard approximations for the explicitly-correlated contributions (see Sec. 9.5 for details)
- the data group `$lcg`, which define the correlation function (here it is in particular important to choose for F12 calculations the exponent; recommended values are 0.9 for cc-pVDZ-F12, 1.0 for cc-pVTZ-F12 and 1.1 for cc-pVQZ-F12 basis sets)
- a complementary auxiliary (CABS) basis set

Furthermore it is recommeded to select in addition an auxiliary JK basis set for the evaluation of the Fock matrix elements. (The `rijk` menu of define can be used for this.)

How To Perform a Calculation

As presently no gradients are available, only single–point calculations are possible:

1. Select in `define` within the menu `pnocc`

   the wavefunction model (submenu `pnoccsd`),

   frozen core options (submenu `freeze`),

   an auxiliary basis for `$cbas` (submenu `cbas`),

   the amount of main memory (option `memory`),

2. Do an SCF calculation using either the `dscf` or the `ridft` module.
3. Invoke the `pnoccsd` program on the command line or with a batch script.

The minimal input for the `$pnoccsd` data group needed for a PNO-CCSD(T) calculation is just:

```
$pnoccsd
   tolpno 1.d-7
   ccsd(t)
```

To active the direct mode, use the keyword

```
$pnoccsd
   direct
```

How to quote:

- for PNO-MP2 calculations: A $\mathcal{O}(\mathcal{N}^3)$-scaling PNO-MP2 method using a hybrid OSV-PNO approach with an iterative direct generation of OSVs. Gunnar Schmitz, Benjamin Helmich, Christof Hättig, Mol. Phys. **111**, 2463–2476, (2013).
- for PNO-CCSD calculations: Explicitly correlated PNO-MP2 and PNO-CCSD and its application to the S66 set and large molecular systems. Gunnary Schmitz, Christof Hättig, David P. Tew, Phys. Chem. Chem. Phys. **16**, 22167–22178, (2014).
- for PNO-based triples (T0) or (T) calculations: Perturbative triples correction for local pair natural orbital based explicitly correlated CCSD(F12*) using Laplace transformation techniques. Gunnar Schmitz, Christof Hättig, J. Chem. Phys. **145**, 234107 (2016)
- for calculations using PAOs (default) Principal Domains in Local Correlation Theory. David. P. Tew, J. Chem. Theory Comput. **15**, 6597 (2019)

12.1 Selection Thresholds used in the `pnoccsd` Program

The accuracy and the computational costs for calculations with the `pnoccsd` program depend crucially on a number of selection thresholds. Their default values are coupled to the PNO truncation threshold `tolpno`, such that the errors from all other approximations are smaller then the PNO truncation error. Depending on the application it might be that, in particular for energy differences, certain errors cancel out to a large extent and significant computational saving might be obtained by loosening specific thresholds.

The following thresholds can be set in the data group `$pnoccsd`:

`tolpno` The PNO truncation threshold. The default value is 10^{-7} and gives typically errors in the (total) correlation energy $< 1\%$ which decrease as a rule of thumb approximately with $\sqrt{\texttt{tolpno}}$.

`tolosv` The OSV truncation threshold (cmp. Ref. [23]) which is by default set to $< 0.1 \times$ `tolpno`. This threshold mainly influences the time for the PNO-MP2 step.

`tolri` The threshold for selecting the local RI basis (cmp. Ref. [23]), which is by default set to $10^{7/12} \times \sqrt{\texttt{tolpno}}$. This threshold is important for the performance and accuracy of the integral evaluation.

`tolpair` The energy threshold for selecting the significant LMO pairs. If not specified in `$pnoccsd` it is set to $(0.1 \times \texttt{tolpno})^{2/3}$.

`toltno` The threshold for the TNO truncation. The default is `toltno` = `tolpno`. Since the TNOs are restricted through the PNOs this is the tightest useful value. Setting it tighter than `tolpno` will usually not improve the accuracy. It only affects the triples corrections and is important for their costs and accuracy.

`toltrip` The selection threshold for LMO triples included for the (T0) and (T) corrections. By default `toltrip` is set to `toltno`. It only affects the triples corrections and is important for their costs and accuracy.

`tolopao` The selection threshold for PAO domains for OSVs (cmp. Ref. [257]).

`tolppao` The selection threshold for PAO domains for PNOs (cmp. Ref. [257]).

`toltpao` The selection threshold for PAO domains for TNOs.

For further options and thresholds see Sec. 23.2.24.

12.2 Characteristics of the Implementation

The `pnoccsd` program works* with localized molecular occupied orbitals (LMOs). Their local nature is exploited to truncate for each LMO pair the number of interacting virtual orbitals and thereby the number of required two-electron integrals. PNO-MP2 belongs to the family of local correlation methods which have in common the key idea that the correlation energy can be written as a sum of pair contributions

$$E_{corr}^{\mathrm{MP2}} = \sum_{i \leq j} e^{ij} = \frac{1}{4} \sum_{iajb} t_{ab}^{ij} \langle ij||ab \rangle \tag{12.1}$$

and that with localized molecular orbitals the pair contributions e^{ij} decrease fast with the distance $|r_{ij}|$ between the LMOs i and j (approximately with $1/|r_{ij}|^6$) so that for large molecules the number of pairs that need to be included to compute the correlation energies within a given accuracy increases only linearly with the number of atoms. Furthermore,

*The program can also run with canonical orbitals but will then not be efficient.

only such virtual orbitals contribute significantly to the correlation energy e^{ij} of a pair ij which are spatially close to i and j.

The OSV-PNO hybrid approach implemented in the `pnoccsd` program starts with the determination of truncated subsets of projected atomic orbitals (PAOs) and orbital-specific virtual (OSV) orbitals for each LMO i as the eigenvectors of diagonal density contributions D^{ii} computed from (approximate) MP2 doubles amplitudes:

$$D^{ii}_{ab} = 2\sum_{c} t^{ii}_{ac} t^{ii}_{cb} = \sum_{\tilde{c}} d^{ii}_{a\tilde{c}} n^{ii}_{\tilde{c}} d^{ii}_{b\tilde{c}} \tag{12.2}$$

with occupation numbers $n^{ii}_{\tilde{c}}$ not smaller than a predefined threshold `tolosv`. The OSVs are first used to calculate for each pair ij estimates for the pair correlation energy contribution e^{ij} and to set up a list of pairs ij for which significant contributions to the total correlation energy are expected. Only for this list of pairs the OSVs of the two LMOs are merged together to an orthonormal set of pre-PNOs $\tilde{c}_{ij}$ which are used to compute the contribution of the pair ij to the density matrix via approximated MP2 amplitudes and as eigenvectors of these densities the coefficients of the pair natural orbitals:

$$D^{ij}_{\tilde{a}_{ij}\tilde{b}_{ij}} = 2\sum_{\tilde{c}_{ij}} t^{ij}_{\tilde{a}_{ij}\tilde{c}_{ij}} t^{ij}_{\tilde{c}_{ij}\tilde{b}_{ij}} = \sum_{\bar{c}} d^{ij}_{a\bar{c}} n^{ij}_{\bar{c}} d^{ij}_{b\bar{c}} \ . \tag{12.3}$$

Similar as for the OSVs also the PNOs with occupation numbers $n^{ij}_{\bar{c}}$ below the PNO truncation threshold `tolpno` are discarded.

For OSV-PNO-MP2 the MP2 equations are then solved with the amplitudes and two-electron integals projected onto the so determined set of pair natural orbitals. For the explicitly-correlated variant OSV-PNO-MP2-F12 similar sets of orbital and pair specific auxiliary orbitals (OSX) are determined for the occupied and the complementary virtual orbital spaces which appear in the calculation of the three-electron integrals for the additional contributions from the geminals.

The thresholds for discarding PNOs and OSVs and the subset of auxiliary orbitals for the local RI approximation for computing two-electron integrals play an important role for the accuracy and the computational costs of calculations with the `pnoccsd` program. Per default all thresholds are coupled to the PNO truncation threshold `tolpno` such that the errors due to the OSV, local RI and pair truncation approximations and due to the screening of (AO) integrals and contributions to the residual are about an order of magnitude smaller than the PNO truncation error. The default value for `tolpno` is 10^{-7} which gives PNO truncation errors in the correlation energy $< 1\%$. Test calculations showed that the relative errors in the correlation energy decrease typically linear with $\sqrt{\texttt{tolpno}}$.

All approximations in the CCSD energy are equivalent for UHF and RHF references such that a UHF-based calculation on a closed shell species gives the same energy as the RHF-based calculation to within numerical precision. In V7.4.2, the approximations involved in the (T) energy lead to small differences between UHF and RHF based calculations of a closed shell system.

12.2.1 Strong pair approximation

Beyond MP2 the PNO implementation makes per default use of a strong pair approximation:

- The 3.rd order CCD ladder terms and corresponding exchange terms:

$$\sum_{kl} t_{ab}^{kl}(ki\tilde{|}lj) + \sum_{cd} t_{cd}^{ij}(ac\tilde{|}cd) - \sum_{ck}\Big((ki\tilde{|}bc)t_{ac}^{kj} + (kj\tilde{|}ac)t_{bc}^{ki}\Big) \tag{12.4}$$

  decay together as R^{-6} with the distance between the centers of the LMOs i and j and for the first term also with the distance between the centers of the LMOs k and l. These terms are only evaluated if ij is a strong pair. For the last two terms in addition ki and kj and for the first term kl, ki, and lj are required to be strong pairs.

- The Coulomb integrals $(ki\tilde{|}ca)$ in the ring terms are neglected if ki is a distant or neglected pair.

Per default the threshold for $T_{\text{strg}} = \sqrt{10 \times T_{\text{PNO}}}$ and $T_{\text{weak}} = T_{\text{pair}}^{2/3} \times T_{\text{strg}}^{1/3}$. Above $(pq\tilde{|}rs)$ denote t_1-dressed two-electron integrals.

Chapter 13

Random Phase Approximation and Beyond: Energy and First-Order Properties

Ground state energies and analytic first-order properties (e.g., 'gradients' for structure optimizations) can be computed within the random phase approximation (RPA) using the `rirpa` module. Theory and development of the `rirpa` module is published in references [258, 259] for the energy and reference [260] for the first-order properties. See also [261] for a survey of applications of RIRPA. For two-component relativistic RPA energy calculations, see reference [262]. For perturbative beyond-RPA calculations, see references [263, 264]. For orbital-self-consistent approach called the generalized Kohn–Sham semi-canonical projected RPA (GKS-spRPA), see reference [265]. σ-functionals based on RPA are described in references [266–269]. For energy and gradients, the resolution-of-the-identity (RI) approximation is used to approximate the two-electron repulsion integrals in the correlation treatment and is combined with an imaginary frequency integration. The RI approximation is also employed by default for the computation of the Coulomb integrals for the HF energy. For the energy, it is optional to use RI for the Fock exchange integrals ('RI-K'), while RI-K for the gradients is not available yet. Open shell systems and the frozen core approximation may be used in RPA energy calculations but are not presently available in gradient calculations. Two-component RPA energy calculations are only possible for Kramers-restricted closed-shell systems. ECPs are presently not compatible with RIRPA gradients. Neither RPA energy nor gradients support symmetry at the moment. The gradients may be used together with the scripts `jobex` (for structure optimizations) and `NumForce` (for numerical harmonic vibrational frequencies).

13.1 Ground State Energy Theory

The RPA energy

$$E^{\mathrm{RPA}} = E^{\mathrm{HF}} + E^{\mathrm{C\ RPA}} \tag{13.1}$$

consists of the Hartree-Fock exact exchange energy E^{HF} and a correlation energy piece $E^{\mathrm{C\ RPA}}$. `rirpa` computes Eq. (13.1) non-selfconsistently from a given set of converged input orbitals. The correlation energy

$$E^{\mathrm{C\ RPA}} = \frac{1}{2}\sum_n \left(\Omega_n^{\mathrm{RPA}} - \Omega_n^{\mathrm{TDARPA}}\right) \tag{13.2}$$

is expressed in terms of RPA excitation energies at full coupling Ω_n^{RPA} and within the Tamm-Dancoff approximation $\Omega_n^{\mathrm{TDARPA}}$. The further discussion is restricted to the one-component (nonrelativistic) treatment, for the sake of convenience. For the derivation of the two-component RPA theory see ref. [262]. The excitation energies are obtained from time-dependent DFT response theory and are eigenvalues of the symplectic eigenvalue problem [270, 271]

$$(\Lambda - \Omega_{0n}\Delta)|X_{0n}, Y_{0n}\rangle = 0. \tag{13.3}$$

The super-vectors X_{0n} and Y_{0n} are defined on the product space $L_{\mathrm{occ}} \times L_{\mathrm{virt}}$ and $L_{\mathrm{occ}} \times L_{\mathrm{virt}}$, respectively, where L_{occ} and L_{virt} denote the one-particle Hilbert spaces spanned by occupied and virtual static KS molecular orbitals (MOs). The super-operator

$$\Lambda = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{A} \end{pmatrix} \tag{13.4}$$

contains the so-called orbital rotation Hessians,

$$(A+B)_{iajb} = (\epsilon_a - \epsilon_i)\delta_{ij}\delta_{ab} + 2(ia|jb), \tag{13.5}$$

$$(A-B)_{iajb} = (\epsilon_a - \epsilon_i)\delta_{ij}\delta_{ab}. \tag{13.6}$$

ϵ_i and ϵ_a denote the energy eigenvalues of canonical occupied and virtual KS MOs. `rirpa` computes so-called direct RPA energies only, i.e. no exchange terms are included in Eqs. (13.5) and (13.6).

In RIRPA the two-electron integrals in Eqs (13.5) are approximated by the resolution-of-the-identity approximation. In conjunction with a frequency integration this leads to an efficient scheme for the calculation of RPA correlation energies [258]

$$E^{\mathrm{C\ RIRPA}} = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} F^{\mathrm{C}}(\omega), \tag{13.7}$$

where the integrand contains $N_{\mathrm{aux}} \times N_{\mathrm{aux}}$ quantities only,

$$F^{\mathrm{C}}(\omega) = \frac{1}{2}\mathrm{tr}\left(\ln\left(\mathbf{I}_{\mathrm{aux}} + \mathbf{Q}(\omega)\right) - \mathbf{Q}(\omega)\right). \tag{13.8}$$

N_{aux} is the number of auxiliary basis functions. The integral is approximated using Clenshaw-Curtiss quadrature.

Prerequisites

Calculations with `rirpa` require

- a converged SCF calculation
- `rirpa`-options may be included by adding them in the lines below the keyword `$rirpa` in the control file. Possible options are:
  - `npoints` ⟨integer⟩ - Number of frequency integration points (default is 60).
  - `nohxx` - HF energy is skipped, (HXX = Hartree + eXact (Fock) eXchange).
  - `rpaprof` - Generates profiling output.
- the maximum core memory the program is allowed to allocate should be defined in the data group `$maxcor` (in MB); the recommended value is ca. 3/4 of the available (physical) core memory at most.
- orbitals to be excluded from the correlation treatment have to be specified in data group `$freeze`
- an auxiliary basis defined in the data group `$cbas`
- an auxiliary basis defined in the data group `$jbas` for the computation of the Coulomb integrals for the Hartree-Fock energy. This is optional if the `nohxx` option is set.
- (optional) an auxiliary basis defined in the data group `$jkbas` for the computation of the exchange integrals for the Hartree-Fock energy. `$rik` should be added to the control file for RI-JK to be effective.

To perform a **two-component relativistic RIRPA calculation** [262] on (Kramers-restricted) closed-shell systems taking into account spin-orbit coupling, the two-component version of `ridft` has to be run before (see Chapter 6.4) using the keywords `$soghf` and `$kramers`. The implementation is currently only available in combination with the `nohxx` option.

13.2 Gradients Theory

All details on the theory and results are published in [260]. The RI-RPA energy is a function of the MO coefficients $\mathbf{C}$ and the Lagrange multipliers $\boldsymbol{\epsilon}$ and depends parametrically (i) on the interacting Hamiltonian $\hat{H}$, (ii) on the AO basis functions and the auxiliary basis functions. All parameters may be gathered in a supervector $\mathbf{X}$ and thus

$$E^{\mathrm{RIRPA}} \equiv E^{\mathrm{RIRPA}}(\mathbf{C}, \boldsymbol{\epsilon}|\mathbf{X}) \; . \tag{13.9}$$

$\mathbf{C}$ and $\boldsymbol{\epsilon}$ in turn depend parametrically on $\mathbf{X}$, the exchange-correlation matrix $\mathbf{V}^{\mathrm{XC}}$, and the overlap matrix $\mathbf{S}$ through the KS equations and the orbital orthonormality constraint. First-order properties may be defined in a rigorous and general fashion as total derivatives

of the energy with respect to a "perturbation" parameter ξ. However, the RI-RPA energy *is not* directly differentiated in our method. Instead, we define the RI-RPA energy Lagrangian

$$\begin{aligned} &L^{\mathrm{RIRPA}}(\mathcal{C},\mathcal{E},\mathcal{D}^{\Delta},\mathcal{W}|\mathbf{X},\mathbf{V}^{\mathrm{XC}},\mathbf{S}) \\ &= E^{\mathrm{RIRPA}}(\mathcal{C},\mathcal{E}|\mathbf{X}) + \sum_{\sigma}\left(\left\langle \mathcal{D}^{\Delta}_{\sigma}(\mathcal{C}^{\mathrm{T}}_{\sigma}\mathbf{F}_{\sigma}\mathcal{C}_{\sigma}-\mathcal{E}_{\sigma})\right\rangle - \left\langle \mathcal{W}_{\sigma}(\mathcal{C}^{\mathrm{T}}_{\sigma}\mathbf{S}\mathcal{C}_{\sigma}-\mathbf{1})\right\rangle\right) . \end{aligned} \tag{13.10}$$

$\mathcal{C}$, $\mathcal{E}$, $\mathcal{D}^{\Delta}$, and $\mathcal{W}$ are independent variables. L^{RIRPA} is required to be stationary with respect to $\mathcal{C}$, $\mathcal{E}$, $\mathcal{D}^{\Delta}$, and $\mathcal{W}$. $\mathcal{D}^{\Delta}$ and $\mathcal{W}$ act as Lagrange multipliers enforcing that $\mathcal{C}$ and $\mathcal{E}$ satisfy the KS equations and the orbital orthonormality constraint,

$$\left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathcal{D}^{\Delta}_{\sigma}}\right)_{\mathrm{stat}} = \mathcal{C}^{\mathrm{T}}_{\sigma}\mathbf{F}_{\sigma}\mathcal{C}_{\sigma}-\mathcal{E}_{\sigma} = \mathbf{0} , \tag{13.11}$$

$$\left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathcal{W}_{\sigma}}\right)_{\mathrm{stat}} = \mathcal{C}^{\mathrm{T}}_{\sigma}\mathbf{S}\mathcal{C}_{\sigma}-\mathbf{1} = \mathbf{0} . \tag{13.12}$$

$\mathcal{D}^{\Delta}$ and $\mathcal{W}$ are determined by the remaining stationarity conditions,

$$\left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathcal{E}}\right)_{\mathrm{stat}} = \mathbf{0} , \tag{13.13}$$

and

$$\left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathcal{C}}\right)_{\mathrm{stat}} = \mathbf{0} . \tag{13.14}$$

It turns out from eqs (13.13) and (13.14) that the determination of $\mathcal{D}^{\Delta}$ and $\mathcal{W}$ requires the solution of a single Coupled-Perturbed KS equation. Complete expressions for $\mathcal{D}^{\Delta}$ and $\mathcal{W}$ are given in [260]. At the stationary point "stat $= (\mathcal{C}=\mathbf{C},\mathcal{E}=\boldsymbol{\epsilon},\mathcal{D}^{\Delta}=\mathbf{D}^{\Delta},\mathcal{W}=\mathbf{W})$", first-order RI-RPA properties are thus efficiently obtained from

$$\begin{aligned} \frac{dE^{\mathrm{RIRPA}}(\mathbf{C},\boldsymbol{\epsilon}|\mathbf{X})}{d\xi} &= \left\langle \left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathbf{X}}\right)_{\mathrm{stat}} \frac{d\mathbf{X}}{d\xi}\right\rangle + \left\langle \left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathbf{V}^{\mathrm{XC}}}\right)_{\mathrm{stat}} \left(\frac{\partial \mathbf{V}^{\mathrm{XC}}}{\partial \xi}\right)_{\mathrm{stat}}\right\rangle \\ &\quad + \left\langle \left(\frac{\partial L^{\mathrm{RIRPA}}}{\partial \mathbf{S}}\right)_{\mathrm{stat}} \frac{d\mathbf{S}}{d\xi}\right\rangle . \end{aligned} \tag{13.15}$$

Finally, the RPA energy gradients may be explicitly expanded as follows:

$$\begin{aligned} \frac{dE^{\mathrm{RIRPA}}(\mathbf{C},\boldsymbol{\epsilon}|\mathbf{X})}{d\xi} &= \left\langle \mathbf{D}^{\mathrm{RIRPA}}\frac{d\mathbf{h}}{d\xi}\right\rangle + \left\langle \boldsymbol{\Gamma}^{(4)}\frac{d\boldsymbol{\Pi}^{(4)}}{d\xi}\right\rangle + \left\langle \mathbf{D}^{\Delta}\left(\frac{\partial \mathbf{V}^{\mathrm{XC}}[\mathbf{D}]}{\partial \xi}\right)_{\mathrm{stat}}\right\rangle \\ &\quad + \left\langle \boldsymbol{\Gamma}^{(3)}\frac{d\boldsymbol{\Pi}^{(3)}}{d\xi}\right\rangle + \left\langle \boldsymbol{\Gamma}^{(2)}\frac{d\boldsymbol{\Pi}^{(2)}}{d\xi}\right\rangle - \left\langle \mathbf{W}\frac{d\mathbf{S}}{d\xi}\right\rangle . \end{aligned} \tag{13.16}$$

where $\mathbf{D}^{\mathrm{RIRPA}}$ is the KS ground state one-particle density matrix $\mathbf{D}$ *plus* the RI-RPA difference density matrix $\mathbf{D}^{\Delta}$ which corrects for correlation and orbital relaxation effects. $\mathbf{h}$ is the one-electron Hamiltonian; $\boldsymbol{\Pi}^{(2/3/4)}$ are 2-, 3-, and 4-centre electron repulsion integrals and the $\boldsymbol{\Gamma}^{(2/3/4)}$ are the corresponding 2-, 3-, and 4-index relaxed 2-particle density matrices; $\mathbf{W}$ may be interpreted as the energy-weighted total spin one-particle density matrix.

This result illustrates the *key advantage* of the Lagrangian method: *Total* RI-RPA energy derivatives featuring a complicated implicit dependence on the parameter $\mathbf{X}$ through the variables $\mathbf{C}$ and $\boldsymbol{\epsilon}$ are replaced by *partial* derivatives of the RI-RPA Lagrangian, whose computation is straightforward once the stationary point of the Lagrangian has been fully determined.

Gradients Prerequisites

Geometry optimizations and first order molecular property calculations can be executed by adding the keyword `rpagrad` to the `$rirpa` section in the control file. RPA gradients also require

- an auxiliary basis defined in the data group `$jbas` for the computation of the Coulomb integrals for the Hartree-Fock energy
- an auxiliary basis defined in the data group `$cbas` for the ERI's in the correlation treatment.
- zero frozen core orbitals; RIRPA gradients are *not* compatible with the frozen core approximation at this time.

The following gradient-specific options may be further added to the `$rirpa` section in the control file

- `drimp2` - computes gradients in the DRIMP2 limit.
- `niapblocks` ⟨integer⟩ - Manual setting of the integral block size in subroutine rirhs.f; for developers.

In order to run a geometry optimization, `jobex` must be invoked with the level set to rirpa, and the `-ri` option (E.g. `jobex -ri -level rirpa`).

In order to run a numerical frequency calculation, `NumForce` must be invoked with the level set to rirpa, e.g., `NumForce -d 0.02 -central -ri -level rirpa`.

13.3 Generalized Kohn Sham scheme for RIRPA

The RIRPA energy, discussed in Sec. 13.1, is non-self-consistently computed using canononical KS orbitals obtained from a prior KS-DFT based calculation. If the KS-DFT densities are poor, the errors can carry over to post-KS RIRPA energetics resulting in poor interaction energies in some cases. Such density-driven errors can be alleviated by an orbital self-consistent approach called the generalized Kohn–Sham semi-canonical projected RPA (GKS-spRPA). In the GKS-spRPA method, the spRPA energy functional

$$E^{\mathrm{spRPA}}[\mathbf{D}, \tilde{\mathbf{H}}_0^{\mathrm{KS}}[\mathbf{D}]] = E^{\mathrm{HF}}[\mathbf{D}] + E^{\mathrm{C\ spRPA}}[\mathbf{D}, \tilde{\mathbf{H}}_0^{\mathrm{KS}}[\mathbf{D}]] \tag{13.17}$$

is minimized with respect to the non-interacting GKS density matrix

$$\mathbf{D} = \sum_{\lambda} \mathbf{P}_\lambda \mathbf{n}_{\lambda\lambda'} \mathbf{P}_\lambda. \tag{13.18}$$

$\mathbf{D}$ is constrained to be normalized to N electrons and have eigenvalues between 0 and 1. $\mathbf{P}_\lambda$ denotes orthogonal projectors belonging to blocks of KS orbitals with degenerate occupation numbers, and $\mathbf{n}_{\lambda\lambda'} = \mathbf{n}_\lambda \delta_{\lambda\lambda'}$ is diagonal, with $\mathbf{n}_\lambda$ denoting occupation number matrices. For

integer KS occupations $\mathbf{n}_\lambda$ has eigenvalues $n_\lambda = 1, 0$. The semicanonical projected (sp) KS Hamiltonian

$$\tilde{\mathbf{H}}_0^{\mathrm{KS}} = \sum_\lambda \mathbf{P}_\lambda \mathbf{H}_0^{\mathrm{KS}} \mathbf{P}_\lambda, \tag{13.19}$$

contains only the diagonal ($\lambda = \lambda'$) blocks of the KS Hamiltonian

$$\mathrm{H}_{0,ij}^{\mathrm{KS}} = \mathrm{h}_{ij} + \sum_{pq} \mathrm{V}_{ipjq} \mathrm{D}_{pq} + \mathrm{V}_{ij}^{\mathrm{XC}}[\mathrm{D}] \, . \tag{13.20}$$

$\mathbf{h}$ is the one-electron Hamiltonian, the second term denotes the Hartree or Coulomb potential and $\mathbf{V}^{\mathrm{XC}}$ is the exchange-correlation potential. The eigenvalues of sp KS Hamiltonian are denoted by $\tilde{\varepsilon}_i^{KS}$. $\mathbf{V}$ is the matrix of two-electron integrals

$$\mathrm{V}_{pqrs} = \int\int d^3r_1 d^3r_2 \frac{\phi_p^*(\mathbf{r}_1)\phi_q^*(\mathbf{r}_2)\phi_r(\mathbf{r}_1)\phi_s(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|} \, . \tag{13.21}$$

The subscripts $i, j, ..$ denote orbital indices. The spRPA energy functional, eq. 13.17, thus generalizes the post-KS RPA energy functional for arbitrary density matrices. A GKS energy minimization in the space of density matrices leads to a set of one-particle equations,

$$\mathbf{H}^{\mathrm{spRPA}}[\mathbf{D}]\phi_p = \varepsilon_p^{\text{GKS-spRPA}} \phi_p \, , \tag{13.22}$$

which are solved self-consistently. The one-particle spRPA Hamiltonian is the functional derivative of the RPA energy functional, and contains contributions from the HF potential and RPA correlation potential,

$$\mathbf{H}^{\mathrm{spRPA}}[\mathbf{D}] = \frac{\delta E^{\mathrm{spRPA}}[\mathbf{D}]}{\delta \mathbf{D}} = \mathbf{H}^{\mathrm{HF}}[\mathbf{D}] + \mathbf{V}^{\mathrm{C\ spRPA}}[\mathbf{D}] \, . \tag{13.23}$$

The self-consistent procedure used in eq. 13.22 is similar to that used in Hartree-Fock and KS-DFT. At the stationary point of the minimization procedure, the GKS-spRPA procedure yields a total many-body energy and one-particle orbital energies, $\varepsilon_p^{\text{GKS-spRPA}}$. The latter are related to approximate ionization potentials and electron affinities. $\varepsilon_p^{\text{GKS-spRPA}}$ provide consistently accurate estimates of IPs and EAs due to the inclusion of static Hartree-exchange effects, via $\mathbf{H}^{\mathrm{HF}}$ contribution, and orbital-correlation and orbital-relaxation effects, via $\mathbf{V}^{\mathrm{C\ spRPA}}[\mathbf{D}]$ contribution. $\varepsilon_p^{\text{GKS-spRPA}}$ provide accurate estimates of valence ionization potentials for neutral and anionic systems [265]. The balanced inclusion of orbital-correlation and -relaxation effects leads to accurate estimates of absolute and relative core-ionization energies [272].

The computation of the complete one-particle spectrum currently has a scaling of $\mathcal{O}(N^6)$. To reduce the computational costs, we carry out the minimization in two steps: (i) We approximate the occupied-occupied and virtual-virtual blocks of $\mathbf{H}^{\mathrm{spRPA}} \approx \mathbf{H}^{HF}$ for the self-consistent procedure and obtain the converged total energy, and then (ii) evaluate the one-particle eigenvalues at the final converged stationary point only. This two step procedure effects the rate of energy-convergence only and not the final total energy while maintaining a computational scaling of $\log(N)\mathcal{O}(N^4)$ for energy evaluation in each iteration of step (i). Thus the cost of GKS-spRPA total energy is number of iterations times the cost for the computation of right-hand side vector, which has a scaling $\log(N)\mathcal{O}(N^4)$. (Note: The

computation of the one-particle eigenspectrum, i.e. the step (ii), will be available in a future release.)

We note that the GKS-spRPA energy functional has a parametric dependence on the KS potential. However, the variational GKS minimization of the spRPA energy functional reduces the dependency; the quality of results, for energy differences and IPs/EAs, is similar for different KS potentials. We also note that GKS energy minimization of small-gap systems may have poor or no convergence. For such cases, the convergence can be improved by a (small) artificial-shift of the KS orbital energy differences.

Prerequisites

GKS-spRPA total energy calculation requires

- a converged KS-DFT calculation which provides the starting guess MOs for GKS-spRPA. scheme.
- relevant `rirpa`-options:
  - `npoints` ⟨integer⟩ - Number of frequency integration points (default is 60).
  - `iter` ⟨integer⟩ - Turns on GKS-spRPA self-consistent iterations. ⟨integer⟩ is the number of GKS-spRPA iterations (default is 0).
  - `ldiis` - Turns on DIIS algorithm which speeds up energy convergence.
  - `eigshift` ⟨real⟩ - (optional) Introduces a shift to the non-interacting gap to aid in the convergence of GKS iterations. This may be required for small-gap systems. (Note: Carefully check your results if eigshift is being used. After the final iteration, rerun a rirpa energy calculation without the `iter` a nd `eigshift` keywords. The resulting energy should be used as the final GKS-spRPA energy.)
  - `output` ⟨string⟩ - (optional) A condensed version of the output relevant to GKS-spRPA energetics is written to the file ⟨string⟩ (default filename is gksrpa.dat).
- the convergence criterion for total energy is controlled by the `$scfconv` keyword.

13.4 σ-Functionals

σ-functionals offer a significant improvement over pure direct (i.e., not self-consistent) RPA at similar computational cost.

Formally, σ-functionals are rooted in perturbation theory along the adiabatic connection which yields an expression for the total energy which closely resembles the RPA energy expression (eq. 13.1, 13.7). [266–268]

$$E^{\sigma f} = E^{\mathrm{HF}} + E^{\mathrm{C},\sigma f} \tag{13.24}$$

with

$$E^{\mathrm{C},\sigma f} = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} F^{\mathrm{C}}(\omega), \tag{13.25}$$

and

$$F^{\mathrm{C}}(\omega) = \frac{1}{2}\mathrm{tr}\left(\ln\left(\mathbf{I}_{\mathrm{aux}} + \mathbf{Q}(\omega)\right) - \mathbf{Q}(\omega) - H(\boldsymbol{\sigma}(\omega))\right). \tag{13.26}$$

where $\mathbf{Q}(\omega)$ is the negative response matrix at the imaginary frequency iω represented in an auxiliary basis, and $\boldsymbol{\sigma}(\omega)$ is a diagonal matrix containing the eigenvalues of $\mathbf{Q}(\omega)$. The function H is represented by cubic splines and has been optimized for different DFT functionals and test sets. [267, 268] Currently, only so-called unscaled σ-functionals are supported.

The present implementation features both ground state energies and analytical gradients facilitating geometry optimizations and calculation of numerical vibrational frequencies. [269]

Prerequisites

A σ-functional calculation can be envoked by adding

`sigmafunc` *`parametrization`*

to the `rirpa` input section where *`parametrization`* can be one of `pbe_s1`, `pbe_s2`, `pbe_w1`, `pbe0_s1`, `pbe0_s2`, `pbe0_w1`, `b3-lyp_w1`, and `tpss_w1`, see Ref. [267, 268] for more details about the different parametrizations. If *`parametrization`* is omitted, the default is `pbe_s1`. It is recommended to use orbitals and eigenvalues generated with the DFT functional indicated in the parametrization. For example, a σ-functional calculation with parametrization `pbe_s1` should be carried out using PBE orbitals and eigenvalues from a preceeding `ridft` run.

Since the function H introduces additional structure to $F^{\mathrm{C}}(\omega)$, it is recommended to switch on Gauss–Legendre quadrature for the imaginary frequeny integration where the default number of 60 frequency points typically yields very well converged results; experience shows that usually 35 – 50 points suffice in case of Gauss–Legendre quadrature. For small-gap and open-shell systems, somewhat more points may be required. See the keyword section 23.2.19 for more details.

Otherwise, the general notes given for the RPA energy / gradient apply.

13.5 Further Recommendations

- The direct RPA correlation energy is defined in a Kohn-Sham context without inclusion of exchange integrals and therefore the use of self-consistent KS orbitals obtained from (semi-)local functionals is recommended. HF-orbitals or KS-orbitals obtained form hybrid functionals may lead to inferior results. In case of σ-functionals, hybrid functionals lead to improvements, especially for main group compounds.
- Experience has demonstrated that the difference in RPA correlation energies obtained from different (semi-)local functionals is very small (much smaller than the inherent error of the method).
- Like MP2, RIRPA results are known to converge very slowly with increasing basis set size, in particular slowly with increasing l-quantum number of the basis set. For

reliable results the use of QZVP basis sets (or higher) is recommended. For non-covalently bound systems larger basis sets (especially, with more diffuse functions) are needed.

- It is recommended to exclude all non-valence orbitals from RIRPA calculations, as neither the `TURBOMOLE` standard basis sets SVP, TZVPP, and QZVPP nor the cc-pVXZ basis set families (with X=D,T,Q,5,6) are designed for correlation treatment of inner shells (for this purpose polarisation functions for the inner shells are needed). The default selection for frozen core orbitals in `Define` (orbitals below -3 a.u. are frozen) provides a reasonable guess. If core orbitals are included in the correlation treatment, it is recommended to use basis sets with additional tight correlation functions as e.g. the cc-pwCVXZ and cc-pCVXZ basis set families.

- We recommend the use of auxiliary basis sets optimized for the corresponding (MO) basis sets. The auxiliary basis sets optimized for RI-MP2 and RI-CC2 are suitable for `rirpa` [258] correlation energy calculations.

- Within the two-component relativistic implementation, RIRPA total energies (HF@KS + correlation) must be computed in two steps. RIRPA correlation energies can be obtained using the `nohxx` option, and the HF energy can then be computed separately, e.g., in `ridft` if the RI-J approximation is used for the Coulomb integrals. To compute the HF@KS energy, compute the KS orbitals first; then disable `$dft` and set `$scfiterlimit 1` in the `control` file to perform a single SCF iteration. Finally add the total HF@KS energy from `ridft` to the correlation energy from the `nohxx-rirpa` calculation to obtain the total RIRPA energy. *Note:* the molecular orbitals are altered by `ridft` after a single-iteration, so the HF@KS energy must be computed *after* the RIRPA correlation energy.

- Tight SCF (`$scfconv 7`) and one-electron density matrix (`$denconv 1d-7`) convergence criteria, large basis sets (QZVP), and large frequency grids which ensure a sensitivity measure of no more than `1d-4` should be used in combination with `rpagrad` for accurate results.

13.6 Comments on the Output

- The most important output for `rirpa` are the Hartree-Fock (HXX) energy and the RIRPA correlation energy which are written to the standard output.

- The optimal scaling parameter for the quadrature grid is printed together with a sensitivity parameter. The sensitivity parameter provides a numerical estimate for the error in the numerical integration used to evaluate $E^{\mathrm{C\ RIRPA}}$. Experience demonstrates that the sensitivity parameter correlates well with the condition number of the matrix $\mathbf{Q}$. Small gap systems have large condition numbers and therefore require large grids. An estimate of the number of eigenvalues smaller than 0.05 H is given and if necessary, a warning to increase the grid size is printed to standard output.

- The molecular dipole and quadrupole moments are written to the standard output whenever a gradient calculation is carried out.
- Rotational constants are provided in the `rirpa` output below the coordinate section.
- Enabling the option `rpaprof` will output additional timings information.

Chapter 14

Many body perturbation theory in the GW approximation

DFT estimates of both single-particle excitation energies and vertical excitation spectra can be systematically improved upon by the GW- and *Bethe–Salpeter* methods, which are both based on a single-particle Green's function.

14.1 Single particle spectra based on the GW approximation

14.1.1 Theoretical background

A method to systematically improve upon DFT estimates of single-particle excitation spectra, that is, ionization potentials and electron affinities, is the GW method. Its central object is the single-particle Green's function G; its poles describe single-particle excitation energies and lifetimes. In particular, the poles up to the Fermi-level correspond to the primary vertical ionization energies. The GW-approach is based on an exact representation of G in terms of a power series of the screened Coulomb interaction W, which is called the *Hedin equations*. The GW-equations are obtained as an approximation to the Hedin-equations, in which the screened Coulomb interaction W is calculated neglecting so called *vertex corrections*. In this approximation the self–energy Σ, which connects the fully interacting Green's function G to a reference non-interacting Green's function G_0, is given by $\Sigma = GW$.

This approach can be used to perturbatively calculate corrections to the Kohn–Sham spectrum. To this end, the Green's function is expressed in a spectral representation as a sum

of quasi particle states.

$$G(\mathbf{r},\mathbf{r}';z)=\sum_n \frac{\Psi_{\mathrm{r},n}(\mathbf{r},z)\Psi^{\dagger}_{\mathrm{l},n}(\mathbf{r}',z)}{z-\varepsilon_n(z)+i\eta\,\mathrm{sgn}(\varepsilon_n-\mu)}. \tag{14.1}$$

Under the approximation that the KS states are already a good approximation to these quasi–particle states $\Psi_{\mathrm{l},n}$ the leading order correction can be calculated by solving the zeroth order quasi–particle equation:

$$\varepsilon_n=\epsilon_n+\langle n|\Sigma[G_{\mathrm{KS}}](\varepsilon_n)-V_{\mathrm{xc}}|n\rangle \tag{14.2}$$

An approximation to the solution of this equation can be obtained by linearizing it:

$$\varepsilon_n=\epsilon_n+Z_n\,\langle n|\,\Sigma(\epsilon_n)-V_{\mathrm{xc}}\,|n\rangle \tag{14.3}$$

here, Z_n is given by:

$$Z_n=\left[1-\langle n|\,\frac{\partial\Sigma(E)}{\partial E}\bigg|_{E=\epsilon_n}|n\rangle\right]^{-1} \tag{14.4}$$

reducing the computational effort to a single iteration.

The self–energy Σ appearing in Eqn. (14.2) is calculated in the *GW* approximation from the KS Green's function and screening. This is the so-called G_0W_0 approximation. The Self–energy splits in an energy independent exchange part Σ^{x} and a correlation part $\Sigma^{\mathrm{c}}(E)$ that does depend on energy. Their matrix elements are given by:

$$\langle n|\,\Sigma^{\mathrm{x}}\,|n'\rangle \quad = \quad =-\sum_i (ni|in')\,, \tag{14.5}$$

and

$$\langle n|\,\Sigma^{\mathrm{c}}(\epsilon_n)\,|n\rangle \quad = \quad \sum_m\sum_{\underline{n}}\frac{|(\underline{n}n|\rho_m)|^2}{\epsilon_n-\epsilon_{\underline{n}}-Z_m\mathrm{sgn}(\epsilon_{\underline{n}}-\mu)}. \tag{14.6}$$

Where $Z_m=\Omega_m-i\eta$ are the excitation energies shifted infinitesimally into the complex plane. The ρ_m are the corresponding excitation densities. More details, tests and benchmark calculations are can be found in Ref. 34.

14.1.2 *GW* features

The *GW* method is implemented in `TURBOMOLE` in the `escf` module supporting the following features:

- LDA, GGA, meta-GGA and their Hybrid functionals can be used for the underlying DFT calculation.
- Single-shot G_0W_0
- Quasiparticle selfconsistent GW (qsGW)
- Eigenvalue-only selfconsistent GW (evGW)

- In G_0W_0, the linearized, Eqn. (14.3), and solved, Eqn. (14.2), quasiparticle equation.
- Both RPA and TDDFT response functions can be used to screen the Coulomb interaction in constructing W, although we only recommend to use RPA response.
- Closed shell and open shell references are supported in all calculations. Additionally, Kramers-restricted closed shell and also Kramers-unrestricted closed and open shell systems within the two-component relativistic framework (inclusion of spin-orbit coupling) can be treated using $soghf. For open shell two-component calculations collinear and non-collinear approaches are implemented.

14.1.3 General recipe for G_0W_0 calculations

Since `TURBOMOLE` 7.5 two fast GW options are available: Just add $g0w0 OR $evgw to the control file and run `escf` with the -gw flag: escf -gw. This automatically sets all parameters to curated sensible values and will provide very good results and starting points for a BSE calculation in the majority of all cases.

The general recipe for a G_0W_0 and evGW calculations with dRPA response using RI is as follows:

1. `define` session
2. Choose reasonable `cbas` auxiliary basis sets (define)
3. `dscf` or `ridft` calculation
4. Provide $gw OR $rigw flags and keywords
5. To trigger the fast RI algorithm add $rick
6. `escf` calculation

Ad 1) The def2-TZVPP basis seems to be the most useful, it comes for all tested systems within 0.1 eV of the def2-QZVP result with about half the number of basis functions. A cbas must be set, the use of jbas-type fitting bases is discontinued for GW since Turbomole 7.3. In the final define menu select the gw menu to set options for the actual GW calculation.

The gw menu in define will set all needed variables for a gw/rigw calculation. The according $scfinstab and $soes flags will be set, and also the $gw or $rigw flags are written to the control file with the chosen options. Also the $rick flag is set. The control file provided by `define`can usually be used for a calculation without further modifications; which are nevertheless described below of one wishes to modify it manually.

Ad 3) Symmetry up to D_{2h} is available for both closed shell (rpas) and open shell system (urpa) for GW, for $gw and $rigw. Especially in the gw module the use of symmetry leads to large speedups and the user is encouraged to exploit symmetry if possible. Two-component calculations can only be performed in C_1. The $gw module can also handle open-shell 2c calculations, while $rigw is formally limited to Kramers-restricted closed shell molecules in

the 2c case. Moreover, the simplified methods xGW and sGW, which neglect contributions from excitation vectors are available for all point groups implemented in Turbomole.

Ad 4) In the last define step the gw menu can be selected to set up a G_0W_0 (and also $evGW$ and $qsGW$) calculation. There are three distinct versions available for G_0W_0 and evGW:

1. $gw uses spectral representations just as the previous version, which scales as N^6 with system size. This version is to be preferred when full quasiparticle spectra are desired (i.e.: QP energies are also needed for non-valence orbitals, or generally for all). It is compatible with nearly all available starting points:
   (a) scalar/non-relativistic closed-shell systems (up to D_{2h} symmetry)
   (b) scalar/non-relativistic open-shell systems (up to D_{2h} symmetry)
   (c) two-component (2c) Kramers-restricted systems (C_1 only)
   (d) two-component (2c) open-shell systems (C_1 only)
2. $rigw; RI-AC-$G_0W_0$ and RI-AC-evGW variants construct the self-energy Σ_C on the imaginary axis using numerical integration. The obtained result is then analytically continued to the real axis using Pade approximants. This ansatz scales roughly as N^4, and yields reliable results for valence orbitals, especially for HOMO and LUMO quasiparticle energies. RI-AC-GW is the default variant for $rigw calculations. It is available for the following starting points:
   (a) scalar/non-relativistic closed-shell systems (up to D_{2h} symmetry)
   (b) scalar/non-relativistic open-shell systems (up to D_{2h} symmetry)
   (c) two-component (2c) Kramers-restricted systems (C_1 only)
   (d) two-component (2c) open-shell systems (C_1 only)
3. $rigw; RI-CD-$G_0W_0$ and RI-CD-evGW variants construct the self-energy Σ_C using contour deformation. This ansatz scales roughly as N^4 (valence orbitals) - N^5 (core orbitals), and yields reliable results for most orbitals. RI-CD-GW is invoked by the *contour* keyword within the $rigw datagroup. It is available for the following starting points:
   (a) scalar/non-relativistic closed-shell systems (up to D_{2h} symmetry))
   (b) scalar/non-relativistic open-shell systems (up to D_{2h} symmetry))
   (c) two-component (2c) Kramers-restricted systems (C_1 only)
   (d) two-component (2c) open-shell systems (C_1 only)

Especially for 2c calculations the prefactor of 256 for a $gw calculation is reduced to 4-8 in a $rigw calculation, making calculations feasible also for large systems. This prefactor reduction is valid for both $rigw variants, RI-AC-$GW$ and RI-CD-GW. Magnetic fields and other Kramers-unristricted references can be used throughout all GW and BSE calculations since Turbomole 7.7.

In cases where the number of states between a given quasiparticle energy and the gap becomes large or if many quasiparticle energies are to be optimized, the RI-CD-*GW* approach may be approximated, by using a sampling strategy for the frequencies required in the calculation of the dielectric function and residues entering the contour. This approach follows the ideas described in Ref. [273], which employs the analytic continuation approach for the evaluation of most residues. The frequency grid is build by sampling the exact set of frequencies required for RI-CD-*GW*, where a frequency is removed if it is close to an already included grid point. Note that calculations within the generalized two-component framework are limited to the Kramers-symmetric systems using this approximation.

Performance & Accuracy: Since Turbomole 7.3 specialized *GW* algorithms are implemented in `escf`, leading to large speedups. The analytic continuation ($rigw) variant usually yields self-energies with *meV* accuracy compared to standard $gw for HOMO/LUMO orbitals, and is much more conservative with memory and CPU requirements. For two-component calculations (including spin-orbit interactions) also the prefactor of G_0W_0 and *evGW* is largely reduced in $rigw. The standard $gw variant is however accurate and reliable throughout all quasiparticle states, also describing non-valence orbitals well and therefore recommended whenever possible.

Possible source of errors: The DFT options used in `dscf` or `ridft` should not be altered before starting `escf`. Otherwise erratic quasiparticle energies may be obtained. Also any files containing excitations (sing, unrs) from other `escf` runs should be removed prior to the *GW* run. Also a cbas must be set before starting `escf`.

14.2 Excitation energies from the Bethe–Salpeter equation

14.2.1 Theoretical background

The Bethe–Salpeter (BS) excitation energies ω_n are obtained as solutions of a pseudo-Hermitian eigenvalue equation, which has the same form as in time-dependent Hartree–Fock (TD-HF) theory. For the sake of simplicity, only the equations for general references are given below. The orbitals are assumed to be complex.

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}* & \mathbf{A}* \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{Y}_n \end{pmatrix} = \omega_n \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{Y}_n \end{pmatrix} . \tag{14.7}$$

In the following, the indices $a, b, \ldots$ $(i, j, \ldots)$ refer to unoccupied (occupied) spinors, the indices $p, q, r, \ldots$ include occupied and unoccupied spinors. The definitions of the matrices $\mathbf{A}$ and $\mathbf{B}$ differ in the one-particle energies, which are the orbital energies in case of TD-HF and the *GW* quasi-particle energies $\varepsilon^{\mathrm{QP}}$ in case of BS. Also, the exchange interaction is screened in case of BS,

$$A_{ia,jb} = \Delta\varepsilon^{\mathrm{QP}}_{ia,jb} + (ai|jb) - (j\bar{i}|ab) , \tag{14.8}$$

$$B_{ia,jb} = (ai|bj) - (j\bar{a}|ib) . \tag{14.9}$$

The exchange interaction $(pq|\bar{r}s)$, which is defined as

$$(pq|\bar{r}s) = \sum_{tu} (\epsilon^{-1})_{pq,tu} (tu|rs) \Leftrightarrow \mathbf{W} = \epsilon^{-1}\mathbf{v} \;, \tag{14.10}$$

is screened by the inverse of the dielectric function,

$$\epsilon_{pq,rs} = \delta_{pr}\delta_{qs} - (pq|rs)\left[\chi_0(\omega=0)\right]_{rs,rs} \;, \tag{14.11}$$

$$\left[\chi_0(\omega=0)\right]_{rs,rs} = \sum_{ia} \frac{\delta_{ra}\delta_{si} + \delta_{ri}\delta_{sa}}{\varepsilon_i^{\mathrm{QP}} - \varepsilon_a^{\mathrm{QP}}} \;, \tag{14.12}$$

which depends on the *GW* quasi-particle energies. It can be expressed in terms of an infinite-order perturbation expansion in the Coulomb interaction,

$$\begin{aligned} \mathbf{W} =& \epsilon^{-1}\mathbf{v} \\ =& \mathbf{v} + \mathbf{v}\chi_0(\omega=0)\mathbf{v} + \mathbf{v}\chi_0(\omega=0)\mathbf{v}\chi_0(\omega=0)\mathbf{v} + \dots \;. \end{aligned} \tag{14.13}$$

To obtain $\mathbf{W}$, the RI-approximation is introduced for the two-electron integrals. As a direct result, the inversion ϵ is replaced by the inversion of a matrix in the auxiliary basis:

$$\begin{aligned} \mathbf{W} \approx \mathbf{b}^\top \tilde{\mathbf{i}}\, \mathbf{b} =& \mathbf{b}^\top \left(\mathbf{1} + \mathbf{i} + \mathbf{i}^2 + \mathbf{i}^3 + \dots\right)\mathbf{b} \\ =& \mathbf{b}^\top \left(\mathbf{1} - \mathbf{i}\right)^{-1}\mathbf{b} \;, \end{aligned} \tag{14.14}$$

$$\mathbf{i} = 2\mathbf{b}\chi_0(\omega=0)\mathbf{b}^\top \Leftrightarrow i_{PQ} = 2\sum_{ia} \frac{b_{P,ai} b_{Q,ai}}{\varepsilon_i^{\mathrm{QP}} - \varepsilon_a^{\mathrm{QP}}} \;. \tag{14.15}$$

For further details, we refer to Ref. [33].

14.2.2 BSE features

The BSE method is implemented in the `escf` module. It is activated by the keyword `$bse`. Its features and different variants are discussed in the following.

Reference determinant

The BSE method supports both Hartree–Fock and Kohn–Sham references, that is, orbitals obtained from a `dscf` or `ridft` computation. Note that symmetry is only supported up to D_{2h} and its subgroups. In case of wave functions of higher symmetry, the orbitals have to be converted using `define`'s option `susy` to a subgroup of D_{2h}. Two-component references ($soghf) including spin-orbit interactions can be used.

Excitations

Singlet, triplet and spin-unrestricted excitation energies based on the Bethe–Salpeter equation are available in `escf`. Similar to the RPA and TDA variants in TD-HF, the eigenvalues of the full 2×2 "super-matrix" can be computed, or of $\mathbf{A}$ only. To define these options, the same input blocks as for a TD-HF computation are read in: `$scfinstab` and `$soes`.

See Section 8 for details. For two-component calculations only the TDA variant is implemented. Note that 2c-BSE is, compared to TD-DFT, not limited to closed shell cases but also implemented for open shell cases.

Quasiparticle energies

The BSE method reads in quasiparticle energies obtained from a preceding *GW* calculation. It supports all different *GW* methods available in `escf`. Since Turbomole 7.3 the number of frozen orbitals can be different in the *GW* and *BSE* step. However qpenergies.dat files from Turbomole 7.2 and earlier are not directly compatible with Turbomole 7.3 and higher, and it is recommended to redo the *GW* calculation. The reason for this is that the quasiparticle states for frozen orbitals are also projected since Turbomole 7.3, and always all orbitals are therefore updated and reported in the qpenergies.dat file.

Quasiparticle energies in A and W

By default, the matrices **A** and **W** are set up using quasiparticle energies. To set up **A** using KS/HF orbital energies instead, i.e.

$$A_{ia,jb} = \Delta\varepsilon_{ia,jb} + (ai|jb) - (ji|ab) \;, \tag{14.16}$$

the option `noqpa` has to be added to the `$bse` block in `control`. Similarly, if the denominator in χ_0 and, consequently, the static screened interaction should be constructed using KS/HF orbital energies instead of *GW* quasiparticle energies, the option `noqpw` has to be set. The quasiparticle energies are read in from the file `qpenergies.dat`, which has to be created in a preceding *GW* calculation using `escf`. A different file name can be defined using the option `file` in the `$bse` block.

Computation of the static screened interaction

The static screened interaction is obtained from the auxiliary matrix $\tilde{\mathbf{i}}$. By default, this is computed by Cholesky decomposition and inversion of $(\mathbf{1} - \mathbf{i})^{-1}$. An alternative iterative procedure,

$$\tilde{\mathbf{i}}^{(i+1)} = \mathbf{1} + \mathbf{i}\,\tilde{\mathbf{i}}^{(i)} \;, \tag{14.17}$$

is activated by the option `iterative`. The convergence threshold (default: `1.0d-12`) and maximum number of iterations (default: 100) can be adjusted by the options `thrconv` and `iterlim`, respectively.

Auxiliary basis sets

The static screened interaction is computed using the RI approximation, therefore, an auxiliary basis set is required. Since the integrals are similar to those in MP2, the `cbas` auxiliary basis sets are recommended. They can be conveniently defined using the `cc` sub-menu of `define`.

14.2.3 General recipe for a BSE calculation

1. `dscf` or `ridft` calculation to converge reference orbitals
2. Provide $gw or $rigw control flags and perform a *GW* calculation
3. Choose reasonable `cbas` auxiliary basis sets if not done in the *GW* step
4. `escf` calculation to obtain `qpenergies.dat`
5. Remove $gw and $rigw control flags
6. Set BSE control flags
7. Add the keyword $rik or $rick if not already present
8. `escf` calculation to obtain excitation energies

Chapter 15

Calculation of Vibrational Frequencies and Vibrational Spectra

Calculation of second derivatives of total energies leads to the molecular Hessian, which enables prediction of vibrational frequencies and infrared spectra (within the harmonic approximation) as well as the application of improved algorithms for geometry optimization and transition state search.

The `aoforce` module calculates analytically harmonic vibrational frequencies within the HF- or (RI)DFT-methods for closed-shell- and spin-unrestricted open-shell-systems. Broken occupation numbers would lead to results without any physical meaning. Note, that RI is only used partially, which means that the resulting Hessian is only a (very good) approximation to exact second derivatives of the RIDFT-energy expression. Apart from a standard force constant calculation which predicts all (symmetry allowed and forbidden) vibrational transitions, it is also possible to specify certain irreps for which the calculation has to be done exclusively or to select only a small number of lowest eigenvalues (and eigenvectors) that are generated at reduced computational cost.

Furthermore, the `NumForce` script allows the calculation of second derivatives for all methods for which a program for analytic gradients is available in `TURBOMOLE`, i.e. the main use of this script is the prediction of vibrational spectra at the MP2 level and for excited states using RI-CC2 or TDDFT.

If force constant calculations result in imaginary frequencies, molecular distortions along these normal modes should lower the energy. To distort the molecule, use the interactive module `vibration`, output of the new coordinates is done to the general input file on `$newcoord`.

Vibrational frequencies also enable calculation of the molecular partition function and thus prediction of thermodynamic functions at temperatures other than 0 K and finite pressure

(within the assumption of an ideal gas and no coupling between degrees of freedom). These functions can be obtained with the interactive module `Freeh`, results are printed to standard I/O.

Prerequisites

1. Both `aoforce` and even more `NumForce` require well converged SCF-/DFT-calculations (e.g. `$scfconv 8` and `jobex [-ri] -gcart 4`).
2. The maximum core memory the program `aoforce` is allowed to allocate should be defined in the data group `$maxcor`; the recommended value is about 50% of the available (physical) core memory (in case of RI-calculations subtract the memory specified in `$ricore`).
3. To start `aoforce` in the lowest eigenvalue search mode, use the keyword `$les`. For its use as well as other keywords dealing with the calculation of only some irreps, see the reference guide part of this manual.
4. `NumForce` additionally requires the file `gradient` and will not work, if the calculation is not done at a stationary point of the molecular total energy. For reliable results, always use `NumForce` with the option `-central` (i.e. central differences) and be aware of effects due to the step length (option `-d` *real*;, default value is 0.02 a.u.). It is strongly recommended to use `NumForce` in DFT calculations only with the option `weight derivatives` in `$dft`, since this provides more accurate gradients and thus frequencies, see Section 23.2.15.
5. The `NumForce` script can be run for different levels of theory, which means that the binaries it calls have to be specified additionally. To perform calculations using the RI approximation, call `NumForce` with the option `-ri`. MP2 and CC2 calculations are requested via the options `-level mp2` and `-level cc2`, respectively. `NumForce` works also on the RI-RPA level with `-level rirpa` (note: the `-ri` option must be used in this case). To select the correct option(s), use the explanations you get by calling `NumForce -h`.

For a review of theory and implementation see refs. [274, 275].

Limitations

The `aoforce` code has presently a number of limitations one should be aware of:

- It can only handle basis sets up to at most g functions.
- Point groups with reducible E-representations (such as C_n and C_{nh} with $n \geq 3$, S_n with $n \geq 5$, or T and T_d)
- Frozen internal or Cartesian coordinates are not recognized. `aoforce` will always evaluate the full Hessian matrix.

15.1 Analysis of Normal Modes in Terms of Internal Coordinates

A note in advance: The analysis of normal modes can (at nearly no computational cost) always be redone as long as you keep a copy of the file `hessian`.

A general prerequisite for this option is that you have defined a set of non-redundant coordinates for all 3N-6 (3N-5) degrees of freedom of your molecule. To make sure that this is the case, you should switch off redundant coordinates (currently, this is only possible by manually removing the data group `$redundant` and also removing the entry `redundant on` in `$optimize`). Run `define` to generate non-redundant coordinates by using the `iaut` command in the internal coordinate menu (or by creating them manually via `idef`). We recommend to use the `irem` command first to delete all previous definitions of internal coordinates. See Section 4 for further details. If the molecules point group is not C_1, `define` will set some of the coordinate to status `d` (display) or `i` (ignore). Use the `ic` command to change all coordinates to `k`. You can also achieve this by editing in the `$intdef` data-group manually.

The analysis in internal coordinates is switched on by adding a line in the data-group `$drvopt` that has the following syntax:

`analysis [only] intcoord [print` *print-level*`]`

Keywords in square brackets are optional. If `only` is added, the program assumes that the file `hessian` exists and runs only the analysis part of `aoforce`. The program will give the following output (controlled by the print level given in parenthesis):

- diagonal elements of the Hessian in internal coordinates (force constants of bonds, angles, etc.) (print level 0)
- complete force constant matrix in internal coordinates (print level 2)
- normal modes in terms of internal coordinates (print level 1)
- Potential energy contributions $\tilde{V}_{ij}^n$, defined as

  $$\tilde{V}_{ij}^n = L_i^n L_j^n F_{ij}/\omega^n$$

  where L_i^n are the elements of the normal coordinate belonging to mode n and F_{ij} are the elements of the force constant matrix, both expressed in the internal coordinate basis; ω is the related eigenvalue. The program will list the diagonal contributions $\tilde{V}_{ii}^n$ (print level 1), the off-diagonal contributions $\tilde{V}_{ij}^n + \tilde{V}_{ji}^n = 2\tilde{V}_{ij}^n$ (print level 2 for up to 10 atoms, else print level 10) and the gross contributions $\sum_i \tilde{V}_{ij}^n$ (print level 1).
- Based on these quantities, the program will give an assignment of normal modes by listing all internal coordinates with large diagonal or gross contributions (print level 0).

Note that for large molecules or complicated topologies the B-matrix (that is used to transform from Cartesian coordinates into internal coordinates and vice versa) may become singular. In this case only the normal modes in the internal coordinate basis can be listed.

15.2 Calculation of Raman Spectra

Vibrational Raman scattering cross sections are computed in the approximation of the polarizability theory from derivatives of the frequency-dependent polarizability tensor with respect to normal modes of vibration,

$$\left(\frac{d\sigma}{d\Omega}\right) = k_\omega \left(c_i \alpha'^2(\omega) + c_a \gamma'^2(\omega)\right) .$$

Here, $\alpha'^2(\omega)$ and $\gamma'^2(\omega)$ denote the isotropic part and the anisotropy of the differentiated polarizability tensor, respectively. The coefficients c_i and c_a depend on the scattering geometry and the polarization of the incident and scattered radiation. The factor

$$k_\omega = \frac{\hbar}{4\pi\epsilon_0^2 c^4} \frac{(\omega - \omega_v)^4 g_v}{2\omega_v}$$

includes the frequency ω_v and the degeneracy g_v of the vibration. c is speed of light and ϵ_0 stands for the dielectric constant of vacuum.

Computation of Raman spectra with `TURBOMOLE` is a three-step procedure. First, vibrational frequencies and normal modes are calculated by `aoforce`. Cartesian polarizability derivatives are computed in the second step by `egrad`, see Section 8.4.10. Finally, the program `intense` is used to project the polarizability derivatives onto vibrational normal modes and to compute Raman scattering cross sections which are written out along with vibrational frequencies and normal modes. The script `Raman` can be used to perform all these steps automatically.

15.3 Calculation of VCD Spectra

VCD intensities are proportional to the rotational strengths, which are defined as scalar product between the electric and magnetic transition dipole moments. The rotational strength for the transition of the nth vibrational normal mode in the electronic ground state is

$$R_n = \mathrm{Im}(\boldsymbol{\mu}_{el,n} \cdot \boldsymbol{\mu}_{mag,n}).$$

Within the harmonic approximation, the electric and magnetic transition dipole moments can be written as

$$(\boldsymbol{\mu}_{el,n})_\beta = \sqrt{\frac{\hbar}{\omega_n}} \sum_{\lambda\alpha} P^\lambda_{\alpha\beta} S^\lambda_{\alpha,n}$$

$$(\boldsymbol{\mu}_{mag,n})_\beta = -\sqrt{2\hbar^3\omega_n} \sum_{\lambda\alpha} M^\lambda_{\alpha\beta} S^\lambda_{\alpha,n}.$$

$P^{\lambda}_{\alpha\beta}$ is the so called *atomic polar tensor* (APT), $M^{\lambda}_{\alpha\beta}$ the so called *atomic axial tensor* (AAT) and $S^{\lambda}_{\alpha,n}$ the transformation matrix from Cartesian coordinates to normal coordinates. ω_n is the frequency of the nth vibrational normal mode, α and β describe Cartesian coordinates and λ labels the atom nuclei.

Both the APT and the AAT are divided in an electronic and a nuclear contribution. According to Chem. Phys. Lett. 252, 221 (1996), the electronic contribution of the AAT within the *coupled perturbed Hartree-Fock* (CPHF) formalism is given by

$$I^{\lambda}_{\alpha\beta} = \sum_{i=1}^{N_{\mathrm{occ}}} \sum_{\mu,\nu=1}^{N_{\mathrm{bf}}} \Bigg[c_{\mu i} c_{\nu i} \langle \chi_{\mu}^{R^{\lambda}_{\alpha}} | \chi_{\nu}^{B_{\beta}} \rangle + \sum_{p=1}^{N} c_{\mu i} c_{\nu p} U_{ip}^{B_{\beta}} \langle \chi_{\mu}^{R^{\lambda}_{\alpha}} | \chi_{\nu} \rangle + \sum_{p=1}^{N} c_{\mu p} c_{\nu i} U_{ip}^{R^{\lambda}_{\alpha}} \langle \chi_{\mu} | \chi_{\nu}^{B_{\beta}} \rangle + \sum_{p,q=1}^{N} c_{\mu p} c_{\nu q} U_{ip}^{R^{\lambda}_{\alpha}} U_{iq}^{B_{\beta}} \langle \chi_{\mu} | \chi_{\nu} \rangle \Bigg].$$

To compute VCD spectra with `TURBOMOLE`, `mpshift` and subsequently `aoforce` have to be called. First, if `$vcd` is set, `mpshift` calculates the disturbed $U^{\boldsymbol{B}}_{ip}$. In the second step vibrational frequencies, normal modes and VCD rotational strengths are calculated by `aoforce`. At present, calculation of VCD spectra is possible at Hartree-Fock and DFT level (for LDA, GGA and hybrid-GGA functionals) in C_1 symmetry. Since the angle between the electric and magnetic transition dipole moments is not gauche invariant, it will only be printed out if the distance between the center of mass and the origin is smaller than 1e-03 a0. The script `VCD` can be used to perform these two steps automatically.

15.4 Vibrational frequencies with fixed atoms using `NumForce`

The `NumForce` script provides with the option `-frznuclei` a possibility to do a vibrational analysis with fixed atoms. The atoms for which the Cartesian coordinates should frozen have to be marked in `$coord` with a "f" behind the atom type. The frozen coordinates will be skipped during the numerical evaluation of the force constant matrix; instead all off-diagonal elements of the force constant matrix which refer to one or two frozen coordinates will be set to zero, while the diagonale elements for the frozen coordinates will be set to an arbitrarily chosen large value.

This feature is mainly intended to allow for a vibrational analysis in embbeded cluster calculations e.g. for defects in ionic crystals. The vibrational analysis uses a kind of "frozen phonon" approximation which corresponds to setting the masses of the fixed atoms to infinity, i.e. decoupling the fixed atoms mechanically from the "mechanically active" subsystem. The resulting vibrational frequencies will thus only provide good approximations to the true (harmonic) frequencies for such modes for which the mechanical coupling to the embedding environment is negligible. In particular the frequencies of stretch modes which involve bonds between the "mechanically active" subsystem and atoms with frozen coordinates will be strongly affected by this approximation.

Note:

- The `-frznuclei` is not compatible with the polyhedral difference algorithm. It can only be used with central differences which should be enforced with the `-central` option.
- If the option `-frznuclei` is switched on, the program assumes that the constraints enforced by fixing coordinates remove the six external degrees of freedom for on overall rotation or translation of the system and therefore the hessian matrix is not projected onto the subspace of internal coordinates. Fixing the coordinates of only one or two atoms might does lead to some artificial small, but non-zero frequencies.
- Zero-point vibrational energies calculated with the `-frznuclei` option are only meaningful for comparison of systems with the same mechanically active atoms and similar embedding, as the contributions from the frozen coordinates are not included.

15.5 Interface to hotFCHT

`aoforce` supports the generation of input files for the hotFCHT code (version 2.0 and later) of R. Berger and co-workers, see hotFCHT, which allows for the calculation of Franck-Condon factors. Just include the keyword `$hotfcht` in the control file. The option is also active in analysis mode, that is as long as you still have the data group `$hessian` (in the control file or in a file referenced in the control file) you can always use `aoforce` (in analysis mode) to quickly generate the hotFCHT input. The program will write three files. The first one, `hotfcht_header.inp` contains a collection of the most important keywords of hotFCHT (set to some default values, please adapt to your needs) and list of all atomic masses (either TURBOMOLE's default masses or the ones given in the `$atoms` data group). The other two, `hotfcht_data_i.inp` and `hotfcht_data_f.inp` contain the vibrational frequencies, normal modes and the names of the irreducible representations of the normal modes. In the former file, these data are associated with the hotFCHT keywords for the initial state, while the latter file contains the same data, but associated with the keywords for the final state. In order to run a hotFCHT calculation, you need to optimize the structures of two electronic states (usually the electronic ground state and an excited or ionized state) and obtain the harmonic force fields for both, using either aoforce or NumForce. In order to generate the hotFCHT input, just concatenate the `hotfcht_header.inp` file (from any of the two calculations) and the `hotfcht_data_i.inp` file from the calculation that refers to the initial state (e.g. the ground state in case of an absorption spectrum) and the `hotfcht_data_f.inp` file from the calculation of the final state (the excited state in case of an absorption spectrum). Carefully edit the keywords in the header of the resultant file and run hotFCHT (please, refer to the hotFCHT documentation for further information).

Chapter 16

First order electron-vibration coupling

16.1 Theoretical background

At the effective single-particle level, the Hamiltonian of the coupled system of electrons and vibrations is given by [276]

$$\hat{H} = \hat{H}^{\mathrm{e}} + \hat{H}^{\mathrm{v}} + \hat{H}^{\mathrm{ev}}, \tag{16.1}$$

where the first term $\hat{H}^{\mathrm{e}}$ describes the electronic system and the second term $\hat{H}^{\mathrm{v}}$ the vibrational degrees of freedom. The last term in the Hamiltonian

$$\hat{H}^{\mathrm{ev}} = \sum_{\mu\nu}\sum_{\alpha} \hat{d}^{\dagger}_{\mu}\lambda^{\alpha}_{\mu\nu}\hat{d}_{\nu}(\hat{b}^{\dagger}_{\alpha} + \hat{b}_{\alpha}) \tag{16.2}$$

describes the first order electron-vibration (EV) interaction. The EV coupling constants are given as

$$\lambda^{\alpha}_{\mu\nu} = \left(\frac{\hbar}{2\omega_{\alpha}}\right)^{1/2} \sum_{\chi} \langle\mu| \frac{\mathrm{d}\hat{H}^{\mathrm{e}}_{1}}{\mathrm{d}\chi} |\nu\rangle \mathcal{A}^{\alpha}_{\chi}, \tag{16.3}$$

where $\chi = (k, u)$ is a shorthand notation that refers both to the displacement of atom k from the equilibrium value of the position $\vec{R}_k$ along the Cartesian component $R_{k,u}$ with $u = x, y, z$ as well as the index pair itself. Furthermore, $\mathcal{A}^{\alpha}_{\chi} = \mathcal{C}^{\alpha}_{\chi}/\sqrt{M_k}$ are the mass-normalized normal modes, obtained from the eigenvectors $\mathcal{C}^{\alpha}_{\chi}$ of the dynamical matrix as calculated from the `aoforce` module [276].

16.2 evib features

The evib module, implemented in TURBOMOLE, allows to calculate the matrix elements of the first order derivative of the Kohn-Sham operator with respect to atomic displacements χ

$$H^e_{\mu\nu,\chi} = \langle\mu|\frac{\mathrm{d}\hat{H}^{\mathrm{e}}_1}{\mathrm{d}\chi}|\nu\rangle, \tag{16.4}$$

which are required to obtain the first order EV coupling constant, as given in Eq. (16.3) [276].

Limitations:

- only c1 symmetry at the moment,
- RI-approximation only partely implemented.

16.3 General usage of evib

Calculating the matrix elements given in Eq. (16.4) consists of two steps. First a force constant calculation using aoforce is performed, where the following control flags have to be added:

```
$nosalc
$sijuai_out
```

,

this will save the derivative of the density matrix, which are necssary for the subsequent evib run.

The matrix elements of $H^e_{\mu\nu,\chi}$ (Eq. (16.4)) are stored by default in binary format in the file dfdxi.dat (dfdxi_a.dat and dfdxi_b.dat for UHF), using formatted Fortran output with a record length of 8 bytes for each matrix element. The matrix elements $H^e_{\mu\nu,\chi}$ with $\chi = (k, u)$ are in the atomic orbital (AO) and ordered as follows: (i) Cartesian component $u = x, y, z$, (ii) atom number k, and (iii) $(N_{\mathrm{ao}} + 1) * N_{\mathrm{ao}}/2$ matrix elements of the upper part of the triangular matrix for the $\mu\nu$ indices.

Optionally,

```
$dfdxi textout
```

can be used to generate text output of the matrix elements.

Additonally the first derivatives of the orbital energies (Kohn-Sham eigenvalues) with respect to the atomic displacements $\mathrm{d}\epsilon_i/\mathrm{d}\chi$ are calculated and stored in the text file dEidR.dat (dEidR_a.dat and dEidR_b.dat for UHF).

Chapter 17

Calculation of NMR Shieldings

The program `mpshift` calculates nuclear magnetic resonance (NMR) shielding constants using the GIAO (Gauge-Including Atomic Orbital) method.

At present the following methods are implemented:

HF-SCF — the coupled-perturbed Hartree–Fock (CPHF) equations in the AO basis are solved using a semi-direct iterative algorithm [39] similar to `dscf`. The (multipole-accelerated-) resolution-of-the-identity-fitting approximation is available for the Coulomb term (MARI-J) [44]. Due to the effective screening approach the exchange part shows a low-order scaling. Also available for open-shell calculations [83, 277]. Full OpenMP support.

2c HF-SCF — the generalized two-component CPHF equations are solved as outlined in [37, 43]. The RI-J and the seminumerical exchange approximation are commended for efficiency. Full OpenMP support.

DFT — using either non-hybrid functionals where no iterations are needed for the coupled-perturbed Kohn–Sham (CPKS) equations [278] or hybrid functionals where the same algorithm as at the HF-SCF level is used. Meta-GGAs and the XCfun library can be further used [44]. Moreover, LibXC and range-separated functionals are supported, see also chapter 6. Meta-GGAs and local hybrid functionals require the generalized kinetic energy density. By default, this is done with the external vector potential [279]. However, it is also possible to use the paramagnetic current density. We strongly recommend to use the current-dependent generalization for meta-GGAs and local hybrid functionals in pNMR calculations (`$curswitchengage`) [125, 280, 281]. Also available for open-shell calculations [83, 277, 281]. The current-dependent version is strongly recommended for the Minnesota functionals and also advantageous for the SCAN functional family. Full OpenMP support.

2c DFT — using semilocal or (range-separated) hybrid functionals in a relativistic two-component formalism [43]. Note that this requires to iteratively solve the CPKS equations even for non-hybrid functionals due to spin–orbit effects and the non-vanishing spin-current densities of the ground state. The XC kernel can be neglected with the keyword `$nmr_ziegler` as suggested in [282]. However, including the kernel is recommended for the Spin-Orbit Heavy Atom on the Light Atom (SO-HALA) effect. The XC kernel should always be included for hybrid DFT calculations, as hybrid functionals require an iterative procedure. Note that the current-dependent formalism for meta-GGAs is not available with GIAOs. Full OpenMP support for both CGO and GIAOs.

MP2 — semi-direct method, see ref. [40] and [283]. In contrast to HF and DFT, MP2 only comes with limited OpenMP capabilities.

The following Hamiltonians are available in addition to the usual non-relativistic one:

ECP — In molecules with ECP-carrying atoms, chemical shieldings on all the other atoms can be computed with `mpshift` in the way suggested by van Wüllen [284]. ECPs can be used to treat scalar-relativistic effects on neighboring atoms [45]. Spin–orbit ECPs are not available.

X2C — A scalar-relativistic or spin-free all-electron exact two-component Hamiltonian can be used to calculate the NMR shielding tensor of heavy elements [41]. A finite nucleus model based on a Gaussian charge distribution is available for the scalar potential and the vector potential (`$finnuc`). This model can be also used with the DLU-X2C Hamiltonian below. Grids with an increased number of radial points [156] (e.g. gridsize 4a) should be used. Moreover, it is recommended to use NMR-tailored basis sets, i.e. the x2c-XVPall-s (X=S,TZ) type bases. These employ additional tight functions to accurately sample the density in the vicinity of the nuclei.

DLU-X2C — A local X2C Hamiltonian (`$rlocal`) is further available and recommended. The diagonal local approximation to the unitary decoupling transformation (DLU) is employed. This results in a very efficient algorithm. The error introduced by the DLU scheme is negligible. For details on the theory, implementation, and application please see ref. [41].

2c X2C — Spin–orbit two-component version of the previously mentioned X2C Hamiltonian. Self-consistent treatment of spin–orbit interaction. Can be performed with a common gauge-origin (`$cgo_nmr`) [42] or gauge-including atomic orbitals (default, no specific keyword) [43]. DLU scheme is also available for efficiency (`$rlocal`). The finite nucleus model (`$finnuc`) and the mSNSO approximation (`$snso`) are further recommended.

17.1 Prerequisites

1. `mpshift` needs converged MO vectors from a SCF or DFT run (`dscf` or `ridft`). The 2c runs need converged spinor vectors from an `ridft`calculation. The flag `$coulex` can set to use analytical Coulomb integrals in 2c `ridft`and `mpshift`calculations.
2. For SCF or DFT calculations, no NMR specifications have to be made in the `control` file.
3. To perform an MP2 calculation of the NMR shieldings, you have to prepare the input with `mp2prep -c`.

17.2 How to Perform a SCF or DFT Calculation

All you have to do for running `mpshift` is typing `mpshift` at the shell level.

The results of a SCF or DFT calculation (the trace of the total shielding tensors, its anisotropy and the CPHF contribution for each symmetry distinct atom) are written into the `control` file after the keyword `$nmr <rhf/uhf/ghf/dft> shielding constants`.

This data group is write-only for `mpshift`, but you can utilize it for graphical rendering of the calculated NMR spectra and for a quick overview of the results. A more detailed output with the complete shielding tensors can be found in the output of `mpshift`, so it is recommended to put the output in a file when calling the program.

For version 7.5 the solver for the CPHF equations has been reimplemented based on the Davidson algorithm used by `escf` and `aoforce`. This utilizes the keyword `$shiftconv` (default 7, i.e. a residuum threshold of 10^{-7}) to check for the convergence of the perturbed orbitals and density. Keywords for the calculation with non-default settings are given in Sec. 23.2.30. Many new features made available with version 7.5 and later such as paramagnetic NMR only support the Davidson solver. The 2c part of the program employs the Davidson solver of Ref. [37] and the same default for the norm of the residuum.

For large-scale calculations, please adapt the data group `$maxcor` in the control file. This will allow for a more efficient batching in the Davidson algorithm.

17.3 How to Perform a MP2 calculation

To perform an MP2 calculation of the NMR shieldings you have to prepare the input with `mp2prep -c`.

`mpshift` will then calculate both the SCF and MP2 shielding constants. The result is written into the `control` file after the keyword `$nmr mp2 shielding constants`. In addition, the HF shielding constants are given after `$nmr rhf shielding constants`.

The script `mp2prep` will create the keywords

```
$csmp2
$thize      .10000000E+10
$mointunit
 type=intermed unit=61 size=0 file=halfint
 type=1112     unit=63 size=0 file=moint#1
 type=1122     unit=64 size=0 file=moint#j
 type=1212     unit=65 size=0 file=moint#k
 type=1212a    unit=70 size=0 file=moint#a
 type=gamma#1  unit=71 size=0 file=gamma#1
 type=gamma#2  unit=72 size=0 file=gamma#2
 type=dtdb#1   unit=76 size=0 file=dtdb#1
 type=dtdb#2   unit=77 size=0 file=dtdb#2
$traloop 1
$statistics mpshift
```

and starts a statistics run of mpshift (by calling `mpshift`). If the resulting disk space requirement exceeds the automatically detected free disk space on your system, it will increase `$traloop` and run a statistics run again. This will be done as long as your free disk space is not sufficient for the calculation.

If the `mp2prep` script fails to run on your system, try to use the -p option or do the procedure described above by hand. Call `mp2prep -h` for more information about `mp2prep`.

17.4 Chemical Shifts

NMR shifts are obtained by comparing nuclear shieldings of your test compound with a reference molecule ($\delta_{\text{subst}} = \delta_{\text{ref}} + \sigma_{\text{ref}} - \sigma_{\text{subst}}$). Therefore, you have to choose a reference molecule with a well-known shift for which you can easily calculate the absolute shielding constant. This implies a certainty about the geometry, too. Furthermore, you have to use the very same basis set for corresponding atoms to minimize the basis set influence. Please note that the output is already given in units of ppm.

Keywords for the Module Mpshift

A list of keywords for the module `mpshift` can be found in Section 23.2.30.

17.5 Further Details: Paramagnetic NMR and 2c NMR

- Two-component calculations require X2C keywords and `$soghf`, see also chapter 6.4. By default gauge-including atomic orbitals are used. The common gauge-origin variant is used with `$cgo_nmr` and the common gauge origin is specified with `$cgo` followed by the number of the atom. If atom number zero is given, then the center of mass is chosen as the gauge origin. By default, the heaviest atom is used as the gauge origin. The GIAO version is used by default, i.e. without setting `$cgo_nmr`.
- Open-shell calculations (pNMR) under consideration of Fermi contact (FC), spin dipole (SD), and spin–orbital paramagnetic spin–orbit (PSOSO) interactions can be performed at the Hartree–Fock and density functional level of theory. Here, the g-tensor is calculated by default. See Sec. 18 for details. For details regarding the theoretical background see [83, 277, 285–287]. The local (X2C) Hamiltonian is available for all terms including the derivatives of the decoupling and the renormalization matrix [83, 277, 286, 287]. For details regarding usage see Section 23.2.30.
- pNMR shieldings of systems with more than one unpaired electron necessitate the inclusion of the zero-field splitting (ZFS) tensor, see Sec. 18 for details. By default, the ZFS tensor is not calculated, i.e. it has to be requested explicitly (e.g. `$pnmr zfs`). Then, the ZFS tensor is computed, however, it is not automatically included in the hyperfine contribution of the pNMR shielding tensor. The pNMR shielding tensor including the ZFS contribution can be computed as outlined in [288] based on the scripts of https://github.com/jautschbach/pnmr-shift. The input for this PNMRShift program is prepared with `$printpnmr`.

17.6 Other Features

- `mpshift` is parallelized by OpenMP. The corresponding environment variables have to be set previously to running `mpshift`.
- The `mpshift` program can be restarted at any stage of computing, since all intermediate results are written into the file `restartcs`. In case of an external program abort you have to remove the `$actual step` flag (by the command `actual -r` or using an editor). `mpshift` analyses this file and decides where to continue.
- Vibrational circular dichroism (VCD) spectra can be calculated using the `gallier` script utilzing the AOFORCE module [45].
- The conductor-like screening model (COSMO) to account for counterions and solvation effects can be selected [44].
- NMR shielding tensors can be calculated for given nuclei only by setting `$nucsel` in the control file. The keyword `$nucsel` can be used followed by the number of the nucleus of interest, e.g., `$nucsel 1,3,7`. Alernatively, you can set the element, e.g., `$nucsel "c","h"`.

- The default maximum number of iterations for the CPHF procedure is set to 30. It can be increased by adapting `$csmaxiter` in the control file.
- Seminumerical exchange can be used for substantial speed-ups in HF and hybrid DFT calculations, i.e. `$esenex` is recommended for large-scale calculations.
- Nucleus-independent chemical shifts (NICS) can be calculated be setting `$nics` in the control file and then listing the Cartesian coordinates in atomic units just as the coordinates of the molecule itself. For details, see Section 23.2.30.
- The (un)perturbed density matrix can be stored on disk by setting `$gimic` in the control file. These matrices are required as input for the gauge-including magnetically induced currents (GIMIC) method [289,290], see https://github.com/qmcurrents/gimic/. This method allows to study electron delocalization pathways and to estimate the degree of aromaticity [291], or to compute the magnetizability [125, 292].

17.7 Known Limitations

- Molecular point groups that contain reducible e representations are not supported (C_n, $C_{n\mathrm{h}}$ S_n, T, and T_h with $n > 2$).
- The following features of `mpshift` are not available for open-shell systems: Calculations of shieldings at the MP2 level of theory, usage of the old CPHF/CPKS solver, calculation of VCD spectra. NICS calculations can only be performed for the orbital contribution of the shielding tensor. Two-component calculations are restricted to closed-shell systems.
- The current-dependent generalization for Meta-GGAs is not yet available for 2c NMR calculations with GIAOs.

Chapter 18

EPR Properties

The electron paramagnetic resonance (EPR) spectra are characterized by the electron–nucleus hyperfine coupling (HFC) and the g-tensor. For systems with a spin $S > 1/2$, the zero-field splitting (ZFS) tensor is important as well. Additionally, the electric field gradient is used to compute the nuclear quadrupole interaction tensor. These quantities are calculated with the modules `dscf`, `ridft`, and `mpshift`. The HFC and g-tensor are also used for the contact and the pseudo-contact term of paramagnetic NMR shielding constants and shifts. Here, we describe the computational work flow, while we refer to Refs. [277], [83], [286], [287], and [293] for the theory.

The HFC, the g-tensor, and the EFG are available within a non-relativistic or scalar-relativistic one-component framework and a spin–orbit two-component framework. For the g-tensor and the paramagnetic spin–orbit part of HFC tensor, spin–orbit perturbation theory is applied in the one-component framework. The ZFS tensor is only available within a non- or scalar-relativistic one-component framework using spin–orbit perturbation theory. Effective core potentials (ECPs) should not be used to compute the HFC, ZFS, or g-tensor, as this may lead to unphysical results due to the lack of explicit core electrons.

If you are interested in all EPR parameters, we recommend to use the flag `$epr` and the `mpshift` module. In a one-component framework, this requires three linear response frameworks, i.e. the same-spin spin–orbit and spin-flip spin–orbit perturbations for the HFC and ZFS tensors as well as the magnetic field treatment for the g-tensor. For convenience, the EPR calculations support a restart option at all stages. With the two-component version, this requires to run three non-collinear SCF calculations as described below. In case of a two-component calculation, the ZFS tensor is not calculated.

18.1 Hyperfine Coupling Constant

One-component treatment: In a non-relativistic or scalar-relativistic framework [277], the HFC tensor is calculated with the `mpshift` module. The HFC tensor consists of the Fermi-contact (FC), the spin–dipole (SD), and the spin–orbital paramagnetic spin–orbit (PSOSO) contribution. The FC and SD term are directly computed from the SCF density matrix. The PSOSO term requires the first-order density matrix with respect to the electron spin. This is evaluated with a spin–orbit perturbation term and the coupled-perturbed Hartree–Fock or Kohn–Sham equations [83]. After the calculation, the HFC tensor is transformed to its principal axis system. The following spin–orbit perturbation operators are available: (Effective) Pauli operator, (modified) screened nuclear spin–orbit (mSNSO, SNSO), and spin–orbit mean-field (SOMF) ansatz. We recommend the SNSO Hamiltonian in non-relativistic calculations and the mSNSO Hamiltonian in scalar-relativistic X2C calculations.

The general keywords are as follows.

`$pnmr` followed by *hfc-only*; HFC tensor is computed with the FC, SD, and PSOSO terms. Without any additional keywords, the one-electron Pauli spin–orbit operator is applied.

`$pnmr` followed by *hfc-only scalar*; Only the FC and SD terms are computed.

`$epr` can be used to compute all EPR parameters in one run. Then, the `$pnmr` settings are ignored.

`$snso` The effective Pauli or SNSO/mSNSO terms are used for PSOSO. Available options for the additional keyword `$snsopara` are

`snsopara 0` SNSO Hamiltonian [145]

`snsopara 1` mSNSO Hamiltonian [146, 147], only available with X2C

`snsopara 2` Effective nuclear charges of [294], not available with X2C.

`snsopara 3` Effective nuclear charges of [295], not available with X2C

With `snsopara 2` effective nuclear charges for the elements B-F and Al-Cl are available. With `snsopara 3` effective nuclear charges for the elements Li-Ar are available. For H and He, the respective actual nuclear charges are used as the effective charges. If a compound contains elements for which no effective nuclear charges are available and which are not H or He, all elements in the compound are described with their actual nuclear charges. We recommend `snsopara 0` for non-relativistic calculations and `snsopara 1` for X2C calculations on heavier compounds.

`$somf` followed by *nuclear real-a coulomb real-b exchange real-c*

The three parameters *real a–c* are used to scale the nuclear, coulomb and exchange contribution. Using 1.0 for all three parameters gives the standard expression for the SOMF ansatz (see e.g. [296]).

Note that only the keyword `$pnmr` is necessary, all SNSO and SOMF related keywords are optional. Using both SNSO and SOMF is not possible. The PSOSO term supports the current-dependent generalization of τ [125, 281] for mGGAs and LHFs (`$curswitchengage`) and the HFC tensor is available for all functionals. The non-generalized kinetic energy density can be used with `$curswitchdisengage`. However, we strongly recommend the current-dependent formalism. Note that the one-component approach is generally sufficient for the isotropic HFC constant up to the 4d elements. For heavier elements, we recommend the two-component treatment.

Two-component treatment: This requires three independent two-component (2c) calculations with the spin magnetization vector aligned along the Cartesian axes. 2c calculations are started with the keyword `$soghf`. This should be done with converged one-component MOs as an initial guess. The spin magnetization is aligned along the axes using the keywords `$sxeig`, `$syeig`, and `$szeig`. Each spin direction then yields 3 tensor elements. The HFC contributions are calculated with converged spinors in `ridft` or `mpshift`. `ridft` requires the additional keyword `$x2c_hfc` and `mpshift` requires `$pnmr hfc-only` or `$epr`. The latter calculates all EPR parameters. We strongly recommend the DLU-X2C approach in combination with the mSNSO correction, see also Sec. 6.4. All functionals are supported [125, 138, 286]. Note that spin–orbit coupling induces a current density in the 2c ground-state calculations. So, mGGAs and LHFs formally need to be generalized to their current-dependent form already at the SCF level (`$curswitchengage`), i.e. a spin–orbit current density functional theory (CDFT) framework is necessary [42].

The complete work flow is as follows:

1. Delete all previously inserted two-component keywords and the spinor files. Alternatively, set up a new directory from scratch. We need to start from UHF/UKS orbitals to align the spin.

2. Run a UHF/UKS calculation with the X2C (`$rx2c`) or DLU-X2C Hamiltonian (`$rx2c` and `$rlocal`). The finite nucleus model (`$finnuc`) is strongly recommended. Please make sure to use the proper grids for X2C and DFT (gridsize 3a etc.). Do not use gridsize m3 or m4.

3. Use *hfcprep.sh* to prepare your 2c input (see *hfcprep.sh –h* for more information; *hfcprep.sh -msnso* is recommended). The keyword `$nucsel` can be used followed by the number of the nucleus of interest, e.g., `$nucsel 1,3,7`. Alternatively, you can set the element, e.g., `$nucsel "c","h"`.

4. Run the 2c SCF calculations in the directories x, y, and z. `ridft` computes the HFC tensor components and stores the results in the control file. Additionally, you can run an `mpshift` calculation with `$pnmr hfc-only` based on the converged spinors of the 2c `ridft` calculation similar to the one-component work flow. The same holds for `$epr` and `mpshift`.

5. Go to parent directory and call *calchfc.py*, which computes the HFC in its principal axis system based on the rank-2 tensor. This approach loses the sign information of the tensor. To recover it, we consider that the trace is the sum

of the eigenvalues. Alternatively, you may use *calchfc.py –s* which yields the principal axis system based on a symmetrization, for which the sign information is retained. The outputs of the two methods can be compared to find the sign. Note that the approach based on the rank-2 tensor is more accurate than that based on the symmetrization.

Note that the FC, SD, and PSOSO terms are coupled in a 2c framework. Therefore, only the total HFC tensor is printed.

Furthermore, the HFC is sensitive towards the chosen basis sets. We recommend decontracted x2c-type basis sets or the x2c-QZVPall-2c basis set. The 2c extensions of the x2c-type basis sets are strongly recommended. Alternatively, the Dyall basis sets can be used in combination with the decontracted Dunning bases for the light elements [286].

Note that the effective spin is taken from a 1c calculation, so this work flow needs to be adapted manually if the spin multiplicity changes from the 1c to the 2c case. If the spin expectation value of the SCF density for the respective direction has a negative sign, `ridft` and `mpshift` automatically change the sign of the corresponding HFC tensor components.

In case of 2c SCF convergence issues with the spin alignment or non-orthogonal spin directions at the end of the three `ridft` runs, you may re-orientate the molecule so that the Cartesian axes become the principal axes of the 1c EPR tensors, see also Sec. 18.4.

18.2 EPR g-Tensor

One-component treatment: The g-tensor is calculated with the (effective) Pauli operator or the SNSO/mSNSO approach. Here, X2C applies the picture-change correction to the spin–orbit perturbation. By default, gauge-including atomic orbitals (GIAOs) are used throughout and the g-tensor is implemented in the `mpshift` module. All functionals including local hybrids are supported [83, 281]. The kinetic energy density is generalized with the external magnetic field to ensure gauge-origin invariance [44] by default. The generalization using the current density is applied with the keyword `$curswitchengage` [280, 281]. This is strongly recommended.

The general keywords are as follows.

`$pnmr` followed by *g-only*; g-tensor is calculated. The g-tensor is transformed into the principal axis system and the Δg-shift with respect to the g-factor of the free electron is given in ppt (parts per thousand).

`$epr` can be used to compute all EPR parameters in one run. Then, the `$pnmr` settings are ignored.

`$snso` The effective Pauli or SNSO/mSNSO terms are used for PSOSO. Available options for the additional keyword `$snsopara` are

`snsopara 0` SNSO Hamiltonian [145]

`snsopara 1` mSNSO Hamiltonian [146, 147], only available with X2C.

`snsopara 2` Effective nuclear charges of [294], not available with X2C.

`snsopara 3` Effective nuclear charges of [295], not available with X2C.

With `snsopara 2` effective nuclear charges for the elements B-F and Al-Cl are available. With `snsopara 3` effective nuclear charges for the elements Li-Ar are available. For H and He, the respective actual nuclear charges are used as the effective charges. If a compound contains elements for which no effective nuclear charges are available and which are not H or He, all elements in the compound are described with their actual nuclear charges. We recommend `snsopara 0` for non-relativistic calculations and `snsopara 1` for X2C calculations on heavier compounds.

In non-relativistic calculations, the g-tensor may be partitioned into the relativistic mass correction (RMC), the diamagnetic correction (DC), and the orbital Zeeman (OZ) term. The derivation of the RMC term is based on the virial theorem for molecules (`$rmc`) and is only rigorous when considering the complete potential, i.e. one and two-electron contributions. Hence, the Pauli kinetic energy (PKE) term is formally more correct, better suited for heavy elements and is used with the keyword `$pke`. By default, the PKE term is considered. X2C calculations use the PKE term only. For the g-tensor and g-shift of EPR experiments, the two-component treatment is suggested for heavy elements.

The common gauge origin (CGO) approach is applied with `$cgo_epr`. By default, the heaviest atom is chosen as common gauge origin. To manually set the CGO, use `$cgo` *integer*, where the atom number of the coordinate file selects the respective nucleus. `$cgo 0` selects the center of mass as common gauge origin. However, the GIAO-based ansatz is strongly recommended.

Two-component treatment: The g-tensor can be obtained with a common gauge origin (CGO) in the `ridft` and `mpshift` modules or gauge-including atomic orbitals in the `mpshift` module only. The latter is recommended [287]. To use a CGO with `mpshift` set `$cgo_epr`, `ridft` always uses a CGO. The work flow is similar to the HFC tensor. This requires three independent two-component (2c) calculations with the spin magnetization vector aligned along the Cartesian axes. Again, this should be done with converged one-component MOs. The spin magnetization is aligned with the keywords `$sxeig`, `$syeig`, and `$szeig`. Then, each spin direction yields 3 tensor elements. The g-tensor contributions are calculated with converged spinors in `ridft` or `mpshift`. `ridft` requires the keyword `$x2c_gtensor` or `$x2c_gtensor rkb` and `mpshift` needs `$pnmr g-only`. The restricted magnetic balance (RMB) condition is employed by default. Alternatively, the flag `$epr` can be used to compute all EPR parameters in one `mpshift` run per spin orientation. We strongly recommend the DLU-X2C approach in combination with the mSNSO correction, see also Sec. 6.4.

The complete work flow is as follows:

1. Delete all previously inserted two-component keywords and the spinor files.

Alternatively, set up a new directory from scratch. We need to start from UHF/UKS orbitals to align the spin.

2. Run a UHF/UKS calculation with the X2C (`$rx2c`) or DLU-X2C Hamiltonian (`$rx2c` and `$local`). The finite nucleus model (`$finnuc`) is strongly recommended. Please make sure to use the proper grids for X2C and DFT (gridsize 3a etc.). Do not use gridsize m3 or m4.

3. Use *gtensprep.sh* to prepare your 2c input (see *gtensrep.sh –h* for more information; *gtensprep.sh -msnso* is recommended) The HFC and g-tensor can be computed simultaneously with *gtensprep.sh -msnso -hfc*.

4. Run the 2c SCF calculations in the directories x, y, and z. `ridft` computes the g-tensor components using a common gauge origin and stores the results in the control file. Alternatively, run an `mpshift` calculation with `$pnmr g-only` based on the converged spinors. Alternatively, you may use `$epr` and `mpshift`.

5. Go to parent directory and call *calcgtens.py*, which computes the g-tensor in its principal axis system based on the rank-2 tensor. Additionally, the Δg-shift is computed in ppt.

By default, the heaviest atom is chosen as common gauge origin. To manually set the CGO, use `$cgo` *`integer`*, where the atom number of the coordinate file selects the respective nucleus. `$cgo 0` selects the center of mass as common gauge origin. Note that 2c calculations with local hybrid functionals are currently restricted to the CGO approach [138]. The same holds for 2c CDFT approaches.

For systems with multiple metal centers, we strongly recommend the GIAO ansatz in `mpshift`. This only leads to a minor computational overhead.

18.3 Electric Field Gradient

The electric field gradient (EFG) can be used to calculate the nuclear quadrupole interaction (NQI) tensor for analysis purposes [287]. The EFG, $\overleftrightarrow{V}$, is evaluated in atomic units (a.u.) and the NQI tensor in MHz follows as

$$\overleftrightarrow{Q}(\mathrm{MHz}) = \frac{234.9648}{2I(2I-1)} \cdot Q_c(\mathrm{b}) \cdot \overleftrightarrow{V}(\mathrm{a.u.}) \tag{18.1}$$

where Q_c refers to the quadrupole constant and I refers to the nuclear spin. The relativistic picture-change correction and large basis sets are crucial for accurate results.

The EFG is calculated by setting `$efg` in the control file. Additionally, `$pcc` and the usual X2C keywords should be set, i.e. `$rx2c` and `$rlocal`. The spin–orbit two-component formalism is supported with `$soghf`. As the EFG only requires the SCF density matrix, a *proper* run of `dscf` or `ridft` is sufficient. To calculate the EFG of

selected nuclei only, the `$nucsel` keywords can be set. The principal axis system is also printed.

The `mpshift` run with `$epr` or `$efg` evaluates the EFG and calculates the NQI for nuclei with a spin of more than one half. Here, the relativistic picture-change correction [37] is including by default with `$rx2c`.

18.4 Principal Axis System and Euler Angles

The principal axis system (PAS) and the respective Euler angles for the g-tensor can be obtained with `mpshift` and the flag `$epr`. The orientation of the PAS of the HFC and EFG is further calculated relative to the g-tensor frame. Note that `mpshift` with `$epr` calculates the g-tensor, the HFC, the EFG, and the NQI.

For two-component runs, the post-processing script *epreuler.py* is needed. Like the scripts *calcgtens.py* and *calchfc.py*, this script accumulates the g-tensor and HFC results from the three non-collinear calculations. By default, the EFG needs to be stored in 'ridft.efg' with the z-spin orientation. Make sure to use `$pcc` for the *proper* of `ridft`. Thus, *gtensprep.sh -efg* adds the `$pcc` automatically.

Note that the Euler angles are not unique, for instance, when flipping the sign of the β angle, we have to add or subtract π to the α and γ angles to get the same rotation matrix.

18.5 ZFS tensor

The ZFS tensor can be calculated for compounds with a spin $S > 1/2$. It is only implemented within a non- and scalar-relativistic one-component treatment using spin–orbit perturbation theory in the `mpshift` module [293]. It is commonly divided into the spin–dipolar (SD) and spin–orbit (SO) contributions (see Ref. [297] for details on the derivations with GGAs and their hybrids, and [293] for the extension to mGGAs, local hybrids and X2C). The SD contribution consists of four-center-two-electron integrals, which are evaluated in a seminumerical scheme on a grid [124] and are contracted with the spin-excess density matrix. There are two approaches for constructing the spin-excess density matrix. On the one hand, we can use the canonical HF or KS MOs (direct method). On the other hand, unrestricted natural orbitals (UNOs) can be employed to reduce the spin contamination (UNO method) [298]. We generally recommend the UNO method and it is chosen by default both for calculations with the keywords `$epr` and `$pnmr`. The SO contribution consists of the same-spin and spin-flip contributions and requires the calculation of first-order orbital-rotation matrices with respect to a spin–orbit perturbation. This is evaluated with the coupled-perturbed Hartree–Fock (CPHF) or Kohn–Sham (CPKS) equations. Here, the following spin–orbit perturbation operators are available: (Effective) Pauli operator, (modified) screened nuclear spin–orbit (mSNSO, SNSO), and spin–orbit mean-field (SOMF) ansatz. We recommend the SNSO Hamiltonian in

non-relativistic calculations and the mSNSO Hamiltonian in X2C calculations. Note that X2C or DLU-X2C always include the relativistic picture-change correction.

After the calculation, the ZFS tensor is transformed to its principal axis system. The axial ZFS parameter D, the rhombic parameter E, and the rhombicity parameter E/D are calculated via the elements of the diagonalized tensor by

$$\mathrm{D} = D_{zz} - \frac{1}{2}\left(D_{xx} + D_{yy}\right), \tag{18.2}$$

$$\mathrm{E} = \frac{1}{2}\left(D_{xx} - D_{yy}\right), \tag{18.3}$$

$$0 \leq \mathrm{E/D} \leq 1/3. \tag{18.4}$$

Additionally, a traceless version of the tensor is calculated as well.

The general keywords are as follows.

`$pnmr` followed by *zfs*; ZFS tensor is calculated additional to the other properties calculated when doing a pNMR calculation. Note, that the calculaton of the paramagnetic shift does NOT use the ZFS tensor, but the simplified equation for doublets, even for $S > 1/2$! The UNO approach is chosen as a default for the SD contribution.

`$pnmr` followed by *zfs-only*; Only the ZFS tensor is calculated.

`$pnmr` followed by *zfs-dip*; Only the SD contribution to the ZFS tensor is calculated.

`$pnmr` followed by *zfs-soc*; Only the SO contribution to the ZFS tensor is calculated.

`$pnmr` followed by *zfs direct*; Direct approach for the SD term is used. Works analogously with *zfs-only* and *zfs-dip*.

`$epr` can be used to compute all EPR parameters in one run. Then, the `$pnmr` settings are ignored. The UNO approach is used for the SD contribution to the ZFS tensor.

`$sdip-sen` is used to set a gridsize for the seminumerical SD contribution by writing `gridsize a` below this keyword (similar to `$dft`). Recommended is the default -2. Other usable grid sizes are `tiny` and -1 ... 9, where for 9 more specifications are necessary (see Section 23.2.10).

`$sddenstol` followed by an integer *int*. It is used to set a threshold of 10^{-int}. SD integrals over shell pairs are negelected if the corresponding elements of the spin-density matrix fall below this threshold. The default for *int* is 14.

`$snso` The effective Pauli or SNSO/mSNSO terms are used for the SO contribution. Available options for the additional keyword `$snsopara` are

`snsopara 0` SNSO Hamiltonian [145]

`snsopara 1` mSNSO Hamiltonian [146, 147], only available with X2C

`snsopara 2` Effective nuclear charges of [294], not available with X2C.

`snsopara 3` Effective nuclear charges of [295], not available with X2C

With `snsopara 2` effective nuclear charges for the elements B-F and Al-Cl are available. With `snsopara 3` effective nuclear charges for the elements Li-Ar are available. For H and He, the respective actual nuclear charges are used as the effective charges. If a compound contains elements for which no effective nuclear charges are available and which are not H or He, all elements in the compound are described with their actual nuclear charges. We recommend `snsopara 0` for non-relativistic calculations and `snsopara 1` for X2C calculations on heavier compounds.

`$somf` followed by *nuclear real-a coulomb real-b exchange real-c*

The three parameters *real a–c* are used to scale the nuclear, coulomb and exchange contribution. Using 1.0 for all three parameters gives the standard expression for the SOMF ansatz (see e.g. [296]).

Note that using both SNSO and SOMF is not possible. The SO contribution supports the current-dependent generalization of τ [125, 281] for mGGAs and local hybrid functionals (`$curswitchengage`) and the ZFS tensor is available for all functionals up to the class of local hybrids. The non-generalized kinetic energy density can be used with `$curswitchdisengage`. We recommend GGA and current-dependent mGGA based hybrid functionals such as ωB97X-D with a substantial admixture of HF exchange for the simultaneous calculation of all EPR properties, see [293].

When looking at the output and extracting the ZFS parameters D and E, it is important to check, whether E/D is close to 1/3. If this is the case, the sign of D becomes unreliable, as small changes in the diagonal elements of the tensor may lead to a flip of the sign. Additionally, note again that the ZFS tensor is not used for the calculation of the pNMR shift, even when using the keyword `$pnmr zfs`. The pNMR shielding tensor including the ZFS contribution can be computed as outlined in [288] based on the scripts of https://github.com/jautschbach/pnmr-shift.

Chapter 19

Embedding and Solvation Effects

TURBOMOLE provides provides a number of different embedding schemes to model the interaction of the quantum system with environments. They reach from a simple point charge embedding to the continuum solvation model `cosmo` and the atomistic polarizable embedding.

Most of these schemes ignore each other. So they should not be used in combination unless it has been tested that the chosen combination works indeed correctly together.

The embedding schemes differ also in the extent to which they have been implemented in the different programs and for different functionalities.

19.1 Charge and multipole embedding

Point Charge Embedding: The embedding of the quantum system in a electric potential of an (non-periodic) set of charges and multipole moments is driven by the data group `$point_charges`. In the simplest case it has the structure:

```
$point_charges
   <x> <y> <z> <q>
```

where `<x>`, `<y>`, `<z>` are the coordinates and `<q>` the value of the point charge.

The point charge embedding is implemented in the `dscf`, `ridft`, `grad`, `rdgrad`, `escf`, `ricc2`, `ccsdf12`, and `pnoccsd` programs and can be used essentially with every method and for all properties including gradients with respect to nuclear coordinates. Exceptions are (auxiliary) basis set gradients, analytic second derivatives (`aoforce` (but they can be computed semi-numerical with `NumForce`.

For QM/MM applications it is also possible to compute the forces that the quantum system exerts on the point charges. For further details about available options and the input see Sec. 23.2.11.

Point Mutipole Embedding: In addition to point charges one can also use point multipoles up to octupole moments. The input uses generalization of the point charge input:

```
$point_charges mxrank=3
   <x> <y> <z> <q> <qx> <qy> <qz> <qxx> <qyy> <qzz> <qxy> <qxz> <qyz> ...
```

The value of keyword `mxrank` defines the maximum multipole rank (0=charge, 1=dipole, 2=quadrupole, 3=octupole) and the tensor components are given for each multipole site after the coordinates in canonical order. For the components of the octupole moment the canonical order is:

```
xxx yyy zzz xxy xxz xyy yyz xzz yzz xyz
```

The multipole embedding (beyond charges) is only implemented for (excitation) energies. Gradients are not (yet) available and symmetry can not be used.

Gaussian Smeared Charges: Instead of point charges one use Gaussian charge distributions of the form $G(\vec{r}) = N \exp\big(-\alpha(\vec{r}-\vec{r}_0)^2\big)$. The input format is in this case:

```
$point_charges gaussians
   <x> <y> <z> <q> <alpha>
```

The normalization factor N is determined internally such that the total (integrated) charge of the distribution is $q = \int_{R^3} G(\vec{r}) d\tau$.

For Gaussian charge distributions gradients are available, but symmetry can not be used.

19.2 Treatment of Solvation Effects with COSMO

The **Co**nductor-like **S**creening **Mo**del [299] (COSMO) is a continuum solvation model (CSM), where the solute molecule forms a cavity within the dielectric continuum of permittivity ε that represents the solvent. The charge distribution of the solute polarizes the dielectric medium. The response of the medium is described by the generation of screening charges on the cavity surface.

CSMs usually require the solution of the rather complicated boundary conditions for a dielectric in order to obtain the screening charges. COSMO instead uses the much simpler boundary condition of vanishing electrostatic potential for a conductor,

$$\mathbf{\Phi}^{tot} = 0.$$

This represents an electrostatically ideal solvent with $\varepsilon = \infty$. The vector of the total electrostatic potential on the cavity surface segments is determined by the solute potential $\mathbf{\Phi}^{sol}$, which consist of the electronic and the nuclear part, and the vector of the screening charges $\mathbf{q}$,

$$\mathbf{\Phi}^{tot} = \mathbf{\Phi}^{sol} + \mathbf{A}\mathbf{q} = 0.$$

$\mathbf{A}$ is the Coulomb matrix of the screening charge interactions. For a conductor, the boundary condition $\mathbf{\Phi}^{tot} = 0$ defines the screening charges as

$$\mathbf{q} \quad = \quad -\mathbf{A}^{-1}\mathbf{\Phi}^{sol}.$$

To take into account the finite permittivity of real solvents, the screening charges are scaled by a factor.

$$\begin{aligned} f(\varepsilon) &= \frac{\varepsilon - 1}{\varepsilon + \frac{1}{2}} \\ \mathbf{q}^{\star} &= f(\varepsilon)\mathbf{q} \end{aligned}$$

The deviation between the COSMO approximation and the exact solution is rather small. For strong dielectrics like water it is less than 1%, while for non-polar solvents with $\varepsilon \approx 2$ it may reach 10% of the total screening effects. However, for weak dielectrics, screening effects are small and the absolute error therefore typically amounts to less than one kcal/mol. As shown in [300] ions can be described more accurately by using the scaling factor $f(\varepsilon) = \frac{\varepsilon-1}{\varepsilon+0}$. This also leads to results (e.g. by comparing solvation free energies) more or less identical to IEFPCM or SS(V)PE. This is possible since `TURBOMOLE` version 7.1 by adding the keyword `ions` to the `epsilon` option in the `$cosmo` section (see 23.2.13).

The dielectric energy, i.e. the free electrostatic energy gained by the solvation process, is half of the solute-solvent interaction energy.

$$E_{diel} = \frac{1}{2} f(\varepsilon)\mathbf{q}^{\dagger}\mathbf{\Phi}^{sol}$$

The total free energy of the solvated molecule is the sum of the energy of the isolated system calculated with the solvated wave function and the dielectric energy

$$E = E(\Psi^{solv}) + E_{diel}.$$

A COSMO energy calculation starts with the construction of the cavity surface grid. Within the SCF procedure, the screening charges are calculated in every cycle and the potential generated by these charges is included into the Hamiltonian. This ensures a variational optimization of both the molecular orbitals and the screening charges and allows for the evaluation of analytic gradients.

Radii based Cavity Construction: In order to ensure a sufficiently accurate and efficient segmentation of the molecular shaped cavity the COSMO implementation uses a double grid approach and segments of hexagonal, pentagonal, and triangular shape. The cavity construction starts with a union of spheres of radii $R_i + RSOLV$ for all atoms i. In order to avoid problems with symmetric species, the cavity construction uses de-symmetrized coordinates. The coordinates are slightly distorted with a co-sinus function of amplitude AMPRAN and a phase shift PHSRAN. Initially a basis grid with NPPA segments per atom is projected onto atomic spheres of radii $R_i + RSOLV$. In order to avoid the generation of points in the problematic intersections, all remaining points, which are not in the interior of another sphere, are projected downwards onto the radius R_i. In the next step a segment grid of NSPH segments per H atom and NSPA segments for the other atoms is projected onto the surface defined by R_i. The basis grid points are associated to the nearest segment grid centers and the segment coordinates are re-defined as the center of area of their associated basis grid points, while the segment area is the sum of the basis grid areas. Segments without basis grid points are discarded. In order to ensure nearest neighbor association for the new centers, this procedure is repeated once. At the end of the cavity construction the intersection seams of the spheres are paved with individual segments, which do not hold associated basis grid points.

Density based Cavity Construction: Instead of using atom specific radii the cavity can be defined by the electron density. In such an isodensity cavity construction one can use the same density value for all atoms types or the so-called scaled isodensity values. In the later approach different densities are used for the different atom types. The algorithm implemented in TURBOMOLE uses a marching tetrahedron algorithm for the density based cavity construction. In order to assure a smooth density change in the intersection seams of atoms with different isodensity specification, this areas are smoothened by a radii based procedure.

Radii based Isosurface Cavity: A cavity construction algorithm based on the triangulation of an iso-surface is available as an alternative to the radii or density based construction. It overcomes deficiencies which have become apparent for the original COSMO standard cavity, especially in concave regions of the molecular shaped cavity. The new construction, called FINE Cavity, is described in details in [301].

To enable the new radii based isosurface cavity the keyword `$cosmo_isorad` has to be added to the control file.

A-Matrix Setup: The **A** matrix elements are calculated as the sum of the contributions of the associated basis grid points of the segments k and l if their distance is below a certain threshold, the centers of the segments are used otherwise. For all segments that do not have associated basis grid points, i.e. intersection seam segments, the segment centers are used. The diagonal elements A_{kk} that represent the self-energy of the segment are calculated via the basis grid points contributions, or by using the segment area $A_{kk} \approx 3.8\sqrt{a_k}$, if no associated basis grid points exist.

Outlying charge correction: The part of the electron density reaching outside the cavity causes an inconsistency that can be compensated by the "outlying charge correction". This correction will be performed at the end of a converged SCF or an iterative MP2 calculation and uses an outer surface for the estimation of the energy and charge correction [302]. The outer surface is constructed by an outward projection of the spherical part of the surface onto the radius $R_i + ROUTF * RSOLV$. It is recommended to use the corrected values.

Numerical Frequency Calculation: The calculation of harmonic frequencies raises the problem of non-equilibrium solvation in the COSMO framework, because the molecular vibrations are on a time scale that do not allow a re-orientation of the solvent molecules. Therefore, the total response of the continuum is split into a fast contribution, described by the electronic polarization, and a slow term related to the orientational relaxation. As can be shown [303] the dielectric energy for the disturbed state can be written as

$$E^d_{diel} = \frac{1}{2} f(\varepsilon)\mathbf{q}(\mathbf{P}^0)\mathbf{\Phi}(\mathbf{P}^0) + \frac{1}{2} f(n^2)\mathbf{q}(\mathbf{P}^\Delta)\mathbf{\Phi}(\mathbf{P}^\Delta) + f(\varepsilon)\mathbf{q}(\mathbf{P}^0)\mathbf{\Phi}(\mathbf{P}^\Delta),$$

where $\mathbf{P}^\Delta$ denotes the density difference between the distorted state and the initial state with density $\mathbf{P}^0$. The interaction is composed of three contributions: the initial state dielectric energy, the interaction of the potential difference with the initial state charges, and the electronic screening energy that results from the density difference. The energy expression can be used to derive the correspondent gradients, which can be applied in a numerical frequency calculation. Because the COSMO cavity changes for every distorted geometry the initial state potential has to be mapped onto the new cavity in every step. The mapped potential of a segment of the new cavity is calculated from the distance-weighted potentials of all segments of the old cavity that fulfill a certain distance criterion. The mapped initial state screening charges are re-calculated from the new potential.

19.2.1 Iterative COSMO-MP2

For ab initio MP2 calculations within the CSM framework three alternatives can be found in the literature [304]. The first approach, often referred to as PTE, performs a normal MP2 energy calculation on the solvated HF wave function. The response of the solvent, also called reaction field, is still on the HF level. It is the only of the three approaches that is formally consistent in the sense of second-order perturbation

theory [305, 306]. In the so-called PTD approach the vacuum MP2 density is used to calculate the reaction field. The third approach, often called PTED, is iterative so that the reaction field reflects the density of the first-order wave function. In contrast to the PTE approach the reaction field, i.e. the screening charges, change during the iterations until self consistency is reached. Gradients are available on the formally consistent PTE level only [307].

19.2.2 COSMO-CC2 for ground-state calculations

The ground state energy and gradient is available for COSMO-CC2 within the post-SCF [308] reaction-field scheme with a CCS-like approximation for the density that used to evaluate the reaction field. In the post-SCF scheme, the ground-state reaction field is first determined self-consistently at the COSMO-HF level. In the subsequent correlation treatment, the correlation effect to the reaction-field is calculated from the correlation contribution to the unrelaxed density and included in the equations for the ground-state wavefunction parameters. In order to set the input file for the calculation of ground-state gradient, energy and relaxed properties the following data groups must be included in the `control` file.

```
$reaction_field
  post-SCF
  ccs-like
$cosmo
  epsilon=   50.000
  rsolv= 1.30
$cosmo_atoms
...
```

19.2.3 Vertical excitations and Polarizabilities for TDDFT, TDA and RPA:

The `escf` program accounts for the COSMO contribution to the excitation energies and polarizabilities. The COSMO settings have to be defined for the underlying COSMO `dscf` or `ridft` calculation. In case of the excitation energies the solvent response will be divided into the so-called slow and fast term [303,309]. The screening function of the fast term depends on the refractive index of the solvent which can be defined in the input. If only the COSMO influence on the ground state should be taken into account we recommend to perform a normal COSMO calculation (`dscf` or `ridft`) and to switch off COSMO (i.e. deactivate `$cosmo`) before the `escf` calculation.

19.2.4 The Direct COSMO-RS method (DCOSMO-RS):

In order to go beyond the pure electrostatic model a self consistent implementation of the COSMO-RS model the so-called "Direct COSMO-RS" (DCOSMO-RS) [310] has been implemented in `ridft` and `dscf`.

COSMO-RS (COSMO for Real Solvents) [311, 312] is a predictive method for the calculation of thermodynamic properties of fluids that uses a statistical thermodynamics approach based on the results of COSMO SCF calculations for molecules embedded in an electric conductor, i.e. using $f(\varepsilon) = 1$. The liquid can be imagined as a dense packing of molecules in the perfect conductor (the reference state). For the statistical thermodynamic procedure this system is broken down to an ensemble of pair wise interacting surface segments. The interactions can be expressed in terms of surface descriptors. e.g. the screening charge per segment area ($\sigma_t = q_t/a_t$). Using the information about the surface polarity σ and the interaction energy functional, one can obtain the so-called σ-potential ($\mu_S(\sigma;T)$). This function gives a measure for the affinity of the system S to a surface of polarity σ. The system S might be a mixture or a pure solvent at a given temperature T. Because the parabolic part of the potential can be described well by the COSMO model, we subtract this portion form the COSMO-RS potential:

$$\tilde{\mu}_S(\sigma;T) = \mu_S(\sigma;T) - (1 - f(\varepsilon))c_0\sigma^2.$$

The parameter c_0 can be obtained from the curvature of a COSMO-RS σ-potential of a nonpolar substance, e.g. hexane. Thus, the remaining part of the chemical potential of a compound i with mole fraction x_i in the mixture Si can be expressed as:

$$\mu^i \cong \sum_{t=1}^{m} f_{pol} a_t \tilde{\mu}_S(\sigma;T) + \mu^i_{C,S} + kT\ln(x_i).$$

where the combinatorial term $\mu^i_{C,S}$ accounts for effects due to the size and shape differences of the molecules in the mixture and a_t denotes the area of segment t. The $kT\ln(x_i)$ can be skipped for infinite dilution. The factor f_{pol} has been introduced to account for the missing solute-solvent back polarization. The default value is one in the current implementation. The free energy gained by the solvation process in the DCOSMO-RS framework is the sum of the dielectric energy of the COSMO model and the chemical potential described above:

$$E_{diel,RS} = \frac{1}{2} f(\varepsilon)\mathbf{q}^\dagger \mathbf{\Phi}^{sol} + \mu^i = E_{diel} + \mu^i$$

From the above expression the solvent operator $\hat{V}^{RS}$ can be derived by functional derivative with respect to the electron density:

$$\hat{V}^{RS} = -\sum_{t=1}^{m} \frac{f(\varepsilon)q_t + q_t^{\Delta RS}}{|\mathbf{r}_t - \mathbf{r}|} = \hat{V}^{cos} - \sum_{t=1}^{m} \frac{q_t^{\Delta RS}}{|\mathbf{r}_t - \mathbf{r}|}.$$

Thus, the solvation influence of the COSMO-RS model can be viewed as a correction of the COSMO screening charges q_t. The additional charges denoted as $q_t^{\Delta RS}$ can

be obtained from $\mathbf{q}^{\Delta RS} = -\mathbf{A}^{-1}\mathbf{\Phi}^{\Delta RS}$, where the potential $\mathbf{\Phi}^{\Delta RS}$ arises from the chemical potential of the solute in the solvent:

$$\phi_t^{\Delta RS} = a_t \left(\frac{\delta \tilde{\mu}_S}{\delta q} \right)_{q=q_t}.$$

In order to get a simple and differentiable representation of the COSMO-RS σ-potential $\mu_S(\sigma;T)$, we use equally spaced cubic splines. An approximate gradient of the method has been implemented. DCOSMO-RS can be used in SCF energy and gradient calculations (geometry optimizations) with `dscf`, `ridft`, `grad`, and `rdgrad`. Please regard the restriction of the DCOSMO-RS energy explained in the keyword section 23.2.13. Because the DCOSMO-RS contribution can be considered as a slow term contribution in vertical excitations it does not have to be taken into account in response calculations. For the calculation of vertical excitation energies it is recommended to use the mos of a DCOSMO-RS calculation in a COSMO response calculation (see above).

19.2.5 Solvation effects on excited states using COSMO in ricc2:

The COSMO approach has been recently implemented into the ricc2 module of TURBOMOLE for excitation energies with CC2 and ADC(2). The ADC(2) method has been implmented in combination with the iterative PTED and the more economical post-SCF reaction field schemes. CC2 is currently only available in combination with the post-SCF reaction field scheme.

Iterative COSMO-ADC(2) within the PTED scheme In the framework of the old implementation of COSMO-ADC(2) within the PTED reaction field (RF) scheme it is possible to equilirate the solvent charges for the ground state at MP2 or any excited state. Using the methods CCS/CIS or ADC(2) the implementation is complete, for CC2 or higher methods, however it still has to be proven if there are terms missing.

The PTED implementation of COSMO-ADC(2) contains contributions to the off-diagonal elements of the one-electron density. Furthermore the energy contributions for non-equilibrated states can be calculated. Non-equilibrated means in this sense, that the slow part of the solvent charges (described by $f(\varepsilon)$) are still equilibrated with a given initial state, while the fast electronic part of the solvent charges (described by $f(n^2)$) are in equilibrium with the target state. To handle this one has to do a macro iteration, like in MP2. This macro iteration can be managed with the script 'cc2cosmo' which is the same as 'mp2cosmo' but using ricc2 instead of rimp2 or mpgrad. To set up the basic settings one can reuse the cosmoprep module, note that the refractive index must be specified 'refind' when observing excited states. To specify the state to which the solvent charges should be equilibrated one inserts the keyword 'cosmorel state=(x)', where the ground state (x) is used normally but can be replaced by any requested excited state. Make sure to request relaxed properties for any desired state, otherwise the COSMO macro iteration will not work in ricc2.

The off-diagonal contributions mentioned above can be switched off by setting the keyword 'nofast' in $cosmo. A typical input might look like:

```
$ricc2
  adc(2)
$excitations
  irrep=a'  multiplicity=  1  nexc=  1  npre=  1  nstart=  1
  irrep=a"  multiplicity=  1  nexc=  1  npre=  1  nstart=  1
  exprop relaxed states=all
$response
  fop relaxed
$cosmo
  epsilon=   50.000
  rsolv= 1.30
  refind=   3.0000
  cosmorel state=(a" 1)
# nofast
$cosmo_correlated
$cosmo_atoms
...
```

This would deliver an excited state calculation for the lowest singlet A' and A" excitations using the ADC(2) method. The solvent charges are equilibrated to state 1^1A" and the non-equilibrium energy contributions for the MP2 ground state and the 1^1A' excited state are calculated furthermore. All contributions to the one-electron density are included since the proper keyword is commented out. Note: when doing solvent relaxations with the CCS/CIS model, no request of relaxed ground–state properties are needed, since the relaxed ground state is identical to the HF ground state.

COSMO-ADC(2) within the post-SCF scheme: The new implementation of COSMO-ADC(2) within the framework of post-SCF reaction–field scheme enables the calculations of vertical excitations energies,transition moments (TM),excited state first-order properties, and analytic gradients. [313] Unlike the PTED scheme, the COMSO-ADC(2) method with post-SCF does not include the macro iterations to calculate vertical excitation energies, meaning that after solving the ground–state reaction field self-consistently and determining the polarized molecular orbitals at the COSMO-HF level the program computes the correlation contribution to the reaction field without coupling back to the ground–state HF.

COSMO-CC2 within the post-SCF scheme: The implementation of COSMO-CC2 within the post-SCF reaction field enables the calculations of vertical excitation

energies, transition moments and Faraday $\mathcal{B}$ terms for the simulation of UV-Vis and magnetic circular dichroism (MCD) absorption spectrum. [314]

A typical input for COSMO-CC(2) and COSMO-ADC(2) any excited-state calculations with the post-SCF scheme include following data groups in addition to the typical COSMO data groups:

```
$reaction_field
  post-SCF
  ccs-like
$cosmo
  epsilon=   50.000
  rsolv= 1.30
  refind=   3.0000
$cosmo_atoms
...
```

The input for `$ricc2`, `$excitations`, `$laplace`, and `$response` data groups is the same as without COSMO.

The Following table shows the avaliability of COSMO-ADC(2) method with the PTED (old implementation) and post-SCF (new implementation), and COSMO-CC2 with the post-SCF scheme for different calculations.

	COSMO-ADC(2) PTED	**COSMO-ADC(2) post-SCF**	**COSMO-CC2 post-SCF**
Excitation Energy	✓	✓	✓
Transition Moments	✓	✓	✓
Excited-state Prop.	✓	✓	-
Excited-state Gradient	-	✓	-
Magnetic Circular Dichroism $\mathcal{B}$	-	-	✓
Two-photon Absorption	-	-	✓

The combination of COSMO-CC2 and COSMO-ADC(2) with SCS and SOC are available in the `ricc2` module. Furthermore, the calculation of all the features available at COSMO-CC2 and COSMO-ADC(2) with post-SCF scheme are possible with open-shell systems and also with the SMP (OpenMP) and MPI parallelizations.

A short summary of the COSMO input is given at the beginning of the ricc2 output as well as a summary of the energy contributions is given at the end of the ricc2 output.

19.3 Frozen Density Embedding calculations

19.3.1 Background Theory

In the subsystem formulation of the density-functional theory a large system is decomposed into several constituting fragments that are treated individually. This approach offers the advantage of focusing the attention and computational cost on a limited portion of the whole system while including all the remaining environmental effects through an *effective embedding potential*. Here we refer in particular to the (fully-variational) Frozen Density Embedding (FDE) [315] with the Kohn-Sham Constrained Electron Density (KSCED) equations [316, 317].

In the FDE/KSCED method the embedding potential required by an embedded subsystem with density ρ_A to account for the presence of another (frozen) subsystem with density ρ_B is:

$$v_{\text{emb}}(\mathbf{r}) = v^B_{\text{ext}}(\mathbf{r}) + v_J[\rho_B](\mathbf{r}) + \frac{\delta T_s^{\text{nadd}}[\rho_A;\rho_B]}{\delta\rho_A(\mathbf{r})} + \frac{\delta E_{xc}^{\text{nadd}}[\rho_A;\rho_B]}{\delta\rho_A(\mathbf{r})} \tag{19.1}$$

where $v^B_{\text{ext}}(\mathbf{r})$ and $v_J[\rho_B](\mathbf{r})$ are the electrostatic potentials generated by the nuclei and electron density of the subsystem B, respectively, and

$$T_s^{\text{nadd}}[\rho_A;\rho_B] = T_s[\rho_A+\rho_B] - T_s[\rho_A] - T_s[\rho_B]\,, \tag{19.2}$$

$$E_{xc}^{\text{nadd}}[\rho_A;\rho_B] = E_{xc}[\rho_A+\rho_B] - E_{xc}[\rho_A] - E_{xc}[\rho_B] \tag{19.3}$$

are the non-additive non-interacting kinetic energy and exchange-correlation energy functionals, respectively. In the expressions above $T_s[\rho]$ is the (unknown) non-interacting kinetic energy density functional and $E_{xc}[\rho]$ is the exchange-correlation energy functional. Note that, while the first two terms in Eq. (19.1) refer to classical electrostatics (and could be described by e.g. external point-charges), the last two terms are related to quantum-mechanical effects.

Using freeze-and-thaw [318] cycles, the role of the frozen and the embedded subsystem is iteratively exchanged, till convergence. If expressions (19.2) and (19.3) are computed exactly, then the density $\rho_A + \rho_B$ will coincide with the exact density of the total system.

Because the FDE/KSCED was originally developed in the Kohn-Sham framework, using standard GGA approximations for $E_{xc}[\rho]$, the non-additive exchange-correlation potential ($\delta E_{xc}^{\text{nadd}}/\delta\rho_A(\mathbf{r})$) can be computed exactly as a functional of the density, leaving the expression of the non-additive kinetic energy term as the only approximation (with respect to the corresponding GGA calculation of the total system), because the exact explicit density dependence of T_s from the density is not known. Using GGA approximations for the kinetic energy functional ($T_s \approx \tilde{T}_s^{GGA}$) we have:

$$T_s^{\text{nadd}}[\rho_A;\rho_B] \approx \tilde{T}_s^{GGA}[\rho_A+\rho_B] - \tilde{T}_s^{GGA}[\rho_A] - \tilde{T}_s^{GGA}[\rho_B] \tag{19.4}$$

and

$$\frac{\delta T^{\text{nadd}}[\rho_A;\rho_B]}{\delta\rho_A(\mathbf{r})} \approx \tilde{v}_T^{\text{GGA}}[\rho_A+\rho_B](\mathbf{r}) - \tilde{v}_T^{\text{GGA}}[\rho_A](\mathbf{r}). \tag{19.5}$$

where $\tilde{v}_T^{\mathrm{GGA}}(\mathbf{r}) = \delta\tilde{T}_s^{\mathrm{GGA}}/\delta\rho(\mathbf{r})$.

The FDE total energy of the total system is:

$$\begin{aligned} E^{FDE}[\tilde{\rho}_A, \tilde{\rho}_B] &= T_s[\tilde{\rho}_A] + T_s[\tilde{\rho}_B] + T_s^{\mathrm{nadd}}[\tilde{\rho}_A; \tilde{\rho}_B] \\ &+ V_{ext}[A+B] + J[\tilde{\rho}_A + \tilde{\rho}_B] + E_{xc}[\tilde{\rho}_A + \tilde{\rho}_B]\,. \end{aligned} \tag{19.6}$$

Note that this energy differs from the KS total energy of the total system due to the approximation in Eq. (19.4) as well as the approximated kinetic potential (see Eq. 19.5) which lead to approximated embedded densities ($\tilde{\rho}_A \approx \rho_A$ and $\tilde{\rho}_B \approx \rho_B$). With the current state-of-the-art GGA kinetic approximations, the error in the binding energy for weakly interacting systems is close to chemical accuracy.

Using the Generalized Kohn-Sham (GKS) theory, also hybrid exchange-correlation functionals can be used in embedding calculations. To obtain a practical computational method, the obtained embedding potential must be approximated by a local expression as shown in Ref. [319]. This corresponds to performing for each subsystem hybrid calculations including the interaction with other subsystems through an embedding potential derived at a semilocal level of theory. When orbital dependent exchange-correlation functionals (e.g. hybrid functional and LHF) are considered within the FDE method, the embedding potential includes a non-additive exchange-correlation term of the form

$$E_{xc}^{\mathrm{nadd}}[\rho_A; \rho_B] = E_{xc}[\Phi^{A+B}[\rho_A + \rho_B]] - E_{xc}[\Phi^{A}[\rho_A]] - E_{xc}[\Phi^{B}[\rho_B]] \tag{19.7}$$

where $\Phi^{A+B}[\rho_A + \rho_B]$ denotes the Slater determinant which yields the total density $\rho_A + \rho_B$. Since such a determinant is not easily available, the non-additive exchange-correlation contribution cannot be determined directly and the non-additive exchange-correlation term can be approximated as [320]

$$E_{xc}^{\mathrm{nadd}}[\rho_A; \rho_B] \approx E_{xc}^{\mathrm{GGA}}[\rho_A + \rho_B] - E_{xc}^{\mathrm{GGA}}[\rho_A] - E_{xc}^{\mathrm{GGA}}[\rho_B]\,. \tag{19.8}$$

The approximation of the kinetic potential within the embedding potential can lead to electron spill-out problems, especially for excitation energy calculations using diffuse basis functions. Here the energy of virtual orbitals is unphysically lowered, and therefore also the corresponding excitation energy, because of a poor description of the Pauli repulsion through the embedding potential. In Ref. [321] all-electron pseudopotentials were introduced at the environment atoms to model the missing Pauli repulsion contributions and additional to the embedding potential the potentials from the all-electron pseudopotentials are added to the vacuum Fock operator(F^{vac}):

$$F_{\alpha\beta}^{FDE(ECP)} = F_{\alpha\beta}^{vac} + \langle\alpha|\nu_{emb}(\mathbf{r})|\beta\rangle + \nu_{ECP}\ . \tag{19.9}$$

Here the ECP potential ν_{ECP} is nonlocal as it employs projection operators to act on specific parts of the one-electron Hilbert space.

19.3.2 Frozen Density Embedding calculations using the FDE script

The shell script `FDE` controls and executes automatically FDE calculations. The script `FDE` prepares the input files (running `define`/`fdetools`), runs the calculations

and combines the results (running `fdetools`). Because the FDE equations are coupled sets of one-electron equations (one for each subsystem), full relaxation of the electron densities of both subsystems is obtained by using a freeze-and-thaw [318] procedure until convergence.

The converged FDE calculations are stored in the subdirectories `STEPN/SUBSYSTEM_AAA`, `STEPN/SUBSYSTEM_AAB` and so on, where N is the number of the FDE iterations. The subdirectories `ISOLATED_SUBSYSTEM_AAA`, `ISOLATED_SUBSYSTEM_AAB` and so on contain instead the calculations of the isolated subsystems (see also Section 19.3.3).

Currently two FDE implementations are available, which differ in the way the embedding potential is computed. One is the original implementation by Laricchia et. al. [319], which needs supermolecular steps in order to compute the embedding potential from matrix elements of the Fock operators. The other one was added by Treß et. al. [321] and it computes the embedding potential on a supermolecular integration grid and therefore avoids any supermolecular steps to compute the embedding potential. This implementation is the current default since it offers applicability for much bigger systems.

Current functionalities and limitations of the default `FDE` implementation are:

- only C_1 point group;
- only for closed-shell systems that consist of closed-shell subsystems
- only total and binding energy calculations (no gradients);
- LDA/GGA kinetic energy functionals (for weakly interacting systems);
- full or pure electrostatic embedding;
- LDA/GGA, hybrid or orbital-dependent exchange-correlation potentials;
- multilevel FDE calculation;
- FDE calculation with frozen subsystems.
- FDE(ECP) method
- serial, OMP and MPI
- works with dscf,ridft,ricc2 and pnoccsd

In order to perform a FDE calculation, the files `coord` and `control` for the total system are necessary to take information on atomic coordinates and basis sets. The input file for the total system can be generated, as usual, with `define` but no calculation on the total system is required. `$denconv 1.d-7` option should be defined in file `control` in order to better converge the embedded densities and better describe the dipole moment. The subsystems have to be defined with the `$frag` keyword, which can either be done by hand or with `define` in case of a HF dimer it may look like this:

```
$frag
   fragment 1 atoms 1-2
   fragment 2 atoms 3-4
```

In addition to the keywords above also the size of the grid and the exchange-correlation functional for the calculation of the embedding potential have to be defined within the keyword `$fde`. It uses the same syntax as the dft keyword and can also be defined with `define`.

After these keywords are defined a FDE calculation can be started by simply invoking the command

```
FDE
```

and an iterative resolution of the KSCED equations with revAPBEk [322,323] as approximation of the non-additive kinetic potential (see Eq. 19.5) in the *monomolecular* basis is performed.

19.3.3 Options

Options for FDE calculations can either be specified as command lines or within the `$fde` keyword. Options specified with the command line can also be specified in file `fde.input`, which is read by the FDE script. If `fde.input` is not present it is created by the FDE script. Command lines options overwrites options found in the `fde.input` file.

Kinetic-energy functionals

In order to use different GGA approximations of the non-additive kinetic potential, the flag `-k` *string* can be used. Here *string* is the acronym used to identify a given GGA kinetic energy approximation, that can be selected among the following functionals:

- *string*=`revapbek`: generalized gradient approximation with a PBE-like enhancement factor, obtained using the asymptotic expansions of the semiclassical neutral atom as reference [322,323] (revAPBEk). This is the default choice;
- *string*=`lc94`: Perdew-Wang (PW91) exchange functional reparametrized for kinetic energy by Lembarki and Chermette [324] (LC94);
- *string*=`t-f`: gradient expansion truncated at the zeroth order (GEA0), corresponding to the Thomas-Fermi functional.

For example, the command

```
FDE -k lc94
```

approximates the non-additive kinetic contribution to the embedding potential through the functional derivative of LC94 kinetic energy functional.

A pure electrostatic embedding can be also performed with `FDE` script, where the embedding potential required by a subsystem A to account for the presence of the B one will be merely:

$$v_{\mathrm{emb}}(\mathbf{r}) = v^{B}_{\mathrm{ext}}(\mathbf{r}) + v_J[\rho_B](\mathbf{r}) \tag{19.10}$$

with $v^{B}_{\mathrm{ext}}(\mathbf{r})$ and $v_J[\rho_B](\mathbf{r})$ the electrostatic potentials generated respectively by the nuclei and electron density of the subsystem B. To perform an electrostatic embedding calculation use

```
FDE  -k electro   .
```

The electrostatic embedding is implemented only for testing purpose. It resembles an electrostatic embedding with external point-charges and/or point-dipoles, but it is "exact" as it is based on the whole densities (i.e. it considers all multipole moments of the density and the polarizabilities at all orders).

Equivalent command: `--kin` *string*

`fde.input` option: `kin=` *string*

The kinetic density functional can also be defined within the `$fde` keyword, for example the LC94 kinetic energy functional:

```
$fde
 gridsize 3
 functional pbe
 kin lc94
```

The `$fde` keyword uses the same abbreviations for the kinetic energy functional as described above for the command line option. In addition to these kinetic functionals also kinetic functionals within the libxc or xcfun library can be specified together with the exchange-correlation functional:

```
$fde
 gridsize 3
 functional libxc 101             #exchange pbe
 functional libxc add 2 130 521   #correlation pbe + LC94
```

FDE charged subsystems

`FDE` can perform calculations for charged closed-shell systems whose charge is localized on one or more subsystems. The charge of a subsystem is defined within the `$frag` keywords. For a sodium cation (atom 1) and water (atoms 2-4) system the `$frag` keyword looks like this:

```
$frag
   fragment 1 charge 1 atoms 1
   fragment 2 charge 0 atoms 2-4
```

FDE(ECP) method

In order to prevent electron spill-out all-electron ECPs can be added to the embedding potential. This can be done by defining the ECPs used for the embedding with the keyword `$ecpemb` and adding *fdeecp* to the keyword `$fde`. Both can be done with `define` in the fde sub menu.

Post Hartree-Fock methods

Post Hartree Fock methods can be used in three different ways for all of them the keyword for the post Hartree Fock (HF) method has to be added first. The first option to use post HF methods is the perturbation to the energy (PE) approach, where contributions in the post HF method only result from the embedded Orbitals obtained from the underlying FDE-HF calculation. This approach can be applied by adding *fdehf* and *kernel 0* to the `$fde` keyword:

```
$fde
  gridsize 3
  functional pbe
  kin lc94
  fdehf
  kernel 0
```

The second option to use post HF methods is the post SCF reaction field method. Here correlation contributions are added to the reaction field after the FaT procedure is converged. This approach can be applied by adding *fdehf* and *kernel 1* to the `$fde` keyword:

```
$fde
  gridsize 3
  functional libxc 101             #exchange pbe
  functional libxc add 2 130 521   #correlation pbe + LC94
  fdehf
  kernel 1
```

As soon as the kernel contributions are added (*kernel 1*) within the post HF methods the kinetic energy functional has to be defined within the libxc or xcfun library. The third option is the perturbation to the energy and density (PTED) approach. Here

the post HF method is included within the FaT cycles and the density from the post HF calculation is used to compute the embedding potential. This approach can be applied by only adding *kernel 1* to the `$fde` keyword:

```
$fde
  gridsize 3
  functional libxc 101             #exchange pbe
  functional libxc add 2 130 521   #correlation pbe + LC94
  kernel 1
```

FDE with frozen environment

`FDE` can perform embedding calculations where the environment of a subsystem is taken *frozen*, i.e. without *scf* calculation on it using an embedding potential. Therefore only one step will be performed if the flag `--frozenenv` will be used

```
FDE --frozenenv 1
```

Here the environment of subsystem 1 is taken *frozen* and the frozen embedding calculation is stored in the subdirectory `STEP1/SUBSYSTEM_AAA`.

Multilevel ansatz within FDE

A multilevel ansatz is defined such that all keywords within the control file of the supermolecular system will also be used for each subsystem and specific keywords for specific subsystems have to be defined in a multilevel reference file. An information about the file has to be added to the `$fde` keyword.

```
$fde
  gridsize 3
  functional pbe
  multilevel file=multi.inp
```

Such a multilevel reference file (here: multi.inp) has to start with the keyword `$multilevel` and end with `$end`. Between those two keywords lines defining the fragments followed by their specific keywords have to be added. For a system containing of three subsystems where subsystem one and two shall be calculated with the DFT method and subsystem three with the CC2 model together with the first two excitation energies the multilevel reference file would look like this:

```
$multilevel
   fragment 1-2
      $dft
```

```
         gridsize 3
         functional pbe
   fragment 3
      $ricc2
        cc2
      $excitations
        irrep=a nexc=2 npre=4 nstart=6
$end
```

It is suggested to define all necessary basis sets already in the control file of the supermolecular system to ensure a correct execution of the calculation.

Parallel calculations

If `PARA_ARCH=SMP` and OMP calculation will be performed. The flag `-nth` *nthreads* can be used to specify the number of threads. For example, with the following command

```
FDE -nth 4
```

will use 4 threads.

Equivalent command: `--nthreads` *integer*

`fde.input` option: `nthreads=` *integer*

Restarting

The script `FDE` checks in the current directory for previous FDE calculations. If these are present, then the FDE calculation will be restarted from the last iteration found. The directories `ISOLATED_SUBSYSTEM_AAA`, `ISOLATED_SUBSYSTEM_AAB` and so on will be overwritten by the converged calculations from the previous run. The energy and the orbital from the isolated systems are saved in the current directory in the files: `isolated_energy.ks`, `mos_AAA.ks`, `mos_AAB.ks` and so on.

Note that a restart is possible only if the same subsystem definition is used. Other flags, e.g. kinetic and xc-functionals and convergence parameters, can be instead modified.

As all the options are saved in the `fde.input` file, to restart a FDE calculation the *FDE script can be invoked without any parameters.*

To force a calculation from scratch use:

```
FDE --scratch
```

19.3.4 Compatibility Mode

The original implementation of Laricchia et. al. can be used by either adding *comp* to the `$fde` keyword or by adding *–comp* to the command line when invoking the FDE script. This implementation has some additional limitations and features:

- serial and OMP `dscf` runs (no MPI);
- monomolecular and supermolecular basis set approach;
- energy-decomposition;
- FDE(ECP) method is not available

Monomolecular and supermolecular basis set approach

The ρ_A and ρ_B densities can be expanded using the supermolecular or monomolecular basis set. In a supermolecular basis set expansion the basis functions $\{\chi\}$ of both subsystems are employed to expand the subsystem electron densities. In a monomolecular basis set expansion, instead, only basis functions $\{\chi^\ell\}$ centered on the atoms in the ℓ-th subsystem are used to expand the corresponding density.

Both monomolecular and supermolecular basis set expansion of the electron densities are implemented in `FDE`: with the flag `-m` a monomolecular expansion is performed, while for a supermolecular one `-s` is used. In the absence of both flags a monomolecular expansion is performed by default.

For an accurate calculation of binding-energies of weakly interacting molecular systems a supermolecular basis set is required (to avoid the basis-set superposition error). Otherwise a very large monomolecular basis set is necessary.

NOTE: The FDE script supports only basis-set in the TURBOMOLE library.

Equivalent command: `--mono` or `--super`

`fde.input` option: `method=mono` or `method=super`

Convergence of the freeze-and-thaw cycles

The script `FDE` runs a self-consistent calculation when a convergence criterion is fulfilled. The convergence criterion is the change in the total dipole moment. This is a tight convergence criterion, as the dipole moment is highly sensitive to small changes in electron density. The convergence parameter ε^j for the j-th step in the freeze-and-thaw procedure is computed by means the following expression

$$\varepsilon^j = \frac{|\Delta\mu_A^j| + |\Delta\mu_B^j|}{2} \tag{19.11}$$

where

$$|\Delta\mu_i^j| \quad = \quad |\mu_i^j| - |\mu_i^{j-1}| \qquad i = A, B$$

is the difference between the dipole moments of two consecutive steps for the i-th subsystem. Eq. (19.11) allows to consider changes in both subsystems or one of them because of the relaxation of their electron densities. By default, `FDE` stops when $\varepsilon^j \leq 0.005$ a.u.. The default value for the convergence criteria can be changed using the flag `--epsilon=` *real* where *real* is a decimal number.

The maximum number of freeze-and-thaw cycles can be specified by `--max-iter=` *integer*, and the default value is 20.

In order to make easy the convergence of the iterative solution of the KSCED coupled equations, a damping factor η must be used for the matrix elements of the embedding potential $\left(v_{\text{emb}}\right)_{ij}$ as perturbation to a given subsystem

$$ {}_d\left(v_{\text{emb}}\right)^j_{ik} = (1-\eta)\left(v_{\text{emb}}\right)^j_{ik} + \eta\left(v_{\text{emb}}\right)^{j-1}_{ik} \tag{19.12} $$

for the j-th iteration. Here ${}_d\left(v_{\text{emb}}\right)^j_{ik}$ is the matrix element effectively used in the j-th iteration after the damping. In `FDE` the starting value of η can be changed using `--start-damp=` *real* (default value is 0.45) where *real* is a decimal number. The damping parameter can also dynamically change at each iterative step (according to the convergence process) of a quantity set by `--step-damp=` *real* (default value is 0.10). The minimum value set by `--max-damp=` *real* (default value is 0.90).

`fde.input` options:
- `epsilon=` *real*
- `max-iter=` *integer*
- `start-damp=` *real*
- `max-damp=` *real*
- `step-damp=` *real*

Embedding energy error

The embedding error in the total energy is computed as

$$ \Delta E = E^{\text{FDE}}[\tilde{\rho}_A; \tilde{\rho}_B] - E^{\text{DFT}}[\rho] \tag{19.13} $$

where E^{DFT} is the DFT total energy of total system with density $\rho(\mathbf{r})$. In order to compute ΔE as well as its components, the flag `--err-energy` must be used. This flag will required also the DFT calculation on the total system. In this case the converged SCF output file must be named `output.dscf`.

An example of session output for the computation of embedding energy and energy error decomposition, when `--err-energy` flag is present, is the following:

```
FDE ENERGY (TOTAL SYSTEM):   -200.99720391651 Ha
FDE BINDING ENERGY:                          4.960885 mHa
                                             3.113002 kcal/mol
FDE ENERGY ERROR:                            2.003352 mHa
ERROR ENERGY DECOMPOSITION
```

```
coulomb contribution:                -0.693026 mHa
nuclear contribution:                -3.136544 mHa
exchange-correlation contribution:   -1.156390 mHa
kinetic contribution:                 6.989320 mHa
```

where the FDE energy (E^{FDE}), the FDE binding energy, the embedding energy error (ΔE) and the error energy decomposition in its coulomb, nuclear, exchange-correlation and kinetic contributions are reported. This output is present at each FDE iteration.

`fde.input` option: `err-energy=1`

FDE with hybrid and orbital-dependent functionals

In order to use local approximations (19.1) and (19.8) with `FDE`, the flag `-f` *string* must be add to the options of the script. Here *string* denotes the local/semilocal approximation to hybrid or orbital-dependent exchange-correlation potentials in $v_{\mathrm{emb}}(\mathbf{r})$. All LDA/GGA functionals in `TURBOMOLE` can be considered as approximations.

For example, the command

```
FDE -f b-lyp --comp
```

can be used to approximate `bh-lyp` or `b3-lyp` hybrid non-additive potentials, while the command

```
FDE -f pbe --comp
```

approximates the `pbe0` hybrid non-additive potentials. Other combinations of functionals are not recommended (meta-GGA are not supported).

Finally, also calculations with the Local Hartree-Fock (LHF) potential can be performed. In this case the command

```
FDE -f becke-exchange --comp
```

can be used to approximate the LHF non-additive potential [320].

Equivalent command: `--func` *string*

`fde.input` option: `func=` *string*

19.4 Periodic Electrostatic Embedded Cluster Method

19.4.1 General Information

The Periodic Electrostatic Embedded Cluster Method (PEECM) functionality [325] provides electronic embedding of a finite, quantum mechanical cluster in a periodic, infinite array of point charges. It is implemented within HF and DFT energy and gradient `TURBOMOLE` modules: `dscf`, `grad`, `ridft`, `rdgrad`, and `escf`. Unlike embedding within a finite set of point charges the PEEC method always yields the correct electrostatic (Madelung) potential independent of the electrostatic moments of the point charges field. It is also significantly faster than the traditional finite point charges embedding.

19.4.2 Theoretical Background

Generally, the PEEC method divides the entire, periodic and infinite system into two parts, the inner (I) part, or so called cluster, and the outer (O) part which describes its environment. Thus, unlike "true" periodic quantum mechanical methods, PEECM primarily aims at calculations of structure and properties of localized defects in dominantly ionic crystals. The innermost part of the cluster is treated quantum mechanically (QM), whereas in the remaining cluster part cations are replaced by effective core potentials (ECPs) and anions by ECPs or by simply point charges. Such an "isolating" outer ECP shell surrounding the actual QM part is necessary in order to prevent artificial polarization of the electron density by cations which would otherwise be in a direct contact with the QM boundary. The outer part or the environment of the cluster is described by a periodic array of point charges, representing cationic and anionic sites of a perfect ionic crystal.

The electronic Coulomb energy term arising from the periodic field of point charges surrounding the cluster has the following form

$$J = \sum_{\mu\nu} \sum_{k}^{N\in \mathrm{UC}} \sum_{\vec{L}\in \mathrm{O}}^{\infty} D_{\mu\nu} q_k \int \frac{\mu(\vec{r})\nu(\vec{r})}{|\vec{r} - \vec{R}_k - \vec{L}|} d\vec{r},$$

where UC denotes the unit cell of point charges, $D_{\mu\nu}$ are elements of the density matrix, μ, ν are basis functions, q_k, $\vec{R}_k$ denote charges and positions of point charges, and $\vec{L}$ denote direct lattice vectors of the outer part O. It is evaluated using the periodic fast multipole method (PFMM) [326] which, unlike the Ewald method [327], defines the lattice sums entirely in the direct space. In general, PFMM yields a different electrostatic potential than the Ewald method, but the difference is merely a constant shift which depends on the shape of external infinite surface of the solid (i.e. on the way in which the lattice sum converges toward the infinite limit). However, this constant does not influence relative energies which are the same as obtained using the Ewald method, provided that the total charge of the cluster remains constant. Additionally, since the electrostatic potential within a solid is not a well defined

quantity, both the absolute total energies and orbital energies have no meaning (i.e. you cannot compare energies of neutral and charged clusters!).

19.4.3 Calculation Setup

There are three key steps in setting up a PEECM calculation. In the first step the periodic field of point charges has to be defined by specifying the point charges unit cell. Next step is the definition of the part infinite of point charges field that will be replaced by the explicit quantum mechanical cluster. Finally, the quantum mechanical cluster together with surrounding ECPs representing cationic sites as well as point charges representing anions is defined and put in place of the point charges. The input preparation steps can be summarized as follows

1. Dimensionality of the system is specified by the keyword `periodic` in the `$embed` section: `periodic 3` means a bulk three-dimensional system, `periodic 2` denotes a two-dimensional surface with an aperiodic z direction.

2. Definition of the unit cell of periodic point charges field is specified in the subsections `cell` and `content` of the `$embed` section.

3. Definition of the values of the point charges by specifying a charge value per species, using the subsection `charges`, or a charge value for each point charge, using the subsection `ch_list`. Note that only one of the subsections can be defined.

4. Definition of the part of point charges field that will be replaced by the QM cluster together with the isolating shell (ECPs, explicit point charges) is specified in the subsection `cluster` of the `$embed` section.

5. Definition of the quantum mechanical cluster as well as the surrounding ECPs and anionic point charges is included in the usual `$coord` section.

The following two examples show the definition of the point charges unit cells.

Example 1. Ca_4F_{19} cluster embedded in bulk CaF_2

In this example a QM cluster with the composition Ca_4F_{19}, surrounded by 212 ECPs and 370 explicit point charges, representing Ca^{2+} cations and F^- anions is embedded in a periodic field of point charges (+2 for Ca and -1 for F) corresponding to the CaF_2 fluorite lattice.

First, the program has to know that this is a three-dimensional periodic system. This is specified by the keyword `periodic 3`, meaning periodicity in three dimensions. The dimensions of the unit cell for bulk CaF_2 are given in the subsection `cell` of the `$embed` keyword. By default, the unit cell dimensions are specified in atomic units and can be changed to Å using `cell ang`. The positions of the point charges in the unit cell are specified in the subsection `content`. In this example positions are given in fractional crystal coordinates (`content frac`). You can change this by specifying `content` for Cartesian coordinates in atomic units or `content ang` for

Cartesian coordinates in Å. The values of point charges for Ca and F are given in the subsection `charges`.

```
$embed
periodic  3
cell
  10.47977 10.47977 10.47977 90.0 90.0 90.0
content frac
  F    0.00   0.00   0.00
  Ca  -0.25  -0.75  -0.75
  F    0.50  -0.50   0.00
  F    0.50   0.00  -0.50
  F    0.00  -0.50  -0.50
  F    0.50  -0.50  -0.50
  F    0.00   0.00  -0.50
  F    0.50   0.00   0.00
  F    0.00  -0.50   0.00
  Ca  -0.25  -0.25  -0.25
  Ca   0.25  -0.75  -0.25
  Ca   0.25  -0.25  -0.75
end
...
charges
  F   -1.0
  Ca   2.0
end
```

The above input defines a periodic, perfect, and infinite three-dimensional lattice of point charges corresponding to the bulk CaF_2 structure. In order to use this lattice for PEECM calculation we have to make "space" for our QM cluster and the isolating shell. This is done by specifying the part of the lattice that is virtually removed from the perfect periodic array of point charges to make space for the cluster. The positions of the removed point charges are specified in the subsection `cluster` of the `$embed` keyword. Note, that the position of the QM cluster and the isolating shell must **exactly** correspond to the removed part of the crystal, otherwise positions of the cluster atoms would overlap with positions of point charges in the periodic lattice, resulting in a "nuclear fusion".

```
cluster
  F    0.00000000000000   0.00000000000000   0.00000000000000
  Ca  -2.61994465796043  -2.61994465796043  -2.61994465796043
  Ca   2.61994465796043  -2.61994465796043   2.61994465796043
  Ca   2.61994465796043   2.61994465796043  -2.61994465796043
  Ca  -2.61994465796043   2.61994465796043   2.61994465796043
```

```
  F   -5.23988931592086   0.00000000000000   0.00000000000000
  F    0.00000000000000   0.00000000000000  -5.23988931592086
  F    5.23988931592086   0.00000000000000   0.00000000000000
  F    0.00000000000000  -5.23988931592086   0.00000000000000
  F    0.00000000000000   0.00000000000000   5.23988931592086
  F    0.00000000000000   5.23988931592086   0.00000000000000
  F   -5.23988931592086  -5.23988931592086   0.00000000000000
  F   -5.23988931592086   0.00000000000000  -5.23988931592086
  F   -5.23988931592086   0.00000000000000   5.23988931592086
  F   -5.23988931592086   5.23988931592086   0.00000000000000
  F    5.23988931592086  -5.23988931592086   0.00000000000000
  ...
```

repeated for $Ca_{216}F_{389}$

```
end
```

By default, the positions of point charges are specified in atomic units as Cartesian coordinates. You can change this by specifying `cluster frac` for fractional crystal coordinates or `cluster ang` for Cartesian coordinates in Å.

Finally, you have to specify the coordinates of the QM cluster along with the surrounding ECPs representing cationic sites and explicit point charges representing anions. This is done in the usual way using the `$coord` keyword.

```
$coord
    0.00000000000000      0.00000000000000      0.00000000000000      f
   -2.86167504097169     -2.86167504097169     -2.86167504097169      ca
    2.86167504097169      2.86167504097169     -2.86167504097169      ca
   -2.86167504097169      2.86167504097169      2.86167504097169      ca
    2.86167504097169     -2.86167504097169      2.86167504097169      ca
    0.00000000000000     -5.24009410923923      0.00000000000000      f
   -5.24009410923923      0.00000000000000      0.00000000000000      f
    0.00000000000000      5.24009410923923      0.00000000000000      f
    0.00000000000000      0.00000000000000     -5.24009410923923      f
    5.24009410923923      0.00000000000000      0.00000000000000      f
    0.00000000000000      0.00000000000000      5.24009410923923      f
    0.00000000000000     -5.24009410923923     -5.24009410923923      f
   -5.24009410923923     -5.24009410923923      0.00000000000000      f
    5.24009410923923     -5.24009410923923      0.00000000000000      f
    0.00000000000000     -5.24009410923923      5.24009410923923      f
    0.00000000000000      5.24009410923923     -5.24009410923923      f
 ...
```

repeated for $Ca_{216}F_{389}$

```
$end
```

This is the standard TURBOMOLE syntax for atomic coordinates. The actual distinction between QM cluster, ECP shell, and explicit point charges is made in the $atoms section.

```
$atoms
f  1,6-23
   basis =f def-TZVP
ca 2-5
   basis =ca def-TZVP
ca 24-235
   basis =none
   ecp   =ca ecp-18 hay & wadt
f  236-605
   basis =none
   charge= -1.00000000
```

In the example above the F atoms 1 and 6-23 as well Ca atoms 2-5 are defined as QM atoms with def-TZVP basis sets. The Ca atoms 24-235 are pure ECPs and have no basis functions (basis =none) and F atoms 236-605 are explicit point charges with charge -1, with no basis functions and no ECP.

This step ends the input definition for the PEECM calculation.

Example 2. Al_8O_{12} cluster embedded in α-Al_2O_3 (0001) surface

In this example a QM cluster with the composition Al_8O_{12}, surrounded by 9 ECPs representing Al^{3+} cations is embedded in a two-dimensional periodic field of point charges (+3 for Al and -2 for O) corresponding to the (0001) surface of α-Al_2O_3.

As in the first example, the program has to know that this is a two-dimensional periodic system and this is specified by the keyword periodic 2. The dimensions of the unit cell for the (0001) α-Al_2O_3 surface are given in the subsection cell of the $embed keyword. The aperiodic direction is always the z direction, but you have to specify the unit cell as if it was a 3D periodic system. This means that the third dimension of the unit cell must be large enough to enclose the entire surface in this direction. The unit cell dimensions are specified in Å using cell ang. The positions of the point charges in the unit cell are specified as Cartesian coordinates in Å (content ang). The values of point charges for Al and O are given in the subsection charges.

```
$embed
periodic 2
cell angs
  4.8043    4.8043   24.0000   90.0000   90.0000  120.0000
content ang
  Al    2.402142286    1.386878848    5.918076515
  Al   -0.000013520   -0.000003382    7.611351967
```

```
  Al   -0.000008912    2.773757219    8.064809799
  Al    2.402041674    1.386946321    0.061230399
  Al   -0.000005568   -0.000003223   10.247499466
  Al    2.402137518    1.386872172    9.977437973
  Al    0.000000070    2.773757935    5.390023232
  Al    0.000006283   -0.000005607    3.696748018
  Al    2.402151346    1.386879444    3.243290186
  Al    0.000100868    2.773690462   11.246870041
  Al   -0.000001982   -0.000005796    1.060600400
  Al    0.000004853    2.773764610    1.330662251
  O    -0.731205344    1.496630311    6.749288559
  O     0.743527174    1.296469569    8.957922935
  O     1.588027477    0.104536049   11.127140045
  O     1.471626759    2.779079437    6.749288559
  O     3.309734344   -0.004341011    8.957920074
  O     3.919768333    1.323050499   11.127141953
  O    -0.740424335    4.045563698    6.749289513
  O    -1.651123047    2.868478537    8.957910538
  O     1.698525310    2.733071804   11.127161026
  O     3.133347750    2.664006472    4.558811665
  O     1.658615232    2.864167213    2.350177050
  O     0.814115047    4.056100845    0.180959582
  O     0.930515707    1.381557465    4.558811188
  O     1.494558096    0.004332162    2.350180149
  O    -1.517625928    2.837586403    0.180958077
  O     3.142566681    0.115072958    4.558810234
  O    -0.751034439    1.292158127    2.350189686
  O     0.703617156    1.427564979    0.180938885
end
...
charges
  O   -2.0
  Al   3.0
end
```

The above input defines a periodic, perfect, and infinite two-dimensional lattice of point charges corresponding to the (0001) α-Al_2O_3 surface. In order to use the lattice for PEECM calculation we have to make "space" for our QM cluster and the surrounding ECP shell. This is done by specifying the part of the lattice that is virtually removed from the perfect periodic array of point charges to make space for the cluster. The positions of the removed point charges are specified in the subsection `cluster` of the `$embed` keyword. Note, that the position of the QM cluster must **exactly** correspond to the removed part of the crystal, otherwise positions of the cluster atoms would overlap with positions of point charges in the periodic lattice,

resulting in a “nuclear fusion”.

```
cluster ang
  Al    -0.000012482    5.547518253    9.977437973
  Al     2.402141094    6.934402943    8.064809799
  Al     2.402144432    4.160642624   10.247499466
  Al     4.804287434    5.547518253    9.977437973
  Al     2.402250767    6.934336185   11.246870041
  Al    -0.000005568    8.321288109   10.247499466
  Al     2.402137518    9.708164215    9.977437973
  Al     4.804294586    8.321288109   10.247499466
  O      0.907584429    4.156304836    8.957920074
  O      1.517618299    5.483696461   11.127141953
  O     -0.703624666    6.893717766   11.127161026
  O      3.145677090    5.457115650    8.957922935
  O      3.990177393    4.265182018   11.127140045
  O      0.751026928    7.029124260    8.957910538
  O      4.100675106    6.893717766   11.127161026
  O      0.743527174    9.617761612    8.957922935
  O      1.588027477    8.425827980   11.127140045
  O      3.309734344    8.316950798    8.957920074
  O      3.919768333    9.644342422   11.127141953
  O      5.555326939    7.029124260    8.957910538
  Al     4.804400921   11.094982147   11.246870041
  Al    -0.000008912    2.773757219    8.064809799
  Al    -2.402049065    6.934336185   11.246870041
  Al     4.804400921    2.773690462   11.246870041
  Al     2.402136564    4.160642624    7.611351967
  Al    -0.000013520    8.321288109    7.611351967
  Al    -0.000008912   11.095048904    8.064809799
  Al     7.206440926    6.934402943    8.064809799
  Al     4.804286480    8.321288109    7.611351967
end
```

The positions of point charges are specified in Å as Cartesian coordinates.

Finally, you have to specify the coordinates of the QM cluster along with the surrounding ECPs. This is done in the usual way using the `$coord` keyword.

```
$coord
   -0.00002358760000     10.48329315900000     18.85463057110000      al
    4.53939007480000     13.10412613690000     15.24028611330000      al
    4.53939638280000      7.86247730390000     19.36497297520000      al
    9.07879006320000     10.48329315900000     18.85463057110000      al
```

```
    4.53959732680000     13.10399998250000     21.25351019750000      al
   -0.00001052200000     15.72496001430000     19.36497297520000      al
    4.53938331720000     18.34577677080000     18.85463057110000      al
    9.07880357850000     15.72496001430000     19.36497297520000      al
    1.71508649490000      7.85428007030000     16.92802041340000      o
    2.86788376470000     10.36268741690000     21.02725683720000      o
   -1.32965829240000     13.02724227310000     21.02729288000000      o
    5.94446987180000     10.31245694970000     16.92802581990000      o
    7.54034461170000      8.06002818410000     21.02725323160000      o
    1.41923561090000     13.28312353520000     16.92800239300000      o
    7.74915508620000     13.02724227310000     21.02729288000000      o
    1.40506312580000     18.17494056150000     16.92802581990000      o
    3.00093786570000     15.92251179600000     21.02725323160000      o
    6.25449323900000     15.71676368210000     16.92802041340000      o
    7.40729073370000     18.22517102690000     21.02725683720000      o
   10.49804944110000     13.28312353520000     16.92800239300000      o
    9.07900452260000     20.96648359440000     21.25351019750000      al
   -0.00001684120000      5.24164297480000     15.24028611330000      al
   -4.53921616520000     13.10399998250000     21.25351019750000      al
    9.07900452260000      5.24151682240000     21.25351019750000      al
    4.53938151440000      7.86247730390000     14.38337475740000      al
   -0.00002554910000     15.72496001430000     14.38337475740000      al
   -0.00001684120000     20.96660974680000     15.24028611330000      al
   13.61820356690000     13.10412613690000     15.24028611330000      al
    9.07878826040000     15.72496001430000     14.38337475740000      al
$end
```

This is the standard `TURBOMOLE` syntax for atomic coordinates. The actual distinction between QM cluster and ECP shell is made in the `$atoms` section.

```
$atoms
al 1-8
   basis =al def-SV(P)
o  9-20
   basis =o def-SV(P)
al 21-29
   basis =none
   ecp   =al ecp-10 hay & wadt
```

In the example above the Al atoms 1-8 and O atoms 9-20 are defined as QM atoms with def-SV(P) basis sets. The Al atoms 21-29 are pure ECPs and have no basis functions (`basis =none`).

This step ends the input definition for the PEECM calculation.

19.5 Polarizable embedding calculations

Realizable embedding (PE) calculations are a based on a hybrid model of quantum mechanics and molecular mechanics (QM/MM) in which the classical region is represented by an electrostatic potential with up to octupole moments and induced point dipole moments. The main improvement over the more common QM/MM approaches without polarizable MM sites can be found for the description of electronic excitations but also for any other process which causes a significant change in the QM density and which is accompanied by a fast response of the environment.

In TURBOMOLE, ground state energies computed with the `dscf`, `ridft`, and `ricc2` module and electronic excitation properties based on RI-CC2 and RI-ADC(2) are implemented. The excited state analytic gradients are also available at the RI-ADC(2) level. The general theory is presented in ref. [328] and [329,330], the PERI-CC2 model and the TURBOMOLE implementation is described in ref. [308].

19.5.1 Theory

In the following, only the most important ideas are presented and discussed with a focus on the PERI-CC2 model. The essential concept is the introduction of an environment coupling operator $\hat{G}(\mathbf{D}^{\mathrm{CC}})$

$$\hat{G}(\mathbf{D}^{\mathrm{CC}}) \quad = \quad \hat{G}^{\mathrm{es}} + \hat{G}^{\mathrm{pol}}(\mathbf{D}^{\mathrm{CC}}) \tag{19.14}$$

with the electrostatic contribution

$$\hat{G}^{\mathrm{es}} \quad = \quad \sum_{m=1}^{M} \sum_{k=0}^{K} \sum_{pq} \Theta_{m,pq}^{(k)} \mathbf{Q}_m^{(k)} \hat{E}_{pq} \tag{19.15}$$

and the polarization contribution

$$\hat{G}^{\mathrm{pol}}(\mathbf{D}^{\mathrm{CC}}) \quad = \quad \sum_{u=1}^{U} \sum_{pq} \Theta_{u,pq}^{(1)} \mu_u^{\mathrm{ind}}(\mathbf{D}^{\mathrm{CC}}) \hat{E}_{pq} \,. \tag{19.16}$$

Here, $\Theta_{m,pq}^{(k)}$ are multipole interaction integrals of order k and μ_u^{ind} are the induced dipoles which can be obtained from the electric field $\mathbf{F}_u$ and the polarizability $\boldsymbol{\alpha}_u$ at a site u:

$$\mu_u^{\mathrm{ind}} = \mathbf{F}_u \boldsymbol{\alpha}_u \tag{19.17}$$

Because the induced dipoles depend on the electron density and *vice versa*, their computations enter the self-consistent part of the HF cycle. Introducing $\hat{G}(\mathbf{D}^{\mathrm{CC}})$

into standard equations for the HF reference state and the CC2 equations leads to a general PE-CC2 formulation. To maintain efficiency, a further approximation has been introduced which makes the operator only dependent on a CCS-like density term. These general ideas define the PERI-CC2 model and allow to formulate the corresponding Lagrangian expression

$$\begin{aligned} L_{\text{PERI-CC2}}(\mathbf{t},\bar{\mathbf{t}}) &= E_{\text{PE-HF}} + \langle \text{HF}|\hat{W}(\hat{T}_1 + \hat{T}_2 + \frac{1}{2}\hat{T}_1^2)|\text{HF}\rangle + \\ &\quad \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\tilde{W} + [\hat{F}^{\text{PE}}, \hat{T}_1] + [\tilde{W}, \hat{T}_2]|\text{HF}\rangle + \\ &\quad \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|\tilde{W} + [\hat{F}^{\text{PE}}, \hat{T}_2]|\text{HF}\rangle \\ &\quad -\frac{1}{2}\sum_{uv} F_u^{\text{elec}}(\mathbf{D}^{\Delta\prime}) R_{uv} F_v^{\text{elec}}(\mathbf{D}^{\Delta\prime}) \end{aligned} \tag{19.18}$$

from which all PERI-CC2 equations including the linear response terms may be derived. Note that the dependency on the density couples the CC amplitude and multiplier equations for the ground state solution vector.

This coupling is avoided by the simplified polarizable embedding method (sPE) described in ref. [331].

$$\begin{aligned} L_{\text{sPERI-CC2}}(\mathbf{t},\bar{\mathbf{t}}) =& E^{\text{PE-HF}} \\ &+ \langle \Lambda|\hat{g}_{\text{N}} + \hat{F}_{\text{N}}^{\text{PE}} + \hat{G}_{\text{N}}^{\text{pol}}(\mathbf{D}^{\Delta\text{CC}})|\text{CC}\rangle \ . \end{aligned} \tag{19.19}$$

The subscript N indicates that the operator is normal ordered with respect to the Hartree–Fock state. Here, a polarization operator $\hat{G}^{\text{pol}}(\mathbf{D})$ was introduced,

$$\hat{G}^{\text{pol}}(\mathbf{D}^{\Delta\text{CC}}) = -\frac{1}{2}\sum_{uv}^{M} \left(\hat{\mathbf{F}}_u\right)^{\top} \mathbf{R}_{uv} \mathbf{F}_v^{\text{el}}(\mathbf{D}^{\Delta\text{CC}}) \ . \tag{19.20}$$

which depends on the elements of the difference density matrix $\mathbf{D}^{\Delta\text{CC}}$. These are defined as

$$D_{pq}^{\Delta\text{CC}} = \langle \text{HF}|\hat{a}_p^{\dagger}\hat{a}_q|\text{CC}\rangle - D_{pq}^{\text{HF}} \ , \tag{19.21}$$

hence, they do not depend on the Lagrangian multipliers.

19.5.2 Computational details: SCF calculations

To carry out a PE-SCF calculation with the DSCF or RIDFT module, you have to specify the following in the `control` file:

```
$point_charges pe [options]
<length unit>
```

```
<no. MM sites> <order k> <order pol> <length exclude list>
<list of MM sites: exclude list, xyz coords, multipole mom., pol. tensor>
```

length unit specifies the unit for the MM site coordinates (use `AA` or `AU`)

no. MM sites the amount of MM sites (length of the list)

order k the order of multipoles used (0: point charges, 1: dipole moments, 2: quadrupole moments, 3: octupole moments)

order pol the treatment of polarizabilities (0: none, 1: isotropic, 2: anisotropic)

length exclude list number of elements in the exclude list

list of MM sites each MM sites is described on one line, entries separated by blanks; first entry is the exclude list of with as much elements as defined in the head line (If the first element in the exclusion list of one site occurs in the exclude list of another site, they do not contribute to each others polarization); next follows the MM site coordinates in (x,y,z positions), the point charge, the dipole moment (for $k \geq 1$, x,y,z component), the quadrupole moment (for $k \geq 2$, xx, xy, xz, yy, yz, zz component), the octupole moment (for $k = 3$, xxx, xxy, xxz, xyy, xyz, xzz, yyy, yyz, yzz, zzz component), the polarizability (one component for pol-order 1, xx, xy, xz, yy, yz, zz component for pol-order 2)

An example for a polarizable embedding with coordinates given in Å, point charges and isotropic polarizabilities:

```
$point_charges pe
AA
6 0  1  1
  39   -0.2765102481    2.5745845304    3.5776314866    0.038060  15.217717
  39    1.3215071687    2.3519378014    2.8130403183   -0.009525  14.094642
  39   -0.5595582934    1.2645007691    4.7571719292   -0.009509  14.096775
  39   -1.5471918244    2.5316479230    2.3240961995   -0.009519  14.096312
  39   -0.3207417883    4.1501938400    4.4162313889   -0.009507  14.096476
  41   -1.1080691595    4.9228723099   -1.6753825535    0.038060  15.217717
  41   -0.9775910525    6.5274614891   -2.4474576239   -0.009525  14.094642
  41   -2.5360480539    4.8923046027   -0.6040781123   -0.009509  14.096775
  41    0.3630448878    4.6028736791   -0.7155647205   -0.009519  14.096312
  41   -1.2817317422    3.6689143712   -2.9344225518   -0.009507  14.096476
```

All values are given in atomic units (except coordinates if stated otherwise). These data are mandatory. An alternative input format can also be used by specifying `daltoninp` as option on the `$point_charges` line. The format is completely compatible with the current Dalton 2015 input format (The definition can be found in the

manual of the Dalton program at https://daltonprogram.org/documentation/. See there for more information).

In addition, you can specify further options on the same line as the `$point_charges` flag. These are:

- `rmin=<float>`: minimum distance between an active MM site and any QM center (in a.u.), treatment is handle by option `iskip`, (DEFAULT: 0.00 a.u.)
- `iskip=(1,2)` : treatment of too close MM sites
  - (1) zeroing all contributions
  - (2) distribute values to nearest non-skipped MM site (DEFAULT)
- `rmax=<float>`: maximum distance between an active MM site and QM center of coordinates (in a.u.), sites too far away are skipped (zeroed) (DEFAULT: 1000.00 a.u.)
- `nomb`: no treatment of many body effects between induced dipoles (all interaction tensors on the off-diagonal of the response matrix are set to Zero); works best with isotropic polarizabilities, speeds up calculations (especially for large response matrices), has reduced accuracy, not well tested so far
- `longprint=(1,2,3)`: sets a flag for additional output
  - (1) print all MM site input information
  - (2) additionally: print all induced dipoles due to nulcei/multipole/electron electric filed
  - (3) additionally: print response matrix
- `file=<input file>`: specifies a file from which the data group `$point_charges` is read. Note that all options which are following on the line in the `control` file are then ignored because reading continues in the input file (But here, further options can be specified after the `$point_charges` flag). The file has to start with `$point_charges` as top line and should be finished with `$end`
- `ccdens`: activates the simplified polarizable embedding method

Limitations with respect to standard SCF computations:

- In PE-SCF computations, symmetry cannot be exploited.
- PE-SCF computations do not work in parallel (MPI parallelization).
- For two-component all-electron calculations, the decoupling of the embedding potential is neglected, i.e. it is affected by a picture-change error just like the two-electron integrals.

The energy of a PE-SCF calculation printed in the output contains the following terms:

$$E_{\text{PE-SCF}} = E_{\text{QM}} + E_{\text{QM/MM,es}} + E_{\text{pol}} \tag{19.22}$$

Here, E_{QM} is the energy of the quantum mechanical method of your choice, $E_{\text{QM/MM,es}}$ the electrostatic interaction energy between the QM and the MM region, and E_{pol} the energy gain due to the total of induced dipole moments. If necessary, missing terms can be computed without knowledge of the electron distribution.

At the moment, TURBOMOLE does not offer the possibility to generate the necessary potentials or to create a potential file from a set of coordinates. Embedding potentials can be obtained from literature or generated by approaches like the LoProp method. [332] Atom centered polarizabilities are also available from other methods or from experiment. Finally, there are some polarizable force fields which, in principle, can be used for the PE method (for example, the AMOEBA force field).

19.5.3 Computational details for post-SCF methods

PERI-CC2 calculations:

Apart from the definition of the embedding described above, the input for PERI-CC2 calculations is the same as without polarizable embedding.

There are several limitations for the use of PERI-CC2:

- only ground state energies, excitation energies and transition moments are supported (no other properties or gradients and so on)
- no use of symmetry
- no MPI parallelization is available (but SMP binaries work)
- open-shell systems are not covered (exception: two-component references)

PE-MP2 within PTED Reaction-Field Scheme:

In addition to the PERI-CC2 method, the calculation of the ground-state energy is available at PE-MP2 level within the framework of PTE and PTED reaction-field schemes. The implementation of PE-MP2 using the PTED approach is inspired by the work published by Lunkenheimer and Köhn. [333] For the detail about the PTED reaction-field scheme read section 19.2.5. For the calculation of the ground-state energy the following data groups must be included in the `control` in addition to the PE data groups given in section 19.5.2 and the `$ricc2` data group:

```
$response
  fop relaxed
```

```
$reaction_field
  PTED
  cycle
```

The `cycle` flag in the `$reaction_field` data group controls the PTED macro-iterations between the SCF and post-SCF calculations. In order to invoke the PTED-PE-MP2 calculations, the `pecc2` script must be used. The combination of PTED-PE-MP2 with SCS and SOC is also available. Furthermore, the calculation of ground-state energy is possible for closed-shell and open-shell systems with the SMP (OpenMP) and MPI parallelizations.

PE-ADC(2) within post-SCF Reaction-Field Scheme

:The new PE-ADC(2) method [334] implemented in the framework of the post-SCF reaction field scheme makes the following calculations available in the `ricc2` module for both closed-shell and open-shell systems:

- Vertical excitation energy and transition moments
- Excited-state analytic gradients
- Excited-state properties

For these calculations, in addition to the typical data groups `$ricc2`, `$excitations` and `$point_charges`, the following keywords should be added to the `control` file:

```
$reaction_field
  post-SCF
  ccs-like
```

Chapter 20

Molecular Properties, Wavefunction Analysis, and Interfaces to Visualization Tools

20.1 Molecular Properties, Wavefunction Analysis, and Localized Orbitals

Molecular properties (electrostatic moments, relativistic corrections, population analyses for densities and MOs, construction of localized MOs, NTOs, etc.) can be calculated with `proper`. This program is menu-driven and reads the input that determines which properties are evaluated from standard input (i.e. the terminal or an input file if started as `proper < inputfile`). The `control` file and files referenced therein are only used to determine the molecular structure, basis sets, and molecular orbitals and to read results computed before with other programs.

`proper` is a post-processing tool, mainly intended for interactive use which reads (almost all) its input from the terminal so that it is (usually) not necessary to modify the `control` file.

Several functionalities are also integrated in the programs that generate MOs or densities and can be invoked directly from the modules `dscf`, `ridft`, `rimp2`, `mpgrad`, `ricc2` and `egrad`, if corresponding keywords are set in the `control` file. If one wants to skip the MO- or density generating step for `dscf`, `ridft`, `rimp2`, `mpgrad`, `ricc2` it is possible to directly jump to the routine that carries out the analyses by starting the program with "`<program> -proper`". (For `ricc2` it is, however, recommended to use instead `ricctools -proper`.) Currently, the respective keywords have to be inserted by hand (not with `define`) in the `control` file.

Here we briefly present the functionalities. A detailed description of the keywords that can be used in combination with the `-proper` flag is found Section 23.2.28.

20.1.1 Selection of densities

The `proper` program tries on start to read all densities that have been pre-calculated with any of the other programs of the `TURBOMOLE` package, prints a list with the densities that have been found and selects one of them for the calculation of properties. The default choice can be changed in the menu that is entered with the option `dens`. Here one can also list or edit additional attributes of the densities or build linear combinations of the available densities to form differences and superposition of densities.

The feature can thus be used to evaluate difference densities between the ground and excited electronic states or differences between densities calculated with different electronic structure methods.

The selected density can then be used in the subsequent menues to evaluate a variety of properties as expectation values or on a grid of points to generate interface files for visualization.

20.1.2 Electrostatic moments

Use the `mtps` option in the `eval` menu in `proper`, or add the `$moments` keyword control file when you start a program with the `-proper` flag to evaluate electrostatic moments. Up to quadrupole moments are calculated by default, on request also octupole moments are available. By default unnormalized traced Cartesian moments are calculated which are defined as

$$Q^{(n)}_{\alpha\beta\ldots\nu} = \int d\mathbf{r}\rho(\mathbf{r})r_\alpha r_\beta \ldots r_\nu \ ,$$

where $\rho(\mathbf{r})$ is the charge density as position $\mathbf{r}$. With the option `Buckingham` one can request in addition the computation of Cartesian traceless (Buckingham) multipole moments defined as:

$$M^{(n)}_{\alpha\beta\ldots\nu} = \frac{(-1)^n}{n!}\int d\mathbf{r}\rho(\mathbf{r})r^{2n}\frac{d^n}{dr_\alpha dr_\beta \ldots dr_\nu}\frac{1}{r}$$

20.1.3 Relativistic corrections

The option `relcor` in the `eval` menu of `proper` or the keyword `$mvd` (when starting a program with the `-proper` flag) initiate the calculation of relativistic corrections. With the `-proper` flag they are calculated for the SCF or DFT total density in case of `dscf` and `ridft`, for the SCF+MP2 density in case of `rimp2` and `mpgrad` and for that of the calculated excited state in case of `egrad`. Quantities calculated are the expectation values $\langle p^2\rangle$, $\langle p^4\rangle$ and the Darwin term $\langle\sum_A 1/Z_A * \rho(R_A)\rangle$. Note,

that at least the Darwin term requires an accurate description of the cusp in the wave function, thus the use of basis sets with uncontracted steep basis functions is recommended. Moreover note, that the results for these quantities are not really reasonable if ECPs are used (a respective warning is written to the output).

20.1.4 Population analyses

For population analyses enter the `pop` menu of `proper`. The available options and parameters that can be specified are the same as those for the `$pop` keyword (vide infra). If an electronic structure program is started with the `-proper` flag the population analyses is requested with the keyword `$pop`.

`mulliken` or `$pop` without any extension start a Mulliken population analysis (MPA). For `-proper` the analysis is carried out for all densities present in the respective program, e.g. total (and spin) densities leading to Mulliken charges (and unpaired electrons) per atom in RHF(UHF)-type calculations in `dscf` or `ridft`, SCF+MP2 densities in `rimp2` or `mpgrad`, excited state densities in `egrad`. Suboptions (see Section 23.2.28) also allow for the calculation of Mulliken contributions of selectable atoms to selectable MOs including provision of data for graphical output (simulated density of states).

With `$pop nbo` a Natural Population Analysis (NPA) [335] is done. Currently only the resulting charges are calculated.

With `$pop paboon` a population analyses based on occupation numbers [336] is performed yielding "shared electron numbers (SENs)" and multicenter contributions. For this method always the total density is used, i.e. the sum of alpha and beta densities in case of UHF, the SCF+MP2-density in case of MP2 and the GHF total density for (two-component-)GHF. Note that the results of such an analysis may depend on the choice of the number of modified atomic orbitals ("MAOs"). By default, the number of MAOs is chosen such that they are reasonable in most cases (see Section 23.2.28). Nevertheless it is recommended to read carefully the information concerning MAOs given in the output before looking at the results for atomic charges and shared electron numbers. For different ways of selecting MAOs see Section 23.2.28.

With `$pop wiberg` Wiberg bond indices (WBI) [337] are calculated.

20.1.5 Generation of localized MOs

The option `lmos` in the `mos` menu of `proper` and the keyword `$localize` with the `-proper` flag trigger the calculation of localized molecular orbitals. Per default a Boys localization including all occupied MOs is carried out (i.e. the squared distance of charge centers of different LMOs is maximized). Alternative localization methods are Pipek-Mezey, Intrinsic Bond Orbitals (IBOs), and minimizations of (powers) of the second or fourth orbital moment.

pm Pipek-Mezey localization, maximizes the sum of the squared orbital charges on

the nuclei:

$$L_{\texttt{PM}} = \sum_i \sum_A^{atoms} \left(q_i^A\right)^2$$

It has the lowest operation count of all the implemented localization procedures, but gives only well-localized orbitals for basis sets without diffuse functions. In difference to Foster-Boys (see below) it conserves the $\pi\sigma$ separation.

boys Foster-Boys localization, minimizes the average orbital spread:

$$L_{\texttt{FB}} = \sum_i \langle i|(\mathbf{r} - \mathbf{r}_i)^2|i\rangle$$

where $\mathbf{r}_i = \langle i|\mathbf{r}|i\rangle$ is the center of the LMO. It is almost as fast as Pipek-Mezey, but stable with the basis set. Its disadvantage relative to Pipek-Mezey and IBO localization is that it breaks the $\pi - \sigma$ separation for double bonds.

sm second moment localization, minimizes a power of the sum of orbital spreads:

$$L_{\texttt{SM},n} = \sum_i \langle i|(\mathbf{r} - \mathbf{r}_i)^2|i\rangle^n$$

The exponent is defined with the additional option `exp=`n where n must be an integer > 0. For $n = 1$ SM localization is identical to Foster-Boys. Values $n > 1$ cause a larger penalty for LMOs with larger spreads and thereby reduce the variance of the LMO spreads at the price of a small increase of the average spread and a small increase in the computational costs.

fm fourth moment localization [338], mimimizes a power of the sum of the fourth central moments:

$$L_{\texttt{FM},n} = \sum_i \langle i|(\mathbf{r} - \mathbf{r}_i)^4|i\rangle^n$$

The exponent is again defined with the additional option `exp=`n where n must be an integer > 0. Fourth moment localization produces in contrast to PM, IBO, FB, and SM localization LMOs with smaller tails Values $n > 1$ cause a larger penalty for LMOs with a larger Kurtosis and thereby reduce the variance of the LMO Kurtosis at the price of a small increase of the average Kurtosis. The computational costs for FM localization are 3–5 times larger than for SM localization.

ibo intrinsic bond orbital localization [339], maximizes the sum of the fourth power of the orbital charges on the nuclei:

$$L_{\texttt{IBO}} = \sum_i \sum_A^{atoms} \left(q_i^A\right)^4$$

The IBO localization is similar to Pipek-Mezey localization, but the localization is carried out in in a minimal basis of intrinsic atomic orbitals (IAOs, see below) which make the procedure stable with respect to basis basis extension. The

exponent of 4 reduces the problem of (nearly) degenerate minima of the PM functional. The cost of the IBO localization is for large systems and/or basis sets dominated by the costs for the construction of the IAOs.

As output one gets localized MOs (written to files `lmos` or `lalp`/`lbet` in UHF cases). In addition information about the dominant contributions of canonical MOs to the LMOs and a population analysis and, if requested, also about the centers, the spread and the Kurtosis of the LMOs is written to standard output.

20.1.6 Intrinsic Bond Orbitals Analysis

The option `ibos` in the `mos` menu of proper initiates the computation of intrinsic bond orbitals [339] (IBOs). Instead of a Mulliken PA in the AO basis the contributions a population analysis in the basis of the intrinsic atomic orbitals [339] (IAOs) and an IAO atomic charge analysis is done. The start guess for the IBO optimization is slightly different then the one used with the option `lmos`, which in the case of degenerate solutions (core orbitals, symmetry-degenerate LMOs located at the same centers) can lead to slightly different IBOs. As proposed in Ref. [339] the IAOs are obtained by projection of the occupied orbitals onto pre-computed atomic orbitals from Hartree-Fock calculations on the isolated atoms in the correlation consistent triple-ζ basis (cc-pVTZ and cc-pVTZ-PP for pseudo-potential basis sets). These are available for most main group and transition metal atoms, for atoms beyond Kr with the standard pseudo-potential cores. If additional definitions are needed, they can be added to the basis set library.

Available reference orbitals for IAOs:

Atoms	all electron	default core	large core
H–He	1s / cc-pVTZ		
Li–Be	2s / cc-pVTZ		
B–Ne	2s1p / cc-pVTZ		
Na–Mg	3s1p / cc-pVTZ		
Al–Ar	3s2p / cc-pVTZ		
K	4s2p / TZV	2s1p / LANL2DZ ECP	1s / ecp-18-sdf
Ca	4s2p / cc-pVTZ		
Sc–Ni	4s2p1d / cc-pVTZ		
Cu–Zn	4s2p1d / cc-pVTZ	2s1p1d /cc-pVTZ-PP	
Ga–Kr	4s3p1d / cc-pVTZ	2s2p1d /cc-pVTZ-PP	
Rb	—		
Y–Cd	—	2s1p1d / cc-pVTZ-PP	
In–Xe	—	2s2p1d / cc-pVTZ-PP	
Cs–Lu	—		
Hf–Hg	—	2s1p1d / cc-pVTZ-PP	
Tl–Rn	—	2s2p1d / cc-pVTZ-PP	

For transition (earth) alkali and transition metal atoms the inclusion of the highest s orbital ($2s$ for Li–Be, $3s$ for Na–Mg, and $4s$ for K–Zn) can cause problems for systems

that contain the respect cations. In the cations these s orbitals are (essentially) unoccupied and lead the unphysical delocalized IAOs. To avoid this problem it is possible to restrict the reference basis for the construction of the IAOs by adding to the `control` file data group `$iaoopts`:

```
$iaoopts
   subset=3s2p1d <list of atom indes>
```

20.1.7 Natural transition orbitals

For excited states calculated at the CIS (or CCS) level the transition density between the ground and an excited state

$$E_{ia} = \langle \Psi_{ex} | a_i^\dagger a_a | \Psi_{ex} \rangle \tag{20.1}$$

can be brought to a diagonal form through a singular value decomposition (SVD) of the excitation amplitudes E_{ia}:

$$[\mathbf{O}^\dagger \mathbf{E} \mathbf{V}]_{ij} = \delta_{ij} \sqrt{\lambda_i} \tag{20.2}$$

The columns of the matrices $\mathbf{O}$ and $\mathbf{V}$ belonging to a certain singular value λ_i can be interpreted as pairs of occupied and virtual natural transition orbitals [340, 341] and the singular values λ_i are the weights with which this occupied-virtual pair contributes to the excitation. Usually electronic excitations are dominated by one or at least just a few NTO transitions and often the NTOs provide an easier understanding of transition than the excitation amplitudes E_{ia} in the canonical molecular orbital basis.

From excitation amplitudes computed with the `ricc2` program NTOs and their weights (the singular values) can be calculated with the `ntos` option in the `mos` menu of `proper` or with `ricctools`. E.g. using the right eigenvectors for the second singlet excited state in irrep 1 with:

```
ricctools -ntos CCRE0-2--1---1
```

Both programs store the results for the occupied and virtual NTOs in files named, respectively, `ntos_occ` and `ntos_vir`. The option `nto` in the `grid` menu of the `proper` program can used to evaluate NTOs for visualization on a grid of points.

Note that the NTO analysis ignores for the correlated methods (CIS(D), ADC(2), CC2, CCSD, etc.) the double excitation contributions and correlation contributions to the ground state. This is no problem for single excitation dominated transition out of a "good" single reference ground state, in particular if only a qualitative picture is wanted, but one has to be aware of these omissions when using NTOs for states with large double excitation contributions or when they are used for quantitative comparisons.

Difference densities based on natural transition orbitals If the excitation vectors have been obtained starting from a GHF reference, the NTOs are complex and contain contributions from both spin function. Moreover, the transitions are usually dominated by two NTOs at least. Thus, the interpretation of 2c-NTOs may become difficult. To get a simple picture of the transition at hand still, approximate difference densities can be computed according to

$$\rho(\mathbf{r})_n = \Re\Big(\sum_{ab}^{N_{\text{vir}}} \phi_a(\mathbf{r})^\dagger \phi_b(\mathbf{r}) \sum_{i}^{N_{\text{occ}}} C_i^{a*} C_i^b - \sum_{ij}^{N_{\text{occ}}} \phi_i(\mathbf{r})^\dagger \phi_j(\mathbf{r}) \sum_{a}^{N_{\text{vir}}} C_i^{a*} C_j^a \Big) . \tag{20.3}$$

The first term corresponds to the increase of the occupation of the virtual NTOs, while the second term corresponds to the decrease of the occupation of the occupied NTOs.
This approximate difference density is available for excitation vectors obtained with the following methods: CCS/CIS, CIS(D∞), ADC(2) and CC2. Symmetry other than C_1 is currently not supported. Note that the approximate difference densities are based on the same approximations as the NTOs, namely ignoring correlation and double excitation contributions.

From excitation amplitudes computed with the `ricc2` program the approximate difference densities are computed with `ricctools`. E.g. using the right eigenvectors for the second singlet excited state in irrep 1:

```
ricctools -diffden CCRE0-1--1---2
```

This resulting density file can be visualized using the analysis mode of the `ricc2` program as described in Section 10.3.3, e.g. by adding the following lines to the `control` file

```
$anadens
 calc my_approx_diffden from
 1d0 cc2-1a-002-approxdiffden.cao
$pointval
```

and running

```
ricc2 -fanal
```

20.1.8 Corresponding Spin Orbitals

The analysis of spin-unrestricted open-shell calculations (UHF or UKS) are often hampered by the fact that the spatial parts of canonical α- and β-spin orbitals can differ a lot. This makes it difficult to identify singly occupied molecular orbitals (SOMOs) and to distinguish almost doubly occupied orbitals from SOMOs and strongly

spin-polarized "magnetic orbitals" that might be present in multireference situations as e.g. during the dissociation of covalent bonds.

Corresponding spin orbitals (CSOs) are defined through unitary transformations of the α- and the β-spin orbitals that maximize the similarity (or overlap) of the spatials parts of α- and β-spin orbitals with same indezes and thereby rotate strongly-polarized "magnetic" parts and SOMOs into a small set of orbitals. This is achieved by a singular value decomposition (SVD) of the overlap matrix between the occupied α- and β-spin orbitals:

$$\mathbf{U}^T\mathbf{S}^{\alpha\beta,oo}\mathbf{V} = \mathbf{s} \tag{20.4}$$

where $s_{ij} = \delta_{ij}s_i$ and s_i are the singular values. The occupied CSOs $\tilde{\phi}_k^\sigma$ are then obtained by transforming the canonical MOs ϕ_k^σ with the unitary matrices $\mathbf{U}$ and $\mathbf{V}$:

$$\tilde{\phi}_k^\alpha = \sum_j \phi_j^\alpha U_{jk} \tag{20.5}$$

$$\tilde{\phi}_k^\beta = \sum_j \phi_j^\beta V_{jk} \tag{20.6}$$

The CSOs are sorted according to decreasing singular values which range from 1.0 to 0.0 and can be classified as follows:

- For CSO pairs with singular values close to 1.0 the spatials parts of the α- and β-spin orbitals are almost the same and they correspond thus to closed-shell MOs in spin-restricted calculations.
- For the majority spin there will be $|n_\alpha - n_\beta|$ CSOs with singular values of 0.0: These are the SOMOs.
- For CSO pairs with singular values significantly lower than 1.0 (but non-zero) the spatial parts of α- and β-spin orbitals are significantly different: these are strongly spin-polarized "magnetic orbitals". Their presence indicates multireference situations.

In the implementation in TURBOMOLE similar transformations are applied to the virtual α- and β-spin orbitals, but in this case sorted according to increasing singular values for $\mathbf{S}^{\alpha\beta,vv}$. Thus, in the output CSO sets the unoccupied counterparts of the SOMOs (in the CSOs for the minority spin) have the same spatials parts as the SOMOs. They are then followed by strongly spin-polarized unoccupied CSOs and the non-polarized CSO pairs that correspond to the virtuals orbitals of spin-restricted calculations have the highest indezes.

By default the coefficients of the CSOs are written to file in the cartesian AO basis (as it is done for NTOs). With the option `lsymao` one can request that the output is in the symmetry-adapted spherical AO basis (as for canonical MOs).

For single-reference cases without strongly spin-polarized "magnetic orbitals" one might want to isolate only the SOMOs, but keep the remaining MOs that correspond

to the closed-shell and virtuals MOs of spin-restricted calculations close to canonical MOs. The can be done obtained wit the option `match`. If switched on, the occupied orbitals for the minority spin are kept in the canonical basis. For the majortiy spin, the CSOs are first calculated as described above and then the $|n_\alpha - n_\beta$ CSOs with the largest singular values are transformed with the inverse of the transformation matrix for other spin to make as similar as possible to the canonical MOs of the minority spin. Again, similar transformations are applied to the virtuals MOs.

20.1.9 Orbitals for weakly interacting fragments

The `frag` option in the `mos` menu of `proper`allows to extract from a supermolecular HF or DFT calculation on weakly interaction fragments MOs and occupation numbers for the individual fragments. This is can be in particular usefull for supermolecular UHF or UKS calculations to obtain information on the spin states of the fragments. The option requires as prerequisite that the control file contains a valid input for the `$frag` data group assigning all atoms to fragments. Furthermore, the interaction between the fragments must be weak so that after localization each occupied LMO can be assigned to one of the fragments. All further input options for `frag` are passed as input to the calculation of LMOs and have the same meaning as for the `LMO` option.

For each fragment the program will generate files containing the orbitals and coordinates and simple control files containing the orbital occupations, basis set information and references to the coordinate and MO data groups that can be used to run calculations for the individual fragments.

20.1.10 Fit of charges due to the electrostatic potential:

`$esp_fit` fits point charges at the positions of nuclei to electrostatic potential arising from electric charge distribution (for UHF cases also for spin density, also possible in combination with `$soghf`). For this purpose the ("real") electrostatic potential is calculated at spherical shells of grid points around the atoms. By default, Bragg-Slater radii, r_{BS}, are taken as shell radii.

A parametrization very close to that suggested by Kollman (a multiple-shell model with shells of radii ranging from 1.4*r_{vdW} to 2.0*r_{vdW}, r_{vdW} is the van-der-Waals radius; U.C. Singh, P.A. Kollman, J. Comput. Chem. 5(2), 129-145 (1984)) is used if the keyword is extended:

```
$esp_fit kolman
```

20.2 Interfaces to Visualization Tools

There are several possibilities to visualize structures, vibrational frequencies, molecular orbitals, densities and a large number of properties.

The easiest one is to use the graphical user interface of TURBOMOLE TmoleX. TmoleX is either included in your TURBOMOLE distribution or can be downloaded for free from the BIOVIA web site TmoleX version (with DEMO version of TURBOMOLE).

TmoleX can be used to create input, run TURBOMOLE jobs on either the local machine or on remote Linux systems or clusters (directly or using queuing systems) and to visualize the results.

Visualization of Molecular Geometries

The tool t2x can be used to convert the atomic coordinates stored in the $grad and $coord data groups into the xyz-format, which is supported by most viewers, e.g. jmol. Typing

```
t2x > opt.xyz
```

in a directory containing the control file generates a series of frames using the information in $grad. Note t2x writes to standard output which here is redirected to a file. If you are only interested in the most recent structure, type

```
t2x -c > str.xyz
```

which only extracts the information on $coord.

Visualization of Densities, MOs, Electrostatic Potentials and Fields

Note that the easiest way to visualize orbitals, densities and electrostatic properties is to use the graphical user interface TmoleX.

There are several other possibilities to visualize molecular orbitals or densities.

The molden option in the export menu of proper converts the MO and geometry information to the molden format. Alternatively one can use the conversion program tm2molden, which is interactive and self-explanatory. The generated file can be read in by the molden program (see molden web site).

For larger systems this may become very time-consuming, as plotting data (values on grids) are calculated by the respective programs (e.g. molden). It is more efficient to calculate the data for plots (MO amplitudes, densities, etc.) with TURBOMOLE and to use a visualization tool afterwards, a way, that is described in the following.

Calculation of data on grids to be used for plots with visualization tools (e.g. gOpenMol, available via gopenmol download) can be generated with the options in the grid menu of proper. Alternatively one can use the keyword $pointval in combination with the -proper option. This keyword is obeyed by all TURBOMOLE program that generate densities as dscf, ridft, rimp2 mpgrad, ricc2 (see Section 10.3.3) and egrad. Note, that with -proper and $pointval all of the following quantities may be

calculated simultaneusly, and that for programs `dscf`, `ridft`, `rimp2` and `mpgrad` the density matrix generating steps may be skipped by typing "`<program> -proper`".

Electron densities The option `dens` in the `grid` menu of `proper` or with the above mentioned programs setting of the keyword

```
$pointval dens
```

or simply

```
$pointval
```

results in the calculation of densities

$$\rho(\vec{R}_P) = \sum_{\nu\mu} D_{\nu\mu}\phi_\nu(\vec{R}_P)\phi_\mu(\vec{R}_P) \tag{20.7}$$

on an orthogonal grid of point $\vec{R}_P$. The size of the grid is automatically adjusted to the size of the molecule and the resolution is adjusted to yield acceptable plots (for specification of non-default grid types (planes, lines) and non-default output formats see Section 23.2.28).

The names of the output files are:

`td.plt`	total density (UHF: α density plus β density)
`sd.plt`	spin density (α density minus β density)
`mp2d.plt`	MP2 density
`mp2sd.plt`	MP2 spin density
`ed.plt`	differential density for excited state
`esd.plt`	differential spin density for excited state

`<myname>.plt` general density passed e.g. by the `ricc2` program.

The `.plt` files may be visualized with gOpenMol; the file `coord.xyz`, which is also necessary for gOpenMol, is generated by the above programs, if `$pointval` is set in the `control`-file.

For two-component wavefunctions (only module `ridft` with `$soghf` is set): Total density is on file `td.plt` like for one-component wave functions; this is also true for all other quantities depending only on the density matrix (electrostatic potential etc.). `sd.plt` contains the absolute value of the spin vector density, which is the absolute value of the following vector:

$$s_i(\mathbf{r}) = \begin{pmatrix} \Psi_\alpha^* & \Psi_\beta^* \end{pmatrix} \boldsymbol{\sigma}_i \begin{pmatrix} \Psi_\alpha \\ \Psi_\beta \end{pmatrix} \qquad i = x, y, z$$

```
$pointval fmt=txt
```

leads to a file containing the spin density vectors, which can be used by gOpenMol. It is advisable to choose ca. one Bohr as the distance between two grid points. The options for visualizing two-component wavefunctions are not yet available in the `proper` program.

TDDFT Transition densities The `escf` program can plot the TDDFT transition densities. For the excited state n the transition density is

$$\rho^n(r) = \sqrt{2}\sum_{ia}(X^n + Y^n)_{ia}\phi_i(r)\phi_a(r) \quad (20.8)$$

$$= \sqrt{2}\sum_{ia}\sqrt{\omega_{ia}/\omega_n}Z^n_{ia}\phi_i(r)\phi_a(r) \quad (20.9)$$

where i (a) are occupied (virtual) orbitals, ω_n is the excitation energy, $\omega_{ia} = \epsilon_i - \epsilon_a$, and Z^n contains of the TDDFT vector coefficients (for singlet closed-shell). Note that TDDFT transition density are different from the differential and non-relaxed densities. Transition density can be plotted setting:

```
$pointval transdens
```

By default the first excited state (of the first irrep) will be printed. To print other states add the group

```
$tdlist
 irrep1 num1 col1
 irrep2 num2 col2
 ....
```

where *irrep* is the number of the irrep considered in the $soes data group, *num* is the number of the state, *col* is the column number in the case of degenerate irreps.

The plot are stored in files named:

```
vcao_td_<irrep>_<num>.<col>.plt
```

or with different extension for the different formats.

Available output formats: To get a list of the available output formats of the `proper` program invoke the `format` option with the flag `help`. For the direct generation of graphics output different formats are available for 3D and 2D plots. They depend furthermore on the graphics libraries linked to `proper`. Use the `format` option with the `help` flag for a 2D and a 3D grid to get the list of formats available for the two cases. For the output formats available in combination with `-proper` keyword to export data on grids see Section 23.2.28.

Electrostatic potentials In an analogous way electrostatic potentials can be calculated on grids. The option (`$pointval`) `pot` leads to the calculation of the electrostatic potential of electrons and nuclei (and external constant electric fields and point charges Q if present).

$$V(\vec{R}_P) = -\int \frac{\rho(\vec{r})}{r_{Pr}} \mathrm{d}^3\vec{r} + \sum_A \frac{Z_A}{R_{PA}} + \left(\vec{R}_P\vec{E} + \sum_Q \frac{Q}{R_{PQ}}\right) \tag{20.10}$$

In order to prevent the calculation of singularities at the positions of nuclei, for grid points that are closer to a nucleus than 10^{-6} a.u. the charge of the respective nucleus is omitted in the calculation of the electrostatic potential for these points. The output files are termed `tp.plt`, `sp.plt`, etc.

Electric fields (as derivatives of potentials) are calculated with (`$pointval`) `fld`. The absolute values of electric fields are written to files `tf.plt`, `sf.plt`, etc. For non-default grid types and outputs that allow also for displaying of components of electric fields see Section 23.2.28.

Exchange-correlation potentials (Only for DFT) Computation of the Kohn-Sham exchange-correlation potential on a grid is requested with

```
$pointval xc
```

This functionality is not (yet) available in the `proper` program.

Canonical molecular orbitals. Visualization of molecular orbitals, i.e. generation of `.plt`-files containing amplitudes of MOs i

$$A_i(\vec{R}_P) = \sum_\nu c_{i\nu}\phi_\nu(\vec{R}_P) \tag{20.11}$$

or in the two-component case

$$A_i^\Gamma(\vec{R}_P) = \sum_\nu c_{i\nu}^\Gamma\phi_\nu(\vec{R}_P) \tag{20.12}$$

with Γ as a part of the coefficient matrix (Re(α), Im(α), Re(β), Im(β)), is achieved e.g. by (`$pointval`) `mo 10-12,15`. This triggers the calculation of amplitudes for the MOs/spinors 10-12 and 15 on the grid. The numbering of MOs refers to that you get from the first column of the output of the tool `Eiger`, the one for spinors refers to the file `EIGS`. The filenames contain the type of the irreducible representation (irrep) of the MO, the current number within this irrep and in case of UHF calculations also the spin, e.g. `2a1g_a.plt` contains amplitudes for the second alpha-spin MO of a_{1g} type. For more-dimensional irreps columns are written to separate files, e.g. `1t2g1_a.plt`, `1t2g2_a.plt` and `1t2g3_a.plt` contain the amplitutes of the three columns of the first irrep (alpha spin) of type t_{2g}.

Two-component wavefunctions (only module `ridft` and only if `$soghf` is set): By default only the density of the chosen spinors is written in files named e.g. `10a_d.plt`. Visualization of the amplitudes of the different spinor parts is achieved e.g. by

`$pointval mo 10-12,15 minco` *real*,

where *real* is a plotting threshold that may take values between zero and one. The corresponding part Γ of the spinor (Re(α), Im(α), Re(β), Im(β)) will be written to file, if N^Γ (see below) is larger than that threshold.

$$\begin{aligned} N^\Gamma &= \mathrm{tr}(\mathbf{D}^\Gamma \mathbf{S}) \\ D^\Gamma_{\mu\nu} &= \sum_i c^{\Gamma *}_{i\nu} c^\Gamma_{i\mu} \end{aligned}$$

The filenames consist of the number of the spinor according to file `EIGS` and an additional number for the respective part Γ of the spinor (1 for Re(α), 2 for Im(α), 3 and 4 for the corresponding β-parts) e.g. `10a_4.plt` for the Im(β) of spinor 10.

Localised molecular orbitals If one has generated localized molecular orbitals (LMOs, see above) they can also be visualized.

```
$pointval lmo 3-6,8
```

as an example, leads to calculation of amplitudes for LMOs 3-6 and 8. The coefficients are read from file `lmos` (UHF: `lalp` and `lbet`), the numbering refers to the output from the localization section. For an UHF case the β spin orbitals get an offset of N_{MO}, where N_{MO} is the total number molecular orbitals for each spin case. If one has e.g. 22 orbitals per spin case and is interested in plotting the first 3 β-type LMOs only, one have to type

```
$pointval lmo 23-25
```

Natural molecular orbitals for two-component wavefunctions (only module `ridft` and only if `$soghf` is set): In two-component calculations it is often useful to visualize natural molecular orbitals. In contrast to one-component calculations the occupation numbers are no longer close to zero, one or two, but can take any value between zero and two. Therefor

```
$natural orbitals    file=natural
$natural orbital occupation    file=natural
```

has to be set additionally to `$soghf` (also possible via `define`).
By setting

```
$pointval nmo 9
```

in `control`-file a gOpenMol-compatible file named `nmo_9.plt` is written which can also be visualized with TmoleX.

Natural atomic orbitals If one has generated natural molecular orbitals (NAOs, see above) they can be visualized with the following command in the control file:

```
$pointval nao 7-9,12
```

where the numbers of the NAOs are in the output of the population analysis.

Natural transition orbitals If natural transition orbitals (NTOs) for electronic excitations are available in files named `nto_nocc` and `nto_vir` for, respectively, the occupied and virtual NTOs, plot files for visualizing them can be generated by setting

```
$pointval nto 1-5
```

This will generate plot files for the first five occupied and virtual NTOs. The plot file are named `nto_vir_`n`.plt`, where n is the NTO index.

Non-default grids are described in detail in Sections 23.2.28. In the `proper` program non-default grids can be specified with the `grid` option. Calculation of the above quantities at single points is needed quite often, thus an example is given here.

```
$pointval geo=point
7 5 3
0 0 7
1 2 3
```

calculates densities at points (7,5,3), (0,0,7) and (1,2,3). Output is (x,y,z, density), output file suffix is `.xyz`.

We note in passing that calculation of electrostatic potential at positions of nuclei may be used as an efficient tool to distinguish atoms of similar atomic numbers thus providing a complement to X-Ray Structure Analysis (for details, see ref. [342]).

Electron localization function (ELF) [343] is calculated with (`$pointval`) `elf`, the ELF is defined as

$$\mathrm{ELF}(\vec{r}) = \left(1 + \chi_\sigma(\vec{r})^2\right)^{-1}, \tag{20.13}$$

where the index σ describes the spin and χ is a dimensionless localization index defined with respect to the uniform electron gas,

$$\chi_\sigma(\vec{r}) = D_\sigma(\vec{r})/D^0_\sigma(\vec{r}) \tag{20.14}$$

$$D^0_\sigma(\vec{r}) = \frac{3}{5}(6\pi^2)^{2/3}\rho_\sigma(\vec{r})^{5/3}. \tag{20.15}$$

Possible values of the ELF range from 0 to 1. ELF = 1 corresponds to perfect localization and ELF = 0.5 to electron-gas like pair probability.

Chapter 21

Orbital Dependent Kohn-Sham Density Functional Theory

21.1 Theoretical Background

Approximations to the exchange-correlation (XC) functional of the Kohn-Sham (KS) Density Functional Theory (DFT) can be classified by the so-called "Jacob's ladder." The ground on which the ladder lies is the Hartree approximation (XC energy is zero), and the first rung is the local density approximation (LDA) in which the XC energy density is a simple local function of the density. The second rung of the Jacob's ladder is the generalized gradient approximation (GGA): in this case the XC energy density depends also on the gradient of the density. In the third rung (meta-GGA) an additional variable is used, the Kohn-Sham kinetic energy density which allows, e.g., to construct self-correlation-free functionals. Functionals in the above rungs can have high accuracy for different class of problems in chemistry and solid-state physics, but their main limitation is the self-interaction error (SIE) [344–347]. To avoid the SIE the exchange must be treated exactly and this can be achieved by functionals in the fourth rung which depend explicitly on all the occupied KS orbitals. In the KS formalism the EXX (exact-exchange) energy is (for closed-shell systems $n_s = 2$) [344–347]:

$$E_x^{\mathrm{EXX}} = -\frac{n_s}{2}\sum_a^{occ.}\sum_b^{occ.}\int\int \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}'\frac{\phi_a^{\mathrm{KS}}(\mathbf{r})\phi_b^{\mathrm{KS}}(\mathbf{r})\phi_a^{\mathrm{KS}}(\mathbf{r}')\phi_b^{\mathrm{KS}}(\mathbf{r}')}{\|\mathbf{r}-\mathbf{r}'\|}\,. \tag{21.1}$$

i.e. the same functional form of the Hartree-Fock (HF) exchange but computed with KS orbitals which are obtained using a self-consistent local EXX potential. At this point we should recall that hybrid DFT functionals (including HF exchange), doesn't belong to the KS formalism: in hybrid DFT, in fact, the non-local HF

exchange operator $\hat{v}_{\rm x}^{\rm NL}(\mathbf{r},\mathbf{r}') = -\sum_a^{occ.} \frac{\phi_a(\mathbf{r})\phi_a(\mathbf{r}')}{\|\mathbf{r}-\mathbf{r}'\|}$ is employed in the self-consistent (Generalized Kohn-Sham) equations determining the orbitals.

While LDA, GGA, meta-GGA and hybrid functionals are implemented (for ground-state calculations) in the `dscf` and `ridft`, the `odft` module considers functionals of the fourth rung. Currently exchange-only orbital-dependent approaches are implemented in the `odft` module. The EXX KS local potential ($v_{\rm x}^{\rm EXX}(\mathbf{r})$) can be obtained using the optimized effective potential (OEP) method (in each self-consistent step): [345–348]:

$$\int \mathrm{d}\mathbf{r}' \chi_s(\mathbf{r},\mathbf{r}') v_{\rm x}^{\rm EXX}(\mathbf{r}) \;=\; \sum_a^{occ.}\sum_s^{vir.} 2n_s \left\langle \phi_a \right| \hat{v}_{\rm x}^{\rm NL} \left| \phi_s \right\rangle \frac{\phi_s(\mathbf{r})\phi_a(\mathbf{r})}{\epsilon_a - \epsilon_s} \tag{21.2}$$

where $\chi_s(\mathbf{r},\mathbf{r}') = \sum_a^{occ.} 2n_s \sum_s^{vir.} \frac{\phi_a(\mathbf{r})\phi_s(\mathbf{r})\phi_s(\mathbf{r}')\phi_a(\mathbf{r}')}{\epsilon_a-\epsilon_s}$ is the non-interacting density response.

An effective approximation to the OEP-EXX potential is given by the Localized Hartree–Fock (LHF) potential [344] which is given by

$$\begin{aligned} v_{\rm x}^{\rm LHF}(\mathbf{r}) \;&=\; -\sum_{ij}^{occ.} n_s \frac{\phi_i(\mathbf{r})\phi_j(\mathbf{r})}{\rho(\mathbf{r})} \int \mathrm{d}\mathbf{r}' \frac{\phi_i(\mathbf{r}')\phi_j(\mathbf{r}')}{\|\mathbf{r}-\mathbf{r}'\|} \\ &+\; \sum_{ij}^{occ.} n_s \frac{\phi_i(\mathbf{r})\phi_j(\mathbf{r})}{\rho(\mathbf{r})} \left\langle \phi_i \right| v_{\rm x}^{\rm LHF} - \hat{v}_{\rm x}^{\rm NL} \left| \phi_j \right\rangle \end{aligned} \tag{21.3}$$

where the first term is called *Slater potential* and the second term *correction term*. If terms $i \neq j$ are neglected in the correction term, the Krieger-Li-Iafrate (KLI) potential [349] is obtained. Note that the Eq. (21.3) depends only on occupied orbitals, whereas Eq. (21.2) depends also on virtual orbitals. The LHF total energy is assumed to be the EXX total energy, even if LHF is not variational (although the deviation from the EXX energy is very small, usually below 0.01%). The LHF potential is equivalent to the Common Energy Denominator Approximation (CEDA) [350] and to the Effective Local Potential (ELP) [351].

Both OEP-EXX and LHF (in contrast to functionals of the first three rungs) satisfy the HOMO condition [349]

$$\left\langle \phi_{HOMO} \right| v_{\rm x} \left| \phi_{HOMO} \right\rangle = \left\langle \phi_{HOMO} \right| \hat{v}_{\rm x}^{\rm NL} \left| \phi_{HOMO} \right\rangle \;, \tag{21.4}$$

and the asymptotic relation [352, 353]

$$v_{\rm x}(\mathbf{r}_l) \overset{r_l\to\infty}{\longrightarrow} \left\langle \phi_M \right| v_{\rm x} - \hat{v}_{\rm x}^{\rm NL} \left| \phi_M \right\rangle - \frac{1}{r_l} \,. \tag{21.5}$$

where ϕ_M is the highest occupied orbital which do not have a nodal surface in the asymptotic region along direction $\mathbf{r}_l$. Considering together with condition (21.4), we finally obtain that:

- $v_{\rm x}(\mathbf{r})$ will approach $-1/r$ along all directions where $\phi_{HOMO}(\mathbf{r})$ does not have a nodal surface in the asymptotic region (e.g. this is the case of atoms);

- on directions which belong to the nodal surface of the HOMO, the $v_{\rm x}(\mathbf{r})$ will approach $\langle\phi_M|\, v_{\rm x} - \hat{v}_{\rm x}^{\rm NL}\,|\phi_M\rangle - 1/r$.

Both OEP-EXX and LHF gives total energies very close to the Hartree-Fock one (actually $E_{\rm LHF} > E_{\rm EXX} > E_{\rm HF}$), thus, without an appropriate correlation functional, these methods are not suitable for thermochemistry. On the other hand OEP-EXX and LHF give very good KS orbital spectra. In fact the eigenvalues of the HOMO is very close to the Hartree-Fock and to exact ionization potential (I.P): this is in contrast to functional of the first three rungs which underestimate the HOMO energy by several eVs. In addition a continuum set of bound unoccupied orbitals are obtained. Thus OEP-EXX or LHF KS orbitals are very good input quantities for computing NMR shielding constants [354], energy-levels in hybrid interfaces [355] and TD-DFT excitation energies [356] (the latter using LDA/GGA kernels, not the hybrid ones).

21.2 Implementation

Both the OEP-EXX and LHF methods can be used in spin–restricted closed–shell and spin–unrestricted open–shell ground state calculations. Both OEP-EXX and LHF are parallelized in the OpenMP mode.

21.2.1 OEP-EXX

In the present implementation the OEP-EXX local potential is expanded as [348]:

$$v_{\rm x}^{\rm EXX}(\mathbf{r}) = \sum_p c_p \int \frac{g_p(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}{\rm d}\mathbf{r}' \;, \tag{21.6}$$

where g_p are gaussian functions, representing a new type of auxiliary basis-set (see directory `xbasen`). Inserting Eq. (21.6) into Eq. (21.2) a matrix equation is easily obtained for the coefficient c_p. Actually, not all the coefficients c_p are independent each other as there are other two conditions to be satisfied: the HOMO condition, see Eq. (21.4), and the Charge condition

$$\int \sum_p c_p g_p(\mathbf{r}){\rm d}\mathbf{r} = -1 \;, \tag{21.7}$$

which ensures that $v_{\rm x}^{\rm EXX}(\mathbf{r})$ approaches $-1/r$ in the asymptotic region. Actually Eq. (21.6) violates the condition (21.5) on the HOMO nodal surfaces (such condition cannot be achieve in any simple basis-set expansion).

Note that for the computation of the final KS Hamiltonian, only orbital basis-set matrix elements of $v_{\rm x}^{\rm EXX}$ are required, which can be easily computes as three-index Coulomb integrals. Thus the present OEP-EXX implementation is *grid-free*, like Hartree-Fock, but in contrast to all other XC-functionals.

21.2.2 LHF

In the LHF implementation the exchange potential in Eq. (21.3) is computed on each grid-point and numerically integrated to obtain orbital basis-sets matrix elements. In this case the DFT grid is needed but no auxiliary basis-set is required. The Slater potential can be computed numerically on each grid point (as in Eq. 21.3) or using a basis-set expansion as [344]:

$$v_{\rm x}^{\rm slat}(\mathbf{r}) \;=\; \frac{n_s}{\rho(\mathbf{r})}\sum_{a}^{\rm occ.} \underline{u}_a^T \underline{\chi}(\mathbf{r})\underline{\chi}^T(\mathbf{r})\underline{\underline{S}}^{-1}\underline{\underline{K}}\,\underline{u}_a \;. \tag{21.8}$$

Here, the vector $\underline{\chi}(\mathbf{r})$ contains the basis functions, $\underline{\underline{S}}$ stands for the corresponding overlap matrix, the vector $\underline{u}_a$ collects the coefficients representing orbital a, and the matrix $\underline{\underline{K}}$ represents the non-local exchange operator $\hat{v}_{\rm x}^{\rm NL}$ in the basis set. While the numerical Slater is quite expensive but exact, the basis set method is very fast but its accuracy depends on the completeness of the basis set.

Concerning the correction term, Eq. (21.3) shows that it depends on the exchange potential itself. Thus an iterative procedure is required in each self-consistent step: this is done using the conjugate-gradient method.

Concerning conditions (21.4) and (21.5), both are satisfied in the present implementation. KS occupied orbitals are asymptotically continued [352] on the asymptotic grid point $\mathbf{r}$ according to:

$$\tilde{\phi}_i(\mathbf{r}) = \phi_i(\mathbf{r}_0)\left(\frac{|\mathbf{r}|}{|\mathbf{r}_0|}\right)^{(Q+1)/\beta_i-1} \mathrm{e}^{-\beta_i(|\mathbf{r}|-|\mathbf{r}_0|)} \;, \tag{21.9}$$

where $\mathbf{r}_0$ is the reference point (not in the asymptotic region), $\beta = \sqrt{-2\epsilon_i}$ and Q is the molecular charge. A surface around the molecule is used to defined the points $\mathbf{r}_0$.

21.3 How to Perform

OEP-EXX

To run OEP-EXX calculations select:

```
$dft
 functional oep
```

As the computation of the OEP functional is completely analytic and grid free, any selection of a grid type or size will not influence the OEP calculation in contrast to other density functionals.

Particular care is instead required to orbital and auxiliary basis set. An arbitrary combination of them can lead to very good total energy (i.e. very close to the

Hartree-Fock one) but unphysical OEP potential. In the present release we strongly recommend to use the *d-aug-cc-pVTZ-oep* basis set and the corresponding auxiliary basis set (directory `xbasen`).

The following options can modify the quality, time and output of an OEP calculation. All the options can be set by `define`.

Every option has a reasonable default value so the user does not need to select any of the options below to run a proper OEP calculation.

`$oep` *options*

Listing of all possible options for the flag `$oep`.

`charge vector` *integer*

The Charge condition expansion coefficients in auxiliary basis set representation can be calculated in different kinds.
The selection of *integer* = 1 will use the following ansatz to calculate the coefficients:

$$c_{P_1} = \begin{cases} -\frac{1}{N'_{aux}} \cdot \frac{1}{G_p} & \text{if } G_P \neq 0 \\ 0 & \text{if } G_P = 0 \end{cases} .$$

G_P is the integral over a normalized Gaussian auxiliary basis function. N'_{aux} is the number of auxiliary basis functions with $G_P \neq 0$.
The selection of *integer* = 2 will use the following ansatz to calculate the coefficients:

$$c_{P_2} = \begin{cases} -\frac{1}{\sum\limits_P G_P} & \text{if } G_P \neq 0 \\ 0 & \text{if } G_P = 0 \end{cases} .$$

The variable *integer* must have an integer value. The default value is `2`.

`condition` [*string2*] *string*

In the OEP method two constraints can be applied in the OEP equation. This is the HOMO condition and the Charge condition. The variable *string* can have the values `none`, `HOMO`, `Charge` and `both`. No condition is chosen when `none` is elected. The HOMO condition is chosen when `HOMO` is elected. The Charge condition is chosen when `Charge` is elected. The HOMO condition and the Charge condition are chosen when `both` is elected.
The variable *string2* is optional and only electable if a spin–unrestric-ted calculation is performed. The variable *string2* can have the values `alpha` and `beta`. If *string2* = `alpha` then the condition is defined for the alpha spin channel. If *string2* = `beta` then the condition is defined for the beta spin channel. Both spin channels can have different values.

Example:

```
$oep
condition alpha HOMO
condition beta Charge
```

If only one spin channel is defined the other spin channel uses the same condition automatically. The default value in any case is *string* = `both`.

`core memory` *integer*
: Core memory is the amount of main memory given to the OEP calculation to store the three index integrals calculated during the OEP calculation. The core memory amount is given MB. The calculation runs as fast as possible if all three index integrals can be stored in the core memory. The variable *integer* must have an integer value. The default value is `200`.

`debug`
: Print further information about the OEP calculation especially matrices and vectors used during the OEP calculation. Use this option carefully since a lot of data is written. The default value is `.false.`.

`eigenvalue difference` *integer*
: Two molecular orbitals are considered as degenerated (due to symmetry or incidentally), if the difference between them is smaller then $10^{-integer}$. The variable *integer* must have an integer value. The default value is `6`.

`plot coefficient` *string*
: The expansion coefficients for the auxiliary basis functions which build the local exact exchange potential are written to the file `oepcVx.dat` or in case of a spin–unrestricted calculation to the files `oepcVxa.dat` and `oepcVxb.dat`.
If *string* is `cartesian` the expansion coefficients are given for a Cartesian atomic orbital auxiliary basis, if *string* equals `spheric` the expansion coefficients are given for a spherical atomic orbital auxiliary basis. In any case the expansion coefficients are given for the single atomic orbital auxiliary basis function and contain no information about the symmetry of the system (c1 case). The default value is `cartesian`.

`reference potential`
: Use the reference potential constructed by the applied conditions to the OEP calculation as exchange potential. The solution of the OEP equation is skipped. The default value is `.false.`.

LHF

To run a LHF calculations select:

```
$dft
   functional lhf
   gridsize   3
```

This can be done using `define` (modified grid are not supported) and then run `odft`.

A more suitable procedure is the following:

1) Do a Hartree–Fock calculation using `dscf`.

2) Use the script `lhfprep` to prepare the `control` file (the old `control` file will be saved in `control.hf` and the molecular orbitals in `mos.hf` or in `alpha.hf` and `beta.hf` for the spin-unrestricted case). See `lhfprep -help` for options. Actually LHF can be started from any guessed orbitals, but if HF orbitals are used, a much faster convergence is expected. By default the script `lhfprep` will add/modify the `control` file with:

```
$dft
   functional lhf
   gridtype   6
   gridsize   3
   radsize    3
$lhf
   off-diag on
   num-slater off
   asymptotic dynamic=1.d-3
   conj-grad conv=1.d-6 maxit=20 output=1 asy=1
   slater-dtresh 1.d-9
   slater-region     7.0 0.5 10.0 0.5
   corrct-region             10.0 0.5
$scfdump
$scfiterlimit 30
$scfconv 6
$scfdamp start=0.000 step=0.500 min=0.50
$scforbitalshift noautomatic
$correction matrix-elements file=lhfcg
$correction alpha matrix-elements file=lhfcg_alpha
$correction beta matrix-elements file=lhfcg_beta
```

3) Run `odft`.

With the LHF potential Rydberg series of virtual orbitals can be obtained. To that end, diffuse orbital basis sets have to be used and special grids are required.

`gridtype 4` is the most diffuse with special radial scaling; `gridtype 5` is for very good Rydberg orbitals; `gridtype 6` (default in `Lhfprep`) is the least diffuse, only for the first Rydberg orbitals.

Only gridsize 3–5 can be used, no modified grids.

Use `test-integ` to check if the selected grid is accurate enough for the employed basis-set, see page 447.

The options in the `$lhf` group are:

`off-diag on`
: The LHF exchange potential is computed (default);

`off-diag off`
: The KLI exchange potential is computed (can be selected by `lhfprep -kli`).

`num-slater on`
: the Slater potential is calculated numerically everywhere: this is more accurate but quite expensive. When ECPs are used, turn on this option. It can be selected by `lhfprep -num`.

`num-slater off`
: the Slater potential is computed using basis-sets. This leads to very fast calculations, but accurate results are obtained only for first-row elements or if an uncontracted basis set or a basis set with special additional contractions is used. This is the default.

`asymptotic`
: for `asymptotic` treatment there are three options:

  `asymptotic off`
  : No asymptotic-treatment and no use of the numerical Slater. The total exchange potential is just replaced by $-1/r$ in the asymptotic region. This method is the fastest one but can be used only for the density-matrix convergence or if Rydberg virtual orbitals are of no interest.

  `asymptotic on`
  : Full asymptotic-treatment and use of the numerical Slater in the near asymptotic-region. It can be selected by `lhfprep -asy`.

  `asymptotic dynamic=1.d-3`
  : Automatic switching on (off) to the special asymptotic treatment if the differential density-matrix rms is below (above) 1.d-3. This is the default.

`pot-file save`
: the converged Slater and correction potentials for all grid points are saved in the files `slater.pot` and `corrct.pot`, respectively. Using `pot-file load`, the Slater potential is *not calculated* but read from `slater.pot` (the correction potential is instead recalculated). For spin unrestricted calculations the corresponding files are `slaterA.pot`, `slaterB.pot`, `corrctA.pot` and `correctB.pot`.

`homo`
: allows the user to specify which occupied orbital will not be included in the calculation of correction potential: by default the highest occupied orbital is selected. This option is useful for those systems where the HOMO of the starting orbitals (e.g. EHT, HF) is different from the final LHF HOMO. `homob` is for the beta spin.

`correlation func=`*functional*
: a correlation functional can be added to the LHF potential: use `func=lyp` for LYP, or `func=vwn` for VWN5 correlation.

For other options see 21.3.

21.4 How to plot the exchange potential

It is recommended to check plots of the exchange potential for both OEP-EXX and LHF potential, to avoid spurious numerical oscillations (which usually originates from too small or too large basis-set). To plot the LHF potential over a line, add to the control file (e.g. for a 2000 points along the z axis):

```
$pointval xc geo=line
 grid1 vector 0 0 1 range -10,10 points 2000
 origin 0 0 0
```

and run `odft -proper`. The plotting subroutine reads the file `lhfcg`, containing the matrix elements of the correlation potential is already generated by a previous run. The file `tx.vec` will be generated with four columns (distance, LHF potential, Slater potential, Correction potential).

The procedure to plot the OEP-EXX potential is the same. In this case the expansion coefficients (see Eq. 21.6) are read from the file `oepcVx.dat` (Cartesian format). The file `tx.vec` will be generated with four columns (distance, EXX potential, EXX potential, zero).

21.5 How to quote

- For LHF calculations with `odft`:
  Efficient localized Hartree–Fock methods as effective exact-exchange Kohn–Sham methods for molecules Fabio Della Sala and Andreas Görling J. Chem. Phys. **115**, 5718 (2001) and The asymptotic region of the Kohn–Sham exchange potential in molecules Fabio Della Sala and Andreas Görling J. Chem. Phys. **116**, 5374 (2002)

- For OEP-EXX calculations with `odft`:
  Numerically stable optimized effective potential method with balanced Gaussian basis sets Andreas Heßelmann, Andreas W. Götz, Fabio Della Sala, and Andreas Gŕling J. Chem. Phys. **127**, 054102 (2007).

Chapter 22

Vibronic absorption and emission spectra

22.1 Theoretical Background

22.1.1 Vibronic spectra at zero temperature

The accurate prediction of absorption and emission spectra often requires a quantum mechanical treatment of nuclear vibration [357]. In addition, the geometrical differences between ground and excited state structures, and the differences in vibrational spectra and normal modes lead to mode mixing, making the prediction of the vibrational structure in electronic spectra non-trivial. Under the neglect of anharmonicity, these effects can be described by the Duschinsky rotation [358]

$$\mathbf{Q}_\mathrm{i} = \mathbf{D} + \mathbf{J}\mathbf{Q}_\mathrm{f}\,, \tag{22.1}$$

where $\mathbf{Q}_\mathrm{i}$ and $\mathbf{Q}_\mathrm{f}$ denote initial and final state vibrational coordinates, respectively. $\mathbf{D}$ is the geometric displacement between ground and excited state vibrational structures and $\mathbf{J}$ is the Duschinsky rotation matrix. In case of absorption, the initial state is the electronic ground state and the final state is the electronically excited state; in case of emission, the opposite order applies.

Within the harmonic oscillator approximation, the absorption and emission spectra are given by:

$$\sigma_{abs}(\omega) \quad = \quad \frac{4\pi^2\omega}{3c}|\mu_{if}|^2 \sum_{\mathbf{v}_\mathrm{f}} |\langle\theta_0(\mathbf{Q}_\mathrm{i})|\theta_{\mathbf{v}_\mathrm{f}}(\mathbf{Q}_\mathrm{f})\rangle|^2 \delta(E_{\mathbf{v}_\mathrm{f}} - E_0 - \omega) \tag{22.2}$$

and

$$\sigma_{em}(\omega) = \frac{4\omega^3}{3c^3}|\mu_{if}|^2 \sum_{\mathbf{v}_\mathrm{f}} |\langle\theta_0(\mathbf{Q}_\mathrm{i})|\theta_{\mathbf{v}_\mathrm{f}}(\mathbf{Q}_\mathrm{f})\rangle|^2 \delta(E_0 - E_{\mathbf{v}_\mathrm{f}} - \omega), \tag{22.3}$$

respectively. Here, E_0 denotes the absolute energy of the initial state where all quantum numbers are zero; $E_{\mathbf{v}_\mathrm{f}}$ denotes the final state with quantum numbers $\mathbf{v}_\mathrm{f}$. Both cases assume that absorption and emission occurs from the lowest vibrational level of the initial state (zero temperature approximation). The absorption spectrum (σ_{abs}) is given as the absorption cross section in atomic units (Bohr^2); the emission spectrum (σ_{em}) is given as the emission rate in inverse atomic time units.

Writing the delta function in Eqs. 22.2 and 22.3 as a Fourier transform and applying Mehler's formula [359, 360], the infinite sum in equations 22.2 and 22.3 can be eliminated and written in terms of a *generating function* $G(t)$ in the time domain, which for absorption reads

$$\sum_{\mathbf{v}_\mathrm{f}} |\langle\theta_0(\mathbf{Q}_\mathrm{i})|\theta_{\mathbf{v}_\mathrm{f}}(\mathbf{Q}_\mathrm{f})\rangle|^2 \delta(E_{\mathbf{v}_\mathrm{f}} - E_0 - \omega) = \int_{-\infty}^{\infty} dt \exp\left[-it(\Delta E_{if} - \frac{1}{2}\sum_j \omega_j^f - \omega)\right] G(t), \tag{22.4}$$

with ΔE_{if} being the adiabatic excitation energy. The generating function is given by

$$G(t) = 2^{\frac{N}{2}} \left(\frac{\det(\mathbf{S}^{-1}\mathbf{\Omega}_\mathrm{i}\mathbf{\Omega}_\mathrm{f})}{\det(\mathbf{L})\det(\mathbf{M})}\right)^{\frac{1}{2}} \exp\left(\mathbf{D}^T(\mathbf{\Omega}_\mathrm{f}\mathbf{B}\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{\Omega}_\mathrm{f}\mathbf{B} - \mathbf{\Omega}_\mathrm{f}\mathbf{B})\mathbf{D}\right) \tag{22.5}$$

where $\mathbf{\Omega}_\mathrm{i}$, $\mathbf{\Omega}_\mathrm{f}$, $\mathbf{S}$, $\mathbf{B}$ are diagonal matrices with $(\Omega_\mathrm{i})_{kk} = \omega_k^\mathrm{i}$, $(\Omega_\mathrm{f})_{kk} = \omega_k^\mathrm{f}$, $S_{kk} = \sinh(i\omega_k^\mathrm{f} t)$, and $B_{kk} = \tanh(i\omega_k^\mathrm{f} t/2)$. ^{T} denotes the transpose of the matrix. ω_k^i and ω_k^f denote vibrational frequencies of initial and final state, respectively. Matrices $\mathbf{L}$ and $\mathbf{M}$ are obtained as $\mathbf{M} = \mathbf{J}^T\mathbf{\Omega}_\mathrm{f}\mathbf{B}\mathbf{J} + \mathbf{\Omega}_\mathrm{i}$ and $\mathbf{L} = \mathbf{J}^T\mathbf{\Omega}_\mathrm{f}\mathbf{B}^{-1}\mathbf{J} + \mathbf{\Omega}_\mathrm{i}$. Hence, knowledge of ground and excited state structures and their vibrational spectra allows the construction of all matrices appearing in Eq. 22.5 and $G(t)$ can be propagated in time. In practice, $G(t)$ has to be truncated after a maximum time t_{max}. In addition, $G(t)$ is multiplied by a damping function $\exp(-t/\tau)$ with lifetime τ, which leads to a Lorentzian broadening of the spectral lines. More details can be found in references [357, 361].

22.1.2 Single Vibronic Level Spectra

RADLESS allows the calculation of spectra originating from vibrationally singly excited initial states, i.e. single vibronic level (SVL) emission spectra and Vibrationally Promoted Electronic Resonance (VIPER) spectra [361]. In the frequency domain the SVL emission spectrum for a vibronic initial state $|\theta_k^1\rangle$, where mode k is singly-excited, reads

$$\sigma^1{}_{em,k}(\omega) = \frac{4\omega^3}{3c^3}|\mu_{if}|^2 \sum_{\mathbf{v}_\mathrm{f}} |\langle\theta_k^1|\theta_{\mathbf{v}_\mathrm{f}}\rangle|^2 \delta(\Delta E_{if} + E_{\mathbf{0}_\mathrm{i}} + \omega_k^\mathrm{i} - E_{\mathbf{v}_\mathrm{f}} - \omega). \tag{22.6}$$

Analogously, the absorption spectrum from a singly excited vibrational state reads

$$\sigma^1{}_{abs,k}(\omega) = \frac{4\pi^2\omega}{3c}|\mu_{if}|^2 \sum_{\mathbf{v}_\mathrm{f}} |\langle\theta^1_k|\theta_{\mathbf{v}_\mathrm{f}}\rangle|^2 \delta(\Delta E_{if} - E_{\mathbf{0}_\mathrm{i}} - \omega^\mathrm{i}_k + E_{\mathbf{v}_\mathrm{f}} - \omega). \quad (22.7)$$

Using Mehler's formula and recursive harmonic oscillator recursive relationships, Eq. (22.6) can be formulated in time-domain for emission

$$\sigma^1{}_{em,k}(\omega) = \frac{4\omega^3}{3c^3}|\mu_{if}|^2 \frac{1}{2\pi}\int_{-\infty}^{\infty} dt \exp\left[it\left(\Delta E_{if} + E_{\mathbf{0}_\mathrm{i}} + \omega^\mathrm{i}_k - \omega\right)\right] G^1_k(t), \quad (22.8)$$

with the generating function

$$G^1_k(t) = G(t) \times \omega^i_k \left[2(M^{-1})_{kk} + 4\mathbf{D}^\dagger\mathbf{\Omega}_\mathrm{f}\mathbf{B}\mathbf{J}\mathbf{R}^k\mathbf{J}^\dagger\mathbf{\Omega}_\mathrm{f}\mathbf{B}\mathbf{D} - 2(L^{-1})_{kk}\right], \quad (22.9)$$

with $R^k_{ij} = (M^{-1})_{ik}(M^{-1})_{kj}$. For absorption, the time-domain expression reads

$$\sigma^1{}_{abs,k}(\omega) = \frac{4\pi^2\omega}{3c}|\mu_{if}|^2 \frac{1}{2\pi}\int_{-\infty}^{\infty} dt \exp\left(-it\left(\Delta E_{if} - E_{\mathbf{0}_\mathrm{i}} - \omega^\mathrm{i}_k - \omega\right)\right) G^1_k(t). \quad (22.10)$$

Here, $E_{\mathbf{0}\mathrm{i}}$ denotes the zero point vibrational energy of the initial state. The generating function $G^1_k(t)$ in Eq. (22.10) is identical in structure to Eq. (22.9), but initial state index i refers to the electronic ground state and state index f refers to the electronically excited state.

22.2 Implementation

Any electronic structure method can be used with RADLESS if the necessary input, i.e. structures and vibrational data for the electronic ground and excited state, is provided. After successful calculation of the ground and excited state structures and their vibrational spectra, matrices of Eq. 22.5 are constructed at different times t to obtain the generating function. Fourier transform of the generating function together with the appropriate prefactor gives the absorption or emission spectra. Matrix products are evaluated using basic linear algebra subroutines (BLAS) [362] for real and complex matrices. The spectra are obtained by discrete Fast Fourier transformation [363] of $G(t)$.

22.3 Functionalities of RADLESS

22.3.1 How to use RADLESS

RADLESS allows the calculation of vibronic absorption and emission spectra. Absorption and emission spectra must be computed in separate jobs. Calculations with RADLESS can only be done within C1 symmetry. The following steps are necessary to prepare the required input:

1. Optimize ground state structure and compute vibrational spectrum with AOFORCE or NUMFORCE.

2. Optimize excited state structure and compute vibrational spectrum using NUMFORCE.

3. In a new directory, copy the S1-geometry `coord` and its control file (`control`), which points to the coordinate file `coord`.

4. In the same directory, copy the S0-geometry `coord-gs` and its control file (`control-gs`) pointing to `coord-gs`.

5. Make sure the excited state control file (`control`) points to vibrational modes, frequencies, and reduced masses of the excited state, whereas the ground state control file (`control-gs`) points to vibrational frequencies, normal modes, and reduced masses of the ground state. This can be achieved either by renaming ground state vibrational files (e.g. `vib_normal_modes-gs`, `vibspectrum-gs` etc.) or by directly adding the vibrational data to the `control-gs` file.

6. Add necessary keywords to excited state control file (`control`):

   (a) `$fluorescence` or `$absorption`, to specify what kind of spectrum will be calculated; the two keywords are exclusive.

   (b) `$gsenergy=` *number* ground state total energy (in a.u.) of the excited state structure

   (c) `$esenergy=` *number* excited state total energy (in a.u.) of the excited state structure

   (d) `$spectral width` *integer*; spectral width for fluorescence (for absorption, spectral width is given in `control-gs`) spectrum in atomic units. Default: adiabatic excitation energy of the current system plus four times the inverse lifetime. The radiative rate will be computed by integration from 0 to *integer*. Too high *integer* can lead to artifacts of the radiative rate due to the numerical error of the discrete Fourier transform in the high frequency region. It is recommended to set *integer* by approximately two times the full width at half maximum higher than the adiabatic excitation energy. The fluorescence spectrum will be written from 0 to *integer* onto the file `FLUORESCENCE_spectrum.dat`.

   (e) `$transdip` *number*: magnitude of transition dipole (length representation) in atomic units (obtained from ESCF or EGRAD output). This is only required for fluorescence spectra calculations; for absorption spectra the transition dipole is given in `control-gs`. Default: 1.0 au.

   (f) `$max time` *integer*: total integration time in atomic units optional keywords. Due to the fast Fourier transform, *integer* should be a power of 2.

7. Add optional keywords to excited state control file (`control`):

   (a) `$broadening`: employ lifetime broadening

(b) `$lifetime` *integer*: if broadening is used, *integer* is the lifetime of the damping function given in atomic units

(c) `$dhoa`: Displaced harmonic oscillator approximation is applied. Duschinsky matrix is set to unity (no mode mixing).

(d) `$delta_t` *number*: use an integration time step of *number* (default: 1) in atomic units

(e) `$zerofil` *number*: increase the number of data points in spectrum by adding zeros to the generating function to obtain *number* times the number of data points in time domain; zerofilling is recommended to increase accuracy in peak positions.

(f) `$deuterium`: replace all hydrogen atoms by deuterium

(g) `$vib1_mode` *number*: calculate SVL spectrum of initial state where mode *number* is singly excited; numbering refers to AOFORCE/NUMFORCE output. Depending on the keyword used (`$absorption` /`$emission`), this will either be the SVL absorption or the SVL emission spectrum.

(h) `$vib1_separate`: calculate SVL spectra of all possible singly excited initial states. Depending on the keyword used (`$absorption` /`$emission`), these will either be the SVL absorption or the SVL emission spectra.

(i) `$debug_radless` : print out generating function

(j) `$scaling` *number*: scales excited state frequencies by *number*

8. Add necessary keywords to `control-gs`

(a) `$gsenergy=` *number* ground state total energy of the ground state structure

(b) `$esenergy=` *number* excited state total energy of the ground state structure

(c) `$transdip` *number*: magnitude of transition dipole (length representation) in atomic units (obtained from ESCF or EGRAD output). (This is only required for absorption spectra calculations). Default: 1.0 au.

(d) `$spectral width` *integer*; spectral width for absorption spectrum in atomic units. Default: twice the adiabatic excitation energy of the current system. The total frequency integrated absorption will be computed by integration from 0 to *integer*. The absorption spectrum will be written from 0 to *integer* onto the file `ABSORPTION_spectrum.dat`.

9. Add optional keywords to `control-gs`

(a) `$scaling` *number*: scales ground state frequencies by *number*

Start a RADLESS calculation with `radless > radless.out`.

22.3.2 Output of RADLESS

1. Absorption and emission spectra are written to files, `ABSORPTION_spectrum.dat` and `FLUORESCENCE_spectrum.dat`, respectively. The first column gives the frequency (ω) in atomic units. In case of absorption, the second column gives the absorption cross section in atomic units (Bohr^2). In case of emission, the second column contains the emission rate in inverse atomic time units.

2. If `$vib1_separate` or `$vib1_mode` are specified, SVL absorption or emission spectra are written to files, `ABSORPTION_spectrum_vib1_`*number*`.dat` and `FLUORESCENCE_spectrum_vib1_`*number*`.dat`, respectively. The first column gives the frequency (ω) in atomic units. In case of absorption, the second column gives the absorption cross section in atomic units (Bohr^2).

3. If `$debug_radless` is specified, additional output files are generated. These include `Genfunc_fluor.dat` and `Genfunc_abs.dat`, containing the generating functions for emission and absorption, respectively. Real part is given in the first column, imaginary part is given in the second column.

4. `DUSCHINSKY.dat` contains the Duschinsky rotation matrix

5. Geometric displacements projected on vibrational modes are given in the standard output.

6. Radiative rates are given in the standard output. The static radiative rate refers to $\frac{4\Delta E_{if}^3}{3c^3}|\mu_{if}|^2$, whereas the vibrational radiative rate is the integral over the fluorescence spectrum.

7. The frequency-integrated absorption coefficient [364] ($\int \sigma_{abs}(\omega)d\omega$) is given in the standard output.

Chapter 23

Keywords in the `control` file

23.1 Introduction

The file `control` is the input file for `TURBOMOLE` which directly or by cross references provides the information necessary for all kinds of runs and tasks. `control` is usually generated by `define`, the input generator. This chapter provides a short-hand documentation: a list of the most important key words, the possible parameters for each keyword, default values, and a brief explanation.

23.2 Format of Keywords and Comments

`TURBOMOLE` input is keyword-directed. Keywords start with a `'$'`, e.g. `$title`. Comments may be given after `$dummy`, or by a line starting with `#`; these lines are ignored by `TURBOMOLE`. Blank lines are also ignored. Keywords may be in any order unless stated otherwise below.

The sample inputs given below should help to give an idea how the keywords are to be used. Complete `control` files are provided in Chapter 24. An alphabetical list of all keywords is given in the index.

The `control` file and other input files referenced therein must end with this keyword:

```
$end
```

23.2.1 Keywords for System Specification

General information defining the molecular system: nuclear coordinates, symmetry, basis functions, number of occupied MOs, etc. which are required by every module.

`$title`
give title of run or project here.

`$symmetry d4h`
Schönflies symbol of the point group. `define`, `dscf`, `grad`, `ridft`, `rdgrad`, `odft`, `escf`, `egrad`, `statpt`, `relax`, and `freeh` support all point groups. `mpshift` and `aoforce` support for NMR shielding and force constant calculations etc. all those groups that do not have complex irreps (C_3, C_3h, T, etc). Use a lower symmetry group in this case. For all other programs see the respective chapters above.

`$atoms`
Example with the same basis set on all atoms:

```
$atoms
   basis = def2-SV(P)
```

An example with mixed basis sets and ECPs:

```
$atoms
   basis =def2-SV(P)
cu 1-4
   basis =cu ecp-18 arep
   jbas  =cu ecp-18
   ecp   =cu ecp-18 arep
se 5-6
   basis =se ecp-28 arep dzp
   jbas  =se ecp-28
   ecp   =se arep
```

For each atom group, one can specify

- the basis set
- and the auxiliary (fitting) basis sets (`jbas`, `jkbas`, `cbas`, `cabs`, `xbas`)
- the ECP if this is used,
- the mass (`mass`), and
- the charge (`charge`).

Additionally, the isotopes and the gyromagnetic ratio can be defined, e.g. for ^{3}H with a ratio of 28.534986500,

```
 h  3
     ncisotope=    3
     gyromag=   28.534986500
```

Here, the given isotopes and the gyromagnetic ratio are used for the calculation of NMR coupling constants or EPR hyperfine couplings. By default, we assume the most common isotopes for NMR and EPR spectroscopy.

Attributes listed immediately below `$atoms` are used as defaults which are assigned to all atoms for which these attributes are not overwritten by subsequent assignments to specific atom groups. In this section the basis set name should not be preceeded by an atom symbol. In setting for specific atoms groups the basis set names should be preceeded by the atom symbols. Where no other specifications are made in `$atoms`, the following defaults are used:

- auxiliary basis sets will be assigned automatically if needed based on the name of the orbital basis if available in the basis set library or the `control` file,
- the atomic masses are by default set to isotopic averages,
- the charges are determined from the atomic symbols.
- for basis sets with names starting with `def-`, `def2-`, or `dhf-`, or ending with `-PP` and for the LANL2DZ basis sets the corresponding default ECPs will automatically added for atoms heavier than Kr if no other ECPs have been specified similar as it is done by `define`.

Definitions of (auxiliary) basis sets and ECPs can be provided in files with names specified in the data groups `$basis`, `$jbas`, `$cbas`, `$cabs`, `$jkbas`, `$xbas`, and `$ecp`. If no definitions are provided the respective information will be searched in the basis set libraries and then stored in the files `basis`, `ecp` and `auxbasis`.

`$pople` *char*

This data group specifies the number of Cartesian components of basis functions (i.e. 5d and 7f in AO-Basis, 6d and 10f in CAO-Basis) for which the SCF calculation should be performed. Possible values for *char* are `AO` (default) or `CAO`. This keyword should usually be set in accordance with the chosen basis set. Most basis sets are defined with only the pure spherical components. This is the case for all Karlsruhe basis sets, the correlation consistent basis set families cc-pVXZ etc., and many other basis sets. An exception are the 6-31G basis sets: the polarization functions for them (i.e. for 6-31G*, 6-31G**, and the corresponding basis sets with diffuse functions indicated with an additional + or ++) are defined as full Cartesian set and should be used with `$pople CAO`.

23.2.2 Start guess for molecular orbitals

TURBOMOLE provides a flexible variety of possibilities to generate the start guesses for the molecular orbitals that are needed for the SCF iterations in the Hartree-Fock and DFT codes.

`define` provides possibilities to generate start orbitals using either an Extended Hückel Theory (EHT) guess with several options to specify the orbital occupation. This is the recommended option for difficult situations where simpler automatic start vector generations may fail. It also contains the possibility to generate start MOs from a preceeding calculation that either used a different basis set or a different point group symmetry.

For molecular calculations, the `dscf` and `ridft` programs provide internally three possibilities for generating start MOs, if no MO are found on file:

- EHT guess for start MOs
- start density from superposition of atomic densities
- core hamiltonian guess

The EHT guess build-in in `dscf` and `ridft` is activated with the data group `$eht`:

```
$eht charge=n   unpaired=m  glueck=K
```

n is an integer number which defines the overall charge of the molecular system (default: n=0)

m is the difference between the number of electrons with α spin and the number of electrons with β spin (default: m=0)

K is the global scaling factor for the EHT Hamiltonian matrix: $H_{ij} = KS_{ij}(E_i + E_j)/2$ (default: K=1.7)

For m= 0 the occupation numbers and MOs will be set for a spin-restriced closed-shell calculation and for m$\neq$ 0 a spin-unrestriced open-shell calculation. The occupied orbitals will be selected according to the Aufbau principle using the EHT orbital energies.

As a simple fall-back option, the coefficients for the start MOs can be determined from the eigenvectors of the one-electron core Hamiltonian matrix, i.e. the sum of the kinetic energy energy and the nuclear attraction potential for the electrons. This option is requested for spin-restricted calculations with

```
$scfmo none
```

and for spin-unrestricred calculations with

```
$uhfmo_alpha none
$uhfmo_beta none
```

It requires that prior to the start of the `dscf` or `ridft` program the occupation numbers have been set in the control file.

Alternatively, a start density can be generated from a superposition of spherically averaged atomic densities. This start density is then used to build a Fock matrix which is diagonalized. The coefficients for the start MOs are determined from the eigenvectors of the Fock matrix.

```
$atomdens
  aos=eht
  charge=+2  1-3
  charge=+3  4-13
  charge=-2  14-31
```

aos defines the atomic orbitals that will be used for the generation of the atomic densities. Possible choices are

iao the reference atomic orbitals for the calculation of intrinisic atomic orbitals

eht the EHT basis

basis the orbital basis itself (only meaningful for generally contracted basis sets where the first contracted orbitals are close to atomic orbitals)

Default: aos=eht.

charge=val list sets the charge for the atoms in "list" to "val". Per default all atoms are assumed to be charge-neutral.

As the core Hamiltonian guess, this option requires that prior to the start of the `dscf` or `ridft` program the occupation numbers have been set in the control file. To use its two-component generalization, the keyword `$atoms2c` additionally needs to be set. Then, the start coefficients are determined from the eigenvectors of the two-component Fock matrix including spin–orbit effects.

RHF

`$closed shells`

Specification of MO occupation for RHF, e.g.

```
 a1g     1-4                                    ( 2 )
 a2g     1                                      ( 2 )
```

`$open shells type=1`

MO occupation of open shells and number of open shells. `type=1` here means that there is only a single open shell consisting e.g. of two MOs:

```
b2g     1                                      ( 1 )
b3g     1                                      ( 1 )
```

```
$roothaan          1
a = 1      b = 2
```

`$roothaan`
: Roothaan parameters for the open shell, here a triplet case. `define` recognizes most cases and suggests good Roothaan parameters.

  For further information on ROHF calculations, see the sample input in Section 24.6 and the tables of Roothaan parameters in Section 6.3.

UHF

`$uhf` directs the program to carry out a UHF run,e.g.

```
$alpha shells
 a1g     1-4                                   ( 1 )
 a2g     1                                     ( 1 )
$beta shells
 a1g     1-4                                   ( 1 )
 a2g     1                                     ( 1 )
```

The specification of MO occupation for UHF, `$uhf` overwrites closed-shell occupation specification.

23.2.3 Keyword for the General Memory Specification

Most post-SCF programs (`aoforce`, `ccsdf12`, `egrad`, `escf`, `evib`, `mpgrad`, `mpshift`, `pnoccsd`, `ricc2`, `rirpa`) in `TURBOMOLE` read the data group

`$maxcor 500 mib per_core`

to control the amount of dynamically allocated memory. The integer number (above "500") one can (optionally) give a SI prefix (`kb`, `mb`, `gb`) or an IEC prefix (`kib`, `mib`, `gib`) to specify the unit. If no unit is specified `mib` (binary megabytes, i.e. 1024^2 bytes) are used.

In addition one can specify a reference for parallel calculations:

`per_proc` indicates that the memory is specified per process

`per_node` indicates that the memory is specified per computer node, i.e. shared by all processes of the calculation that run on the machine with the same host name

`per_core` indicates that the memory is specified per core, i.e. per thread

`total` means the specified memory should be divided by all processes and nodes

If not given the specified memory will be used per thread. Note that in release versions before V7.2 the memory was specified per process. For sequential calculations this sub-option has no effect.

If `$maxcor` is not given 500 MiB per thread will be used as limit for dynamically allocated memory.

Note:

- `$maxcor` defines only the memory controled by the electronic structure code. Additional memory can be allocated by the math and MPI libraries linked into the program and by the operating and I/O systems. It is therefore recommended to set `$maxcor` not higher than to 75% – 85% of the physical core memory that is available for the calculation.
- Some programs use additional keywords (`$incore`, `$ricore`, `$ricore_slave`, `$rpacor`) to specify limits for dynamically allocated memory for certain tasks.

23.2.4 Keyword for frozen core approximation

Orbitals for the frozen core approximation in post-HF and post-KS calculations can be specified in the data group `$freeze` in four alternative formats which might be usefull for different types of applications.

In the most explicit format the indices of the frozen orbitals are spefied per irreducible representation:

```
$freeze
 a1g     1-2
 t1u     1
```

In general, these can be occupied orbitals for a frozen core and/or virtual orbitals for anti-core orbitals that should be excluded from calculations of correlation and/or excitation energies.

A more compact definition of the number of frozen occupied and virtuals orbitals is possible with the `implicit` option:

```
$freeze
 implicit core=5 virt=2
```

This will freeze the 5 energetically lowest occupied and 2 highest virtual orbitals (alpha and beta count as one in UHF cases). Note that for degenerate orbitals each degenerate component is counted.

Since version 7.7 two additional options are available to determine the number of frozen core orbitals automatically.

With the format

```
$freeze
  fpc=-3.0  fpv=50.0
```

all orbitals with energies below −3.0 Hartree or above 50.0 Hartree will be frozen. If the option `fpv` is left out, only orbitals below −3.0 Hartree will be frozen. If both options are used that have to be specified on the same line. The values should be chosen such that the energy gaps between frozen and non-frozen orbitals are sufficiently large. This is in particular important for the calculation of reaction energies and potential energy curves or surfaces to ensure that a consistent number of orbitals and, as much as possible, also orbitals of the same shape are frozen for products and educts or all structures.

Alternatively, the option

```
$freeze
   defcore
```

can be used to request a default frozen core. The number of frozen core orbitals will then be determined from the atomic symbols and charges according to the table below. It corresponds approximately, although not strictly to a freezing point of −3 Hartree. Dummy atoms or atoms with charges below 3.2 au will be ignored in when determination of the default frozen core. If atomic charges are modified with `charge` option in the `$atoms` data group, the size of the suggest default core should be checked carefully. Additional information about the number of core orbitals included per atom can be obtained by adding two or more question marks in the line with `$freeze`.

Note that this scheme will likely not work for systems that mix atoms with charges just below the next larger core is used with such that are just above such a value as e.g. Co and Ni. In such cases alternative freezing points can be set for the the option `fpc`.

Limitations:

- Freezing of virtual orbitals is not supported by `mpgrad` and not by the by F12 methods implemented in `ccsdf12`, `ricc2`, and `pnoccsd`.
- The calculation of gradients in `mpgrad` does not support frozen occupied or virtuals orbitals.
- For the limitations regarding frozen orbitals GW calculations see Sec. 14

Default core orbitals:

H 0						He 0
Li 0	Be 2				B-F 2	Ne 2
Na 2	Mg 2				Al-Cl 10	Ar 10
K 10	Ca 10		Sc-Co 10	Ni-Zn 18	Ga-Br 18	Kr 18
Rb 28	Sr 28		Y-Rh 28	Pd-Cd 30	In-I 36	Xe 36
Cs 36	Ba 36	La-Yb 46	Lu-Ir 46	Pt-Hg 62	Tl-At 68	Rn 68
Fr 68	Ra 68	Ac-No 78	Lr 78			

Be–Mg (2) [He] core: $1s^2$

Al–Co (10) [Ne] core: $1s^2 2s^2 2p^6$

Ni–Kr (18) [Ar] core: $1s^2 2s^2 2p^6 3s^2 3p^6$

Rb–Rh (28) [Ni] core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10}$

Pd–Cd (30) [Zn] core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2$

In–Ba (36) [Kr] core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6$

La–Ir (46) [Pd] core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10}$

Pt–Hg (62) [Pd] $4f^{14} 5s^2$ core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{14} 5s^2$

Tl–Ra (68) [Pd] $4f^{14} 5s^2 5p^6$ core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{14} 5s^2 5p^6$

Ac–Lr (78) [Pt] core: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{14} 5s^2 5p^6 5d^{10}$

23.2.5 Other General Keywords

```
$operating system unix
$path
$lock off
$suspend off
```

The four keywords above are set by `define`, but are not necessary.

```
$statistics dscf
```
or
```
$statistics mpgrad
```

Only a statistics run will be performed to determine file space requirements as specified for `dscf` or `mpgrad`. On return the statistics option will be changed to `$statistics off`.

`$actual step dscf`

means *current step*. Keyword and data group (as e.g. `dscf`) is set by every program and removed on successful completion.

`$last step relax`

Keyword and data group (as e.g. `relax`) set by every program on successful completion.

General file cross-references:

```
$coord              file=coord
$intdef             file=coord
$user-defined bonds file=coord
$basis         file=basis
$ecp           file=basis
$jbas          file=auxbasis
$scfmo           file=mos
$uhfmo_alpha     file=alpha
$uhfmo_beta      file=beta
$natural orbitals           file=natural
$natural orbital occupation file=natural
$energy        file=energy
$grad          file=gradient
$forceapprox   file=forceapprox
```

It is convenient not to include all input in the `control` file directly and to refer instead to other files providing the corresponding information. The above cross references are default settings from `define`; you may use other file names. `define` will create most of these files. Examples of these files are given below in the samples.

`$coord` (and `$intdef` and `$userdefined bonds`)
: contains atom specification—type and location—and the bonds and internal coordinates convenient for geometry optimizations.

`$basis`
: specification of basis sets.

`$ecp` specification of effective core potentials.

`$jbas`
: auxiliary (fitting) basis for the Coulomb terms in `ridft`.

`$scfmo`, `$uhfmo_alpha`, `$uhfmo_beta`
: MO vectors of SCF or DFT calculations for RHF or UHF runs.

`$natural orbitals`, `$natural orbital occupation`
: keywords and data groups set by unrestricted `dscf` or `ridft` runs. Contain natural MO vector and orbital occupation.

`$energy`, `$grad`
: energies and gradients of all runs, e.g. for documentation in a geometry optimizations.

`$forceapprox`
: approximate force constant for geometry optimizations.

23.2.6 Keywords for redundant internal coordinates in `$redund_inp`

With the parameters in `$redund_inp` the generation of redundant internal coordinates can be modified. All entries have to be made in the `control` file before invoking the `ired` option. Important options are:

`iprint` n
: print parameter for debug output: The larger n is, the more output is printed $n \geq 0, n \leq 5$ (default: 0)

`metric` n
: method for generating and processing of redundant internal coordinates $n \geq -3, n \leq 3, n \neq 0$ (default: 3)

  Values for the `metric` option:

  `n = 1` "Delocalized Coordinates"
  The $\mathbf{BmB^t}$ matrix is diagonalized for the complete set of redundant internal coordinates, matrix $\mathbf{m}$ is a unit matrix.

  `n = -3` Delocalized Coordinates obtained with a modified matrix $\mathbf{m}$, the values of $\mathbf{m}$ can be defined by user input (see below).

  `n = -1` "Hybrid Coordinates"
  Natural internal coordinates are defined as in the old `iaut` option. If a cage remains, delocalized coordinates (as for n=1) are defined for the cage.

  `n = -2` Very similar to the $n = 1$ option, but for the remaining cage delocalized coordinates with modified matrix $\mathbf{m}$ are defined as for $n = -3$.

  `n = 2` "Decoupled coordinates"
  The redundant coordinates are divided into a sequence of blocks. These are expected to have decreasing average force constants, i.e. **stretches, angle coordinates, torsions and "weak" coordinates**.

The $\mathbf{BB^t}$ matrix is diagonalized for each block separately after the columns of $\mathbf{B}$ were orthogonalized against the columns of $\mathbf{B}$ of the the preceding blocks.

`n = 3` "Generalized natural coordinates"
Natural internal coordinates are defined first, for the remaining cage decoupled coordinates are defined.

`type` *r*
a positive real number, which is an approximate "force constant", can be read in for each type of coordinate (see below). The force constants are used for the definition of the matrix $\mathbf{m}$ in $\mathbf{BmB^t}$.

Types of internal coordinates for the definition of m

The matrix $\mathbf{m}$ is assumed to be a diagonal matrix. For each type of coordinate a different value for the force constants m_{ii} can be read in. Types of coordinates are:

`stre` bond stretch (default: 0.5)

`invr` inverse bond stretch (default: 0.5)

`bend` bond angle (default: 0.2)

`outp` Out of plane angle (default: 0.2)

`tors` dihedral or "torsional" angle (default: 0.2)

`linc` Special angle coordinate for collinear chains, bending of the chain a–b–c in the plane of b–c–d (default: 0.2)

`linp` bending of the chain a–b–c perpendicular to the plane of b–c–d (default: 0.2)

`wstr` stretch of a "weak" bond, i.e. the bond is assumed to have a very low force constant, e.g. a "hydrogen bond" or a "van der Waals bond" (default: 0.05)

`winv` inverse stretch of a weak bond (default: 0.05)

`wbnd` bond angle involving at least one weak bond (default: 0.02)

`wout` Out of plane angle for weak bonds (default: 0.02)

`wtor` dihedral angle for weak bonds (default: 0.02)

`wlnc` linc coordinate for weak bonds (default: 0.02)

`wlnp` linp coordinate for weak bonds (default: 0.02)

23.2.7 Keywords for Module uff

One has to specify only the Cartesian coordinates (data group $coord) to start a uff run. The program uff requires the data groups $uff, $ufftopology, $uffgradient and $uffhessian. If these keywords do not exist in the control file the program will generate these data groups.

The data group $uff contains the parameters described below. The default values—in the control file—are:

```
      1         1          0 ! maxcycle,modus,nqeq
 111111                     ! iterm
0.10D-07  0.10D-04          ! econv,gconv
    0.00  1.10              ! qtot,dfac
0.10D+03  0.10D-04    0.30  ! epssteep,epssearch,dqmax
      25      0.10    0.00  ! mxls,dhls,ahls
    1.00      0.00    0.00  ! alpha,beta,gamma
       F         F       F  ! transform,lnumhess,lmd
```

The explanation of the variables are as follows:

maxcycle
: number of max. optimization cycles (maxcycle=1: single-point calculation).

modus
: can have the values +1 or -1. If modus = -1 only the topology will be calculated.

nqeq each nqeq cycle the partial charges will be calculated. If nqeq = 0, then the partial charges are calculated only in the first cycle, if the file ufftopology does not exist.

iterm
: switch for the different types of force field terms:

100000	bond terms will be calculated.
010000	angle terms will be calculated.
001000	torsion terms will be calculated.
000100	inversion terms will be calculated.
000010	non bonded van der Waals terms will be calculated.
000001	non bonded electrostatic terms will be calculated.

econv, gconv
: convergence criteria for energy and gradient.

qtot total charge of the molecule.

`dfac` distance parameter to calculate the topology. If the distance between the atoms I and J is less than the sum of the covalent radii of the the atoms multiplied with `dfac`, then there is a bond between I and J.

`epssteep`
if the norm of the gradient is greater than `epssteep`, a deepest-descent-step will be done.

`epssearch`
if the norm of the gradient is smaller than `epssearch`, no line-search step will be done after the Newton step.

`dqmax`
max. displacement in a.u. for a coordinate in a relax step.

`mxls`, `dhls`, `ahls`
parameters of linesearch:

`ahls`	start value
`dhls`	increment
`mxls`	number of energy calculations

`alpha`, `beta`, `gamma`
modification parameter for the eigenvalues of the Hessian (see below): $f(x) = x * (\texttt{alpha} + \texttt{beta} * \exp(-\texttt{gamma} * x))$.

`transform`
a switch for the transformation in the principal axis system.

`lnumhess`
a switch for the numerical Hessian.

`lmd` a switch for a MD calculation.

Input Data Blocks Needed by UFF

`$coord`
cartesian coordinates of the atoms (default: `$coord file=coord`)

`$ufftopology`
contains a list of the next neighbours of each atom (see Section 23.2.7). Sometimes it is useful to enter the connectivity (in the input block `nxtnei12` in the file `ufftopology`) by hand (not always necessary; default: `$ufftopology file=ufftopology`).

Beyond this `uff` reads the force field parameters for the atoms from the file `parms.in`. If this file exists in the directory from which one starts an `uff` calculation the program will use this file, if not the program reads the data from the file `$TURBODIR/uff/parms.in`.

If one wants own atom types, one has to add these atoms types in the file `parms.in`. For each new atom type one has to specify the *natural* bond distance, the *natural* bond angle, the *natural* non-bond distance, the well depth of the Lennard-Jones potential, the scaling factor ζ, the effective charge, torsional barriers invoking a pair of sp^3 atoms, torsional barriers involving a pair of sp^2 atoms, generalized Mulliken–Pauling electronegativities, the idem potentials, characteristic atomic size, lower bound of the partial charge, upper bound of the partial charge. Distances, energies and charges are in atomic units and angles are in rad.

UFF Output Data Blocks

`$coord`
: contains the (updated) Cartesian coordinates of the atoms (default: `$coord file=coord`).

`$ufftopology`
: contains the full information of the topology of the molecule and the whole force field terms (see below; default: `$ufftopology file=ufftopology`).

`$uffgradient`
: contains the accumulated Cartesian analytical gradients (default: `$uffgradient file=uffgradient`).

`$uffhessian`
: contains the Cartesian analytical Hessian;
(default: `$uffhessian file=uffhessian0-0`).

The file `ufftopology`

The topology file `ufftopology` contains the blocks nxtnei12, nxtenei13, nxtnei14, connectivity, angle, torsion, inversion, nonbond and qpartial. It starts with `$ufftopology` and ends with `$end`. The first three blocks (nxtnei12, nxtnei13, nxtnei14) have the same form: they start with the atom number and the number of its neighbours, in the next line are the numbers of the neighbour atoms. Then the *connectivity*-block follows starting with the number of bond terms. Each line contains one bond term:

$$I \quad J \quad \text{d} \quad \text{BO}.$$

Here are I and J the number of the atoms, d the distance in a.u. and BO is the bond order.

The angle terms follow, starting with the number of the angle terms. In each line is one angle term:

$$J \quad I \quad K \quad \text{wtyp} \quad \theta \quad nr_{JI} \quad nr_{IK}.$$

Here are J, I and K the atoms number, where atom I is in the apex. "wtyp" is the angle type and has the following values:

wtyp = 1 linear case

wtyp = 2 trigonal planar case

wtyp = 3 quadratic planar case

wtyp = 6 octahedral case

wtyp = 9 all other cases.

θ is the angle value in degree. nr_{JI} and nr_{IK} are the number of the bonds between J and I and the bond between I and K. The hybridization of atom I determines "wtyp".

Then the torsion terms follow, starting with the number of the torsion terms. Each line contains one torsion term:

$$I \quad J \quad K \quad L \quad nr_{JK} \quad \text{ttyp} \quad \phi \quad \theta_{IJK} \quad \theta_{JKL}.$$

Here are I, J, K and L the atom numbers. nr_{JK} is the number of the bond between J and K. "ttyp" is the torsion type:

ttyp = 1 J (sp^3)–K (sp^3)

ttyp = 11 like ttyp=1, but one or both atoms are in Group 16

ttyp = 2 J (sp^2)–K (sp^3) or vice versa

ttyp = 21 like ttyp=2, but one or both atoms are in Group 16

ttyp = 22 like ttyp=2, but J or K is next a sp^2 atom

ttyp = 3 J (sp^2)–K (sp^2)

ttyp = 9 all other cases.

ϕ is the value of the torsion angle in degree. θ_{IJK} is the angle value of $(I - J - K)$ and θ_{JKL} is the cwone for $J - K - L$. The hybridizations of J and K determine "ttyp".

The inversion terms follow starting with the number of inversion terms (e.g. the pyramidalisation of NH_3). In each line is one inversion term:

$$I \quad J \quad K \quad L \quad \text{ityp1} \quad \text{ityp2} \quad \text{ityp3} \quad \omega_1 \quad \omega_2 \quad \omega_3.$$

I, J, K and L are the atom numbers. Atom I is the central one. ityp1, ityp2, ityp3 are the types of the inversions:

ityp = 10 atom I is C and atom L is O

ityp = 11 like ityp=10, but L is any atom

ityp = 2 I is P

ityp = 3 I is As

ityp = 4 I is Sb

ityp = 5 I is Bi

ityp = 9 all other cases.

ω_1, ω_2 and ω_3 are the values of the inversion angles in degree.

The nonbond terms follow starting with the number of the non-bonded terms. In each line is one nonbond term:

$$I \quad J \quad \mathrm{d}\ .$$

Here I and J are the atom numbers, d the distance in a.u. Then the partial charges follow.

If the determination of the molecule connectivity failed, you can specify the block nxtnei12 in the file `ufftopology`. Then the program calculates the other blocks.

Based on the numbers of the next neighbours (block nxtnei12 in the file `ufftopology`) the program tries to determine the UFF type of an atom. The following rules are implemented: If the atom has three next neighbours and it is in the nitrogen group, then it has a hybridization three. If it is not in the nitrogen group, it has hybridization two. If the atom has four next neighbours and it is in the carbon group, it has hybridization three. If it is not in the carbon group, it becomes hybridization four. If the number of next neighbours is six, then it gets the hybridization six.

Since the smallest eigenvalues λ_i of the Hessian has the greatest influence on the convergence of the geometry optimization, one can shift these values with

$$\tilde{\lambda}_i = \lambda_i \cdot \left(\alpha + \beta \cdot e^{-\gamma x}\right)$$

and calculates a new Hessian with these modified eigenvalues.

23.2.8 Keywords for `tb`

Module tb which performs GFN-xTB or GFN2-xTB extended tight binding calculations reads in the keyword `$tb`

```
$tb
  charge <number>
  gfn <method-number>
  accuracy <number>
  etemp <number>
  broydamp <number>
  maxiter <number>
```

The options are:

`charge -1`
: sets the molecular total charge. ...,-2, -1, 0, 1, 2,... If not specified, a neutral input is assumed: `charge 0`

`gfn 2`
: choose method, GFN-xTB (1) or GFN2-xTB (2), default is 2

`accuracy 1.0`
: accuracy for SCC calculations, lower value means higher accuracy. Default is 1.0

`etemp 300`
: electronic temperature in Kelvin, default is 300K

`broydamp 0.4`
: Broyden damping for charge mixing, default is 0.4

`maxiter 250`
: maximum number of SCC iterations, default is 250

23.2.9 Keywords for `woelfling`

Module WOELFLING reads options from data group

```
$woelfling
```

The below values of the options are default values with the following meaning:

```
 ninter           14
```

Number of interpolated structures for optimization.

```
 ncoord            2
```

Number of input structures provided by user.

```
 align             0
```

Align input structures by translation/rotation 0=yes, 1=no.

```
 maxit          40
```

Maximum number of iterations.

```
 dlst   3.00000000000000
```

Threshold for accuracy of LST-interpolation.

```
thr  1.000000000000000E-004
```

Threshold for mean of norms of projected gradients.

```
method q
```

Use standard optimization from initial LST-path (method q) or grow reaction path (method qg).

Furthermore,

```
riter 0
```

counts the number of completed iteration (no option).

23.2.10 Keywords for Modules `dscf` and `ridft`

`$denconv` *real*

Convergency criterion for the root mean square of the density matrix. If you want to calculate an analytical MP2 gradient (program `mpgrad`) *real* should be 1.d-7 or less.

`$dft` *options*

Listing of all possible sub-keywords for `$dft` (cross-references are given).

The user normally has to choose only the functional and the grid size, see below. All other parameters have proven defaults.

`functional` *name*

Specification of the functional, default is BP86, printed as `functional b-p`. For all possible—and useful—functionals, please refer to page 443 and for definition of the functionals the section 6.2 on page 155.

Example (default input):

```
$dft
   functional b-p
```

`gridsize` *integer* or m*integer*

Specification of the spherical grid (see section 23.2.10 on page 443). Default is `gridsize m3`.

Example:

```
$dft
   gridsize m3
```

`gridtype` *integer* —not recommended for use—

Specification of the mapping of the radial grid.

Possible values for `gridtype` are $1, \ldots, 6$, but `gridtype 4` to `6` is only for the use with `functional lhf` (see page 447). For the definition of `gridtype 1` – `3`, please refer to Eq. (16), (17) and, (19) in Ref. [365].

Example (default value):

```
$dft
   gridtype 3
```

`debug` *integer*

Flag for debugging. `debug 0` means no debug output (default), `debug 1` means some output, `debug 2` means a lot more output. Be careful!

`nkk` *integer*

Specification of the sharpness of the partition function as proposed by Becke [366], default is `nkk 3`. The usage of `nkk` makes sense only in the range $1 \leq$ `nkk` ≤ 6.

Example:

```
$dft
   nkk 3
```

`ntheta` *integer*
`nphi` *integer* —not recommended for use—

Only for user-specified Lobatto grids, i.e. `gridsize 9`, `ntheta` specifies the number of θ points and `nphi` specifies the number of ϕ points. For the fixed Lobatto grid, i.e. `gridtype 8`, the default value is `ntheta 25` and `nphi 48`.

When `gridsize 9` is given, you have to specify both, `ntheta` and `nphi` (see below), otherwise the program will crash. The restriction for user-defined Lobatto grids is the number of grid points, which must not exceed 2000 grid points.

Example:

```
$dft
  gridsize 9
  ntheta 30
  nphi 60
```

`old_RbCs_xi`

Original grids had not been carefully optimized for all atoms individually. This has now been done, which let to changes of ξ for Rb and Cs only resulting in minor improvements. If you have ongoing projects, which have been started with the old grids, you should continue using them with the keyword `old_RbCs_xi`.

Example:

```
$dft
   old_RbCs_xi
```

`radsize` *integer*

Specifies the number of radial grid points. Default values depend on type of atom and grid (see keyword `gridsize`). The formula for the radial gridsize is given as,

$$\text{number of radial grid points} = \text{ioffrad} + (\texttt{radsize} - 1) * 5\,.$$

ioffrad is atom-dependent, the more shells of electrons, the larger ioffrad:

elements	ioffrad	elements	ioffrad
for H–He	20	for K–Kr	40
for Li–Ne	25	for Rb–Xe	45
for Na–Ar	30	for Cs–Lw	50

The grids for relativistic calculations, i.e. gridsize 3a, use the atom number Z in a modified formula given as,

$$\text{number of radial grid points} = 20 + (\texttt{radsize} - 1) * 5 + Z\,.$$

The radial grid size increases further for finer grids:

`gridsize`	1	2	3	4	5	6	7	8	9
`radsize`	1	2	3	6	8	10	14	9	3

If you want to converge results with respect to radial grid size, increase `radsize` in steps of 5, which is convenient (see equation above).

`diffuse` *integer*

Serves to increase quadrature grids; this is recommended in case of very diffuse wavefunctions. With the keyword `diffuse` grids are modified by changing the linear scaling factor ξ of radial grid points and the radial gridsize:

`radsize` $\Longrightarrow$ `radsize` + *incr*
$\xi \Longrightarrow \xi$ * *scal*

`diffuse` *integer*	1	2	3	4	5	6
incr	1	2	3	4	5	6
scal	1.5	2.0	2.8	4.0	5.0	6.0

For information about the linear scaling parameter ξ, see Eq. (16)–(19) and Table 1 in Ref. [365].

In addition, the reduction of spherical grid points near nuclei is suppressed, i.e. `fullshell on` is set (see page 431).

Note: the keyword `radsize` *integer* overrules the setting of *incr*; for more information, see p. 429.

Recommendation: For diffuse cases use `gridsize m4` (or larger) in combination with `diffuse 2` and check the number of electrons; for more difficult cases use `diffuse 4`. In case of doubt, verify the calculation with a larger grid, i.e. `gridsize 7`.

The test suite example `$TURBODIR/TURBOTEST/dscf/short/H3PO4.DSCF.DIFFUSE` provides an example of usage; this also gives reasonable values for damping and orbitalshift to reach convergence in this and similar cases, see `$scfdamp` and `$scforbitalshift` (p. 436 and p. 439).

Example (Recommendation):

```
$dft
  gridsize m4
  diffuse 2
```

`rhostart` *integer*
`rhostop` *integer* —for developers only—

Radial grid points have a linear scaling parameter ξ, see Eq.(16)–(19) and Table 1 in Ref. [365]. With the following input,

```
$dft
   rhostart 50
   rhostop 200
```

one performs a numerical integration for the density and the exchange correlation term for $\xi = 0.5, (0.01), 2.0$ for given MOs and functional. NOTE: only molecules with a single atom type can be used. The results serve to establish stable, optimal ξ values, see Figure 1 in Ref. [365]. Program stops after this testing.

`reference`

Usage of the reference grid, which is a very fine grid with very tight thresholds. The default values for the different variables are:

```
gridsize 7
radsize 14
fullshell 1
dgrenze 16
fgrenze 16
qgrenze 16
fcut 16
```

Please refer to the corresponding sub-keywords for explanation.

If you want to use the reference grid, you have to skip the keyword `gridsize`, and type instead `reference`. Example:

```
$dft
   functional b-p
   reference
```

Note: The sub-keywords `radsize`, `fcut`, `dgrenze`, `fgrenze`, `qgrenze` can be used to overwrite the above values, but not vice versa.

`test-integ`
Check if the selected grid is accurate enough for the employed basis-set by performing a numerical integration of the norm of all (occupied and virtual) orbitals. Useful for LHF. 447.

`batchsize` *integer*
Grid points are sorted into batches, which are then processed. This increases efficency. This should be changed only by developers. Default is `batchsize 100`.

`fullshell`
Standard grids have reduced number of spherical grid points near nuclei. With the keyword `fullshell` this reduction is suppressed. Reference grid (see keyword `reference`) always has full spherical grids with 1202 points. Should be used to checked the influence of spherical grid reduction.
Example for the usage of `fullshell`:

```
$dft
   functional b-p
   gridsize m4
   fullshell
```

`symblock1` *real*
`symblock2` *real* —for developers only—
Values of real effects efficiency of the quadrature, default is `symblock1 0.001` and `symblock2 0.001`, one can try higher or smaller values.

`xparameter` *integer* —not recommended for use—
Where `xparameter` (default) can be: `sgrenze` (8), `fgrenze` (10), `qgrenze` (12), `dgrenze` (12) and, `fcut` (14). These parameters control neglect of near zeros of various quantities. With `xparameter` *integer* one changes the default. *integer* larger than defaults will increase the numerical accuracy. Tighter threshold are set automatically with keyword `$scfconv` (see section 23.2.10 on page 436).

`weight derivatives`
Includes the derivatives of quadrature weights to get more accurate results. Default is that the derivatives of quadrature weights will be not considered, see section 23.2.15 on page 470.

`gridordering`
Grid points are ordered into batches of neighbouring points. This increases efficiency, since now zeros can be skipped for entire batches.

`gridordering` is default for serial version, not for the parallel one. You cannot use `weight derivatives` and `gridordering` together.
Example for switching off `gridordering`:

```
$dft
   gridordering 0
```

`s-junc` *integer*
`p-junc` *integer*

Within the semi-numerical integration scheme employed for local hybrid functionals, S- and P-junctions feature pre-screenings of the analytical two-center integrals with respect to shell pair overlap and shell pair conjunction through the density matrix, respectively. This allows the neglection of certain shell pairs and thus accelerates the calculation. `s-junc` and `p-junc` set the corresponding thresholds to $10^{-integer}$. Smaller values increase the number of neglected shell pairs and thus reduce the calculation cost, but also decrease the accuracy of the results. Note that for gradient and scf calculations the P-junctions also depend on the grid weights. If larger grids are employed seeking more accurate results, the threshold for P-junctions should thus be adjusted (increasing the value for `p-junc 6`). The defaults settings are `s-junc 6` and `p-junc 6` for `grad`, `rdgrad`, and `ridft`. Too small values for `s-junc` and `p-junc` might result in non-converging calculations or deteriorated results. This holds particularly for excited state calculations. For `escf` the default values are therefore `s-junc 8` and `p-junc 8`.

`$electrostatic field`

Specification of external electrostatic field(s). The specification may take place either by `Ex, Ey, Ez` or by `x, y, z, |E|`. See also `$fldopt`.

Example:

```
$electrostatic field
       0.1000E-03   0.000       0.000
```

Electrostatic fields are also possible for 2c calculations, turning on `geofield` is mandatory in these calculations. `$pcc` activates picture change correction for fields integrals in 1c and 2c (local) BSS/DKH/X2C calculations.

`$fermi tmstrt=<300.0> tmend=<100.0> tmfac=<0.9> hlcrt=<1.0E-01> stop=<1.0E-03> nue=<N>`

Requests calculation of occupation numbers at a finite temperature T. For an orbital with the energy ε_i the occupation number $n_i \in [0,1]$ is calculated as

$$n_i = \frac{1}{2}\mathrm{erfc}\left(\frac{\varepsilon_i - \mu}{fT}\right),$$

where μ is the Fermi level. The factor $f = 4k/\sqrt{\pi}$ is chosen to yield the same slope at the Fermi level as the Fermi distribution.

Calculation of the fractional occupation numbers starts when the current HOMO-LUMO gap drops below the value given by `hlcrit` (default: 0.1). The initial temperature given by `tmstrt` (default: 300 K) is reduced at each SCF cycle by the factor `tmfac` (default: 1.0) until it reaches the value `tmend` (default: 300 K). Note that the default values lead to occupation numbers calculated at a constant $T = 300$ K. Current occupation numbers are frozen if the energy change drops below the value given by `stop` (default: $1{\cdot}10^{-3}$). This prevents oscillations at the end of the SCF procedure.

Calculation of fractional occupation numbers often requires much higher damping and orbital shifting. Please adjust the values for `$scfdamp` and `$scforbitalshift` if you encounter convergence problems.

In UHF runs this option can be used to automatically locate the lowest spin state. In order to obtain integer occupation numbers `tmend` should be set to relatively low value, e.g. 50 K.

Calculation of fractional occupation numbers should be used only for single point calculations. When used during structure optimizations it may lead to energy oscillations.

The optional `nue` value (number of unpaired electrons) can be used to force a certain multiplicity in case of an unrestricted calculation. `nue=0` is for singlet, `nue=1` for dublet, etc.

The option `addTS` adds the entropic contribution of the smearing to the total energy which is important for very high temperatures only.

Finally the option `noerf` will use the full correct Fermi statistics rather than the term given above.

`$firstorder`

Perform first-order SCF-calculation, i.e. perform only one SCF-iteration with the start MOs (which should be the orthogonalized MOs of two independent subsystems as is explained in detail in Chapter 20).

`$fldopt` *options*

Specification of options related with external electrostatic fields. The following options are available:

`1st derivative on/off`

Calculate numerical 1st derivative of SCF energy with respect to electrostatic field (default: off), increment for numerical differentiation is `edelt` (see below).

`2nd derivative on/off`

Calculate numerical 2nd derivative of SCF energy with respect to electrostatic field (default: off), increment for numerical differentiation is edelt.

`edelt=` *real*

Increment for numerical differentiation (default: 0.005).

`fields on/off`
: Calculate SCF energy for non-zero external electrostatic fields defined in `$electrostatic field`.

`geofield on/off`
: Calculate SCF energy for one external field definition and dump disturbed MOs onto `$scfmo`. This enables to evaluate properties or perform geometry optimizations in the presence of an external field.

Caution: don't use the RI approximation for all these calculations since this will lead to non-negligible errors!!

`$incore` *integer*
: By using this option the two-electron integrals are kept in RAM; *integer* specifies how many megabytes should be allocated. If the integrals exceed the RAM allocated the program reverts to the standard mode. Supports all methods which process two-electron integrals, i.e. SCF and DFT (including hybrid functionals); RHF and UHF.

  The following condition must be met:

  `$scfdenapproxl 1`

  and rhfshells 1 or 2. It is advisable to set `$thize` as small as possible (e.g. `$thize 0.1d-08`) and to remove the keyword `$scfdump`.

  Note: this keyword does not work for parallel runs.

`$intsdebug sao`
: Output of one-electron matrices expressed in symmetry-adapted AO basis. Can be used in C1 symmetry or with symmetry. In case of symmetry, the matrices are printed for each irreducible representation. Note that `$intsdebug cao` is not supported anymore.

  Note that the output gives one triangle of the one-electron matrices. Thus the entries are:

  (11) (21) (22)
  (31) (32) (33)
  (41) (42) (43)
  ...

  In C1, the order of the basis functions is such that all s-orbitals are given first, then all p-orbitals, all d-orbitals and so on. So we have:

  1. atom 1s,2s,3s...
  2. atom 1s,2s,3s...

  ...

  Note that the SAO basis uses 5 d-functions and CAO uses 6.

`$mo-diagram` *only* `nirreps=`*integer*

If this keyword is set the energies and symmetry labels of all occupied MOs will be dumped to this data group. This may be helpful to draw mo-diagrams. If *only* has been set only the start MOs are dumped and the program quits.

`nirreps` will hold the total number of displayed orbitals after the successful run.

`$mom`

This keyword enables the use of the maximum overlap method in unrestricted `ridft` calculations to access excited states self-consistently. [367]

`$moprint`

If this keyword is present all occupied orbitals are dumped to standard output. Be careful about this option as it can create huge output files in case of many basis functions.

`$mo output format` *format*

If this line is present, the dscf program is forced to output the MOs using the new FORTRAN format *format* regardless of the *format*-option in data group `$scfmo`. Otherwise the input format will be used.

Example: `$mo output format(3(2x,d15.8))`

`$natural orbitals`

This data group will be written after an UHF calculation (together with $natural orbital occupation) and contains the natural space orbitals (same syntax as `$scfmo`).

`$natural orbital occupation`

This data group will be written after an UHF calculation (together with `$natural orbitals`) and contains the occupation of natural orbitals (syntax as any data group related with orbital occupation information, e.g. `$closed shells`), e.g.

```
a       1-5                    (     2.00000000000000 )
a       6                      (     1.99949836999366 )
a       7                      (     1.99687490286069 )
a       8                      (     1.00000000000000 )
a       9                      (      .00312509713931 )
a       10                     (      .00050163000634 )
```

`$prediag`

concerns the first SCF iteration cycle if start MOs from an EHT guess are used.

The SCF iteration procedure requires control mechanisms to ensure (fast) convergence, in `TURBOMOLE` these are based on orbital energies ϵ_i of the preceeding iteration used for level shifting and damping (besides DIIS, see below). This

feature cannot be used in the first iteration if EHT MOs are employed as start, since ϵ_i are not available. The keyword `$prediag` provides 'ϵ_i of the zeroth iteration' by diagonalization of occ–occ and virt–virt part of the first Fock matrix, to allow for level shifting etc.. See `$scfdiis` below.

`$restart dscf` *twoint*

Try a `dscf` restart. The program will read the data group `$restartd` (which must exist, also `$scfmo` has to exist!) and continue the calculation at the point where it ended before. If the additional option `twoint` is appended, the program will read the two-electron integrals from the files specified in `$scfintunit`, so there will be almost no loss of cpu-time.

All this information is normally provided by the previous `dscf` run if the keyword `$scfdump` (see there) was given.

`$restartd` *data*

Data provided by a previous dscf run that has been interrupted. This keyword is created when `$scfdump` was given.

`$rundimensions` *data*

is set by define so usually no changes are necessary. The dimensions must be greater or equal to those actually required, i.e. you can delete basis functions and keep rundimensions. This keyword is not necessary for small cases.
Example:

```
dim(fock,dens)=6072
natoms=6
nshell=34
nbf(CAO)=108
nbf(AO)=98
dim(trafo[SAO<-->AO/CAO])=256
rhfshells=1
```

`$scfconv` *integer*

SCF convergency criterion will be $10^{-integer}$ for the energy. Gradients will only be evaluated if $integer > 6$.

`$scfdamp start=<.500> step=<.050> min=<.100>`

Damping parameters for SCF iterations in order to reduce oscillations. The old Fock-operator is added to the current one with weight 0.5 as *start*; if convergence is good, this weight is then reduced by the *step* 0.05 in each successive iteration until the *min*imum of 0.1 is reached. (These are the default settings of define for closed-shell RHF). DSCF automatically tries to adjust the weight to optimize convergence but in difficult cases it is recommended to start with a large weight, e.g. 1.5, and to set the minimum to a larger value, e.g. 0.5.

`$scfdebug` *options*

Flags for debugging purposes. Following options are available:

`vectors` *integer*
: Output level concerning molecular orbitals. *integer*=0 (default) means minimal output, >1 will output all start MOs and all MOs in each iteration.

`density` *integer*
: Output level concerning difference density matrices.

`debug` *integer*
: *integer* > 0 will dump a lot of information—be careful!

`$scfdenapproxl` *integer*
: Direct SCF procedures build the Fock matrix by exploiting information from previous iterations for better efficiency. With this keyword information from the last *integer* iterations will be used. This feature is switched on with the default value 20, even if the keyword is absent. The user may reduce the number of iterations employed to smaller values (e. g. 10) in cases were numerical stability could become an issue. With the value 0 this feature is switched off; the Fock matrix is constructed from scratch in each iteration.

`$scfdiis` *options*
: Control block for convergence acceleration via Pulay's DIIS *.
Options are:

`errvec=`*char*specifies the kind of error vector to be used (two different kind of DIIS algorithms)

char=`'FDS' or 'SDF' or 'FDS-SDF'`
: uses (FDS − SDF) as error vector.

char= `none`
: no DIIS

char= `sFDs`
: use $S^{-1/2}FDS^{1/2}$ − transposed

Further suboptions:

`maxiter=`*integer*
: maximum number of iterations used for extrapolation.

`debug=`*integer*
: debug level (default: 0)

`integer=1` print applied DIIS coefficients

`integer=2` print DIIS matrix and eigenvalues, too

`qscal=`*real*
: scaling factor in DIIS procedure: `qscal` > 1 implies some damping, `qscal` = 1.0: straight DIIS.

`thrd=`*real*
: directs the reduction of qscal to `qscal` = 1.0 (no damping in DIIS), done if $||errvec|| <$ `thrd`.

*P.Pulay; Chem. Phys. Lett., **73**, 393 (1980), P.Pulay; J. Comput. Chem., **4**, 556 (1982)

Defaults for `$prediag` (see above) and `$scfdiis`

`errvec=FDS-SDF`, `maxiter=5`, `qscal=1.2`, `thrd=0.0`, this implies DIIS damping in all iterations, prediag is switched of.

Recommended:

`errvec=sFDs` leads to the following defaults:
`qscal=1.2`, for SCF runs: `maxiter=6` and `thrd=0.3`, prediag is off; for DFT runs: `maxiter=5` and `thrd=0.1` prediag is on. If you want to switch off prediag put
`$prediag none`.

`$scfdump`

Dump SCF restart information onto data group `$restartd` and dump SCF MOs in each iteration onto `$scfmo` (scfdump = iter). Additionally, a data block `$scfiterinfo` will be dumped containing accumulated SCF total-, one- and two-electron energies of all previous SCF iterations. Information that will allow you to perform a restart if your calculation aborts will be dumped on data group `$restartd` (see also `$restart`).

`$scfintunit` *options*

Disc space specification for two-electron integrals. The following suboptions are available (and necessary):

`unit=`*integer*
: A legacy suboption which is ignored.

`size=`*integer*
: Filespace in megabytes for this file. `size=0` leads to a fully direct run. `size` is set by a statistics run, see `$statistics`. DSCF switches to direct mode if the file space is exhausted.

`file=`*char*
: Filename. This may also be a complete path name, if you want to store the integrals in a special directory. Make sure the file is local, otherwise integrals are transmitted over the network.

Thus your data group `$scfintunit` may look like this:

```
$scfintunit
 unit=30       size=35       file=twoint1
 unit=31       size=35       file=/users/work/twoint2
```

Maximal 30 files may be specified in this way.

`$scfiterlimit` *integer*

Maximum number of SCF iterations (default: 30).

`$scfmo` *none* `file=`*char*

Input/output data group for SCF MOs. You can specify

`none`
: To perform a calculation without a start vector (i.e. use a core Hamiltonian guess).

`file=`*char*
: The file where the MOs are written on output (default: mos).

These two options can also be used for `$uhfmo_alpha` and `$uhfmo_beta` to use a core guess and write the molecular orbitals to `file`.

After running `define` or a `TURBOMOLE` calculation additional options may appear specifying the origin of the MOs:

`expanded`
: These MOs were obtained by projection form another basis set. They should not be used for wavefunction analysis.

`scfconv=`*integer*
: The MOs are converged SCF MOs , the convergence criterion applied was $10^{-integer}$

`scfdump=`*integer*
: The MOs are unconverged SCF MOs which were written on this data group after iteration *integer*. The latter three options are mutually exclusive.

`format`*(format string)*
: This specifies the FORTRAN format specification which was used for MO output. The standard format is (4d20.14). (See data group `$mo output format`.)

Example:
Your data group `$scfmo` could look like this after a successful `TURBOMOLE` run :

```
$scfmo  scfconv=7  format(3(1x,d19.13))
1  a1   eigenvalue=-.524127   nsao=6
.1234567890123d+01 -.1234567890123d+00  .1234567890123d-01
.1234567890123d+01 -.1234567890123d+00
3  a2   eigenvalue=-.234810
...
```

`$scforbitalorder on/off`
: Order SCF MOs with respect to their energies (default: on)

`$scforbitalshift` *options*
: To assist convergence, either the energies of unoccupied MOs can be shifted to higher energies or, in open-shell cases, the energies of closed-shell MOs to lower energies. In general a large shift may help to get better convergence.

Options are:

`noautomatic`
: Automatic virtual shell shift switched off.

`automatic` *real*
: Automatic virtual shell shift switched on; the energies of virtual orbitals will be shifted if the HOMO-LUMO gap drops below *real* such that a gap of *real* is sustained. This is the default setting if the keyword is missing with *real*=0.1.

`closedshell=`*real*
: Option for open-shell cases. Closed shells are shifted to lower energies by *real*. The default shift value is `closedshell=0.4`.
Note: Normally this will disable the automatic shift of energies of virtual orbitals! To override this, you should append an exclamation mark to the 'automatic' switch, i.e. specify '`automatic!` *real*'.

`individual`
: Set shifts for special occupied MOs. To use this option, start the line with the symmetry label and the list of MOs within this symmetry and append the desired shift in brackets as in the following example:

```
a1  1,2,4-6  (-.34)
b1  8        (+.3)
```

`$scftol` *real*
: Integral evaluation threshold. Integrals smaller than *real* will not be evaluated. Note that this threshold may affect accuracy and the convergence properties if it is chosen too large. If `$scftol` is absent, a default value will be taken obtained from `$scfconv` by $real = \frac{10^{-(scfconv+1)}}{3 \cdot \#bf}$ (#bf = number of basis functions).

`$scratch files`
: The scratch files allocated by `dscf` can be placed anywhere in your file systems instead of the working directory by referencing their path names in this data group. All possible scratch files are listed in the following example:

```
$scratch files
    dscf    dens          path1/file1
    dscf    fock          path2/file2
    dscf    dfock         path3/file3
    dscf    ddens         path4/file4
    dscf    statistics    path7/file7
    dscf    errvec        path8/file8
    dscf    oldfock       path9/file9
    dscf    oneint        path10/file10
```

The first column specifies the program type (`dscf` stands for SCF energy calculations, i. e. the `dscf` program), the second column the scratch file needed

by this program and the third column the path name of the file to be used as scratch file.

`$statistics` *options*

The following options are allowed

`off`	Do not perform integrals statistics
`dscf`	Perform integrals statistics for `dscf`
`mpgrad`	see `mpgrad`
`dscf parallel`	see PARALLEL PROCESSING

Options `dscf parallel, grad, mpgrad` will be described in the related chapters.

If `$statistics dscf` has been given integral prescreening will be performed (which is an n^4-step and may therefore be time-consuming) and a table of the number of stored integrals as a function of the two parameters `$thize` and `$thime` will be dumped. Afterwards, the files pace needed for the current combination of `$thize` and `$thime` will be written to the data group (`$scfintunit`) and `$statistics dscf` will be replaced by `$statistics off`.

`$thime` *integer*

Integral storage parameter, which is related to the time needed to calculate the integral. The larger *integer* the less integrals will be stored. The default value is *integer* = 5. (see also `$thize`, `$statistics`)

`$thize` *real*

Integral storage parameter, that determines, together with `$thime`, the number of integrals stored on disc. Only integrals larger than *real* will be stored. The default value is *real* = 0.100E-04.

RHF/ROHF

`$closed shells`

Specification of MO occupation for RHF, e.g.

```
 a1g     1-4                                    ( 2 )
 a2g     1                                      ( 2 )
```

`$open shells type=1`

MO occupation of open shells and number of open shells. 'type=1' here means that there is only a single open shell consisting e.g. of two MOs:

```
b2g     1                                     ( 1 )
b3g     1                                     ( 1 )
```

`$rohf`

This data group is necessary for ROHF calculations with more than one open shell. Example:

```
$rohf          1
    a -a     a=0  b=0
    h -h     a=1  b=2
    a -h     a=1  b=2
```

This example is for the 7S state of chromium ($3d^5\,4s^1$) in symmetry group I. Note that for this option being activated, `$roothaan` also has to be specified in your `control` file, although its parameter has no meaning in this case. For more details see Section 6.3.

`$roothaan`

For ROHF-calculations with only one open shell the Roothaan parameters[†] a and b have to be specified within this data group (see also `$rohf`). Example:

```
$roothaan
    a = 3/4       b = 3/2
```

This example is for the 3P ground state of carbon ($2p^2$) in symmetry group I. `define` recognizes most cases and suggests good Roothaan parameters.

For further information on ROHF calculations (e.g. with more than one open shell), see the sample input in Section 24.6 and the tables of Roothaan parameters in Section 6.3.

Note that this keyword toggles the ROHF mode also for more than one open shell. If it is not given, the open-shell electrons are simply ignored.

UHF

`$alpha shells` and `$beta shells`

these two data groups specify the occupation of alpha and beta spin UHF MOs (syntax as any data group related with orbital occupation information, e.g. `$closed shells`)
Example:

```
$alpha shells
a      1-8                                    ( 1 )
b      1-2                                    ( 1 )
$beta shells
a      1-7                                    ( 1 )
b      1-3                                    ( 1 )
```

[†] C. C. J. Roothaan: Rev. Mod. Phys. **32** (1960) 179.

`$uhf`
: directs the program to carry out a UHF run. `$uhf` overwrites closed-shell occupation specification.

`$spin constraint {real}`
: The presence of `$uhf` activates the UHF mode. An additional constraint can be imposed for the $(\text{spin})^2$ exceptation value by using `$spin constraint {real}`. A non-zero *real* value fixes the $(\text{spin})^2$ expectation value by using [82]
$F^{\alpha}(\text{SAO}) - 2 * \tau * SD^{\beta}S$
$F^{\beta}(\text{SAO}) - 2 * \tau * SD^{\alpha}S$
instead of F^{α}, F^{β}; in the limiting case (*real* $\rightarrow$ INFINITY) the ROHF $(\text{spin})^2$ expectation value would result. Default spin constraint 0, this is also used if the key is not set. We suggest to increase the constraint in the sequence 0.01, 0.1, 1.0, and 10.0 to check the impact on the $(\text{spin})^2$ expectation value.

`$uhfmo_alpha` and `$uhfmo_beta`
: These two data groups contain the UHF MO vectors for alpha and beta spin respectively (same syntax as `$scfmo`).

`$uhfmo_beta`
: see `$uhfmo_alpha`

DFT

```
$dft
 functional b-p
 gridsize m3
```

for DFT calculations one has to specify the functional and the grid (for the quadrature of the exchange correlation part). The settings above are default: both lines can be left out if the B-P86 functional and grid m3 are required. Other useful functionals supported are:

`b-lyp`

`b3-lyp`

`b3-lyp_Gaussian` (equivalent to the Gaussian98 keyword B3LYP with VWNIII)

`bh-lyp`

`s-vwn`

`s-vwn_Gaussian` (equivalent to the Gaussian98 keyword SVWN with VWNIII)

`tpss`

`tpssh`

Possible grids are 1–7 and m3–m5 where grid 1 is coarse (least accurate) and 5 most dense. The so-called multiple grids m3–m5 use the following ansatz: SCF iterations

with grid 1–3, final energy and gradient with grid 3–5. Usually m3 is fine: for large or delicate systems, try m4. For general response properties and parallel calculations, the multiple grids are not recommended, see also Sec. 8. Multiple grids should not be used for magnetic properties. In relativistic all-electron calculations grids with an increased number of radial points are recommended. These can be selected by appending the character a after the gridsize, e.g. `gridsize 4a`. For a reference calculation with a very fine grid and very tight thresholds use 'reference' as grid specification instead of 'gridsize xy'.

Note: the functionals b3-lyp_Gaussian and s-vwn_Gaussian are made available only for comparability with Gaussian. The functional VWNIII is much less well founded than VWN5 and the `TURBOMOLE` team does not recommend the use of VWNIII.

RI

`Dscf` does not run with the keyword $rij: you must call the RI modules `Ridft` and `Rdgrad` for energy and gradient calculations. However, it does run with the keyword $rik, but it will ignore all RI settings and do a conventional non-RI Hartree–Fock or DFT calculation.

`$rij`
: Enforces an RI-J calculation if module `ridft` is used, can be used for Hartree-Fock as well as for DFT calculations with pure or hybrid functionals.

`$ridft`
: Obsolete keyword - use $rij instead!

`$rik`
: Enforces a RI-JK calculation if module `ridft` is used, can be used for Hartree-Fock as well as for DFT calculations with pure or hybrid functionals.

`$ricore` *integer*
: Choose the memory core available (in megabyte) for special arrays in the RI calculation (the less memory you give the more integrals are treated directly, i.e. recomputed on the fly in every iteration)

`$jbas file=`*auxbasis*
: Cross reference for the file specifying the auxiliary basis as referenced in `$atoms`. We strongly recommend using auxbasis sets optimized for the respective MO basis sets, e.g. use SVP (or TZVP) for the basis and the corresponding auxbasis as provided by `define` (default: `file=auxbasis`).

`$ripop`
: Calculation of atomic charges according to the s partial wave and atomic dipole moments according to the p partial wave as resulting from the auxbasis representation of the density

RI-JK

If the keyword `$rik` is found in the `control` file, `ridft` performs a Hartree–Fock–SCF calculation using the RI-approximation for both Coulomb and HF-exchange (efficient for large basis sets). For this purpose needed (apart from `$ricore`):

`$jkbas file=auxbasis`
: Cross reference for the file specifying the JK-auxiliary basis as referenced in `$atoms`. This group is created by the `rijk` menu in `define`.

MARI-*J*

Multipole-Accelerated-Resolution-of-Identity-J. This method partitions the Coulomb interactions in the near- and far-field parts. The calculation of the far-field part is performed by application of the multipole expansions and the near-field part is evaluated employing the RI-J approximation. It speeds up calculation of the Coulomb term for large systems. It can only be used with the `ridft` module and requires setting of the `$ridft` keyword.

```
$marij
  precision   1.0D-06
  lmaxmom     10
  nbinmax     8
  wsindex     0.0
  extmax      20.0
  thrmom      1.0D-18
```

The following options are available:

`precision` specifies precision parameter for the multipole expansions. Low-precision MARI-J calculations require $1 \cdot 10^{-6}$, which is the default. For higher precision calculations it should be set to $1 \cdot 10^{-8}$–$1 \cdot 10^{-9}$.

`lmaxmom` maximum l-moment of multipole expansions. It should be set to a value equal at least twice the maximum angular momentum of basis functions. Default value is 10 and it should probably never be set higher than 18.

`thrmom` Threshold for moment summation. For highly accurate calculations it should be set to $1 \cdot 10^{-24}$.

`nbinmax` number of bins per atom for partitioning of electron densities. Default value is 8 and hardly ever needs to be changed.

`wsindex` minimum separation between bins. Only bins separated more than the sum of their extents plus `wsindex` are considered as far-field. Default is 0.0 and should be changed only by the experts.

extmax	maximum extent for charge distributions of partitioned densities. Extents with values larger then this are set to extmax. Hardly ever needs to be changed.

Seminumeric HF-Exchange

If the keyword $senex is found in the control file, ridft performs a Hartree–Fock–SCF calculation using the seminumerical approximation for HF-exchange. Standard dft-grids can be used for the numerical integration. Smaller grids (-1,0) and the corresponding m-grids (m1,m2) have been defined as well. For high precision energies the use of de-aliasing is recommended (do_sfit, C1 symmetry only) which will yield reliable energies with the m1 grid in nearly all cases. Alternatively grids of at least size m3 are recommended for heavy atoms. The gridsize can be modified just like in dft-calculations. We introduced three different keywords to finetune where you want to employ the seminumerical exchange approximations. The keyword $senex activates seminumerical calculations in energies (ridft), excitation energies (escf), vibrational frequencies (aoforce) and chemical shifts (mpshift). The keyword $esenex activates seminumerical exchange in escf, aoforce, mpshift, egrad(excitation energy part only) but not in ridft. Further it resets the default grid to -1 in these modules; which was found to be sufficient in most cases. The keyword $dsenex activates seminumerical gradient calculations in ground (rdgrad) and excited states (egrad; excitation energy AND gradient part). An example using the default grid and de-aliasing for SCF (m1) and grid m2 for gradients looks like this:

```
$senex
 do_sfit
$dsenex
 gridsize m2
```

Seminumeric Coulomb+Exchange: The pseudospectral approach

The modules escf, aoforceand egradcan make use of the pseudospectral approximation to calculate Coulomb contribution seminumerically on a grid. To do so simply add the keyword $pseudospectral additionally if $senex or $esenex are present. This is especially valuable when no RI-Fitting basis is available for the chosen basis set (e.g. "cbas" for Sapporo-type basis sets, ANO-type basis sets, x2c-TZVPPall etc.) as the pseudospectral approach does not need any auxiliar/fitting basis sets at all. Excitation energies, harmonic frequencies and ZPEs obtained from the pseudospectral approximation are extremely accurate (usually considerably lower errors than RI) and still orders of magnitude faster than using classic Coulomb integrals. For ground states (e.g. ridft) the pseudospectral approximation is not yet available. As the RI approximation provides true upper bounds in this case it will outperform the pseudospectral approximation. Further auxiliary basis sets of type "jbas" are,

unlike "cbas" auxiliary basis sets, widely available for the whole periodic table. In C1-symmetry we recommend the use of de-aliasing using the `do_sfit` keyword. An example for applying the de-aliased pseudospectral approximation looks like this:

```
$senex
 do_sfit
$pseudospectral
```

LHF

Use the Localized Hartree–Fock (LHF) method to obtain an effective Exact-Exchange Kohn–Sham potential (module `dscf`). The LHF method is a serial implementation for spin-restricted closed-shell and spin-unrestricted ground states.

```
$dft
 functional lhf
 gridsize 6
```

With the LHF potential Rydberg series of virtual orbitals can be obtained. To that end, diffuse orbital basis sets have to be used and special grids are required.

`gridtype 4` is the most diffuse with special radial scaling; `gridtype 5` is for very good Rydberg orbitals; `gridtype 6` (default in `Lhfprep`) is the least diffuse, only for the first Rydberg orbitals.

Only gridsize 3–5 can be used, no multiple grids.

Use `test-integ` to check if the selected grid is accurate enough for the employed basis-set.

How to do LHF runs

1) Do a Hartree–Fock calculation using `dscf`.
2) Use the script `lhfprep` to prepare the `control` file (the old `control` file will be saved in `control.hf` and the molecular orbitals in `mos.hf` or in `alpha.hf` and `beta.hf` for the spin-unrestricted case). See `lhfprep -help` for options.
3) Run again `dscf`.

Otherwise the LHF functional can be selected in `define`: in this case default options are used.

Options for the LHF potential can be specified as follows (see also `lhfprep -help`)

```
$lhf
   off-diag    on
```

```
numerical-slater off
pot-file save
asymptotic dynamic=1.d-3
homo     1b1u
homob    1b1u    # ONLY UNRESTRICTED
conj-grad conv=1.d-7 maxit=20 output=1 cgasy=1
slater-dtresh    1.d-9
slater-region     7.0 0.5 10.0 0.5
corrct-region             10.0 0.5
slater-b-region   7.0 0.5 10.0 0.5 # ONLY UNRESTRICTED
corrct-b-region           10.0 0.5 # ONLY UNRESTRICTED
correlation func=lyp
```

`off-diag off`
: calculation of the KLI exchange potential. By default the LHF exchange potential is computed (`off-diag on`).

`numerical-slater on`
: the Slater potential is calculated numerically everywhere: this is more accurate but much more expensive. When ECP are used, turn on this option.

`numerical-slater off`
: leads to accurate results only for first-row elements or if an uncontracted basis set or a basis set with special additional contractions is used: in other cases `numerical-slater on` has to be used (this is default).

`asymptotic`
: for `asymptotic` treatment there are three options:

  `asymptotic off`
  : No asymptotic-treatment and no use of the numerical Slater. The total exchange potential is just replaced by $-1/r$ in the asymptotic region. This method is the fastest one but can be used only for the density-matrix convergence or if Rydberg virtual orbitals are of no interest.

  `asymptotic on`
  : Full asymptotic-treatment and use of the numerical Slater in the near asymptotic-region.

  `asymptotic dynamic=1.d-3`
  : Automatic switching on (off) to the special asymptotic treatment if the differential density-matrix rms is below (above) 1.d-3. This is the default.

`pot-file save`
: the converged Slater and correction potentials for all grid points are saved in the files `slater.pot` and `corrct.pot`, respectively. Using `pot-file load`,

the Slater potential is *not calculated* but read from `slater.pot` (the correction potential is instead recalculated). For spin unrestricted calculations the corresponding files are `slaterA.pot`, `slaterB.pot`, `corrctA.pot` and `correctB.pot`.

`homo`
allows the user to specify which occupied orbital will not be included in the calculation of correction potential: by default the highest occupied orbital is selected. This option is useful for those systems where the HOMO of the starting orbitals (e.g. EHT, HF) is different from the final LHF HOMO. `homob` is for the beta spin.

`correlation func=`*functional*
a correlation functional can be added to the LHF potential: use `func=lyp` for LYP, or `func=vwn` for VWN5 correlation.

For expert users
Options for the conjugate-gradient algorithm for the computation of the correction potential: rms-convergence (`conj-grad conv=1.d-7`), maximum number of iteration (`maxit=20`), output level `output=0-3`, asymptotic continuation in each iteration (`cgasy=1`).

With `slater-dtresh= 1.d-9` (default) the calculations of the numerical integrals for the Slater potential is performed only if it changes more than 1.d-9.

Asymptotic regions specification:

`corrct-region` R_F Δ_F
$0 \ldots R_F - \Delta_F$: basis-set correction potential
$R_F - \Delta_F \ldots R_F + \Delta_F$: smooth region
$R_F + \Delta_F \ldots + \infty$: asymptotic correction
Defaults: $R_F = 10$, $\Delta_F = 0.5$

`slater-region` R_N Δ_N R'_F Δ'_F
$0 \ldots R_N - \Delta_N$: basis-set Slater potential
$R_N - \Delta_N \ldots R_N + \Delta_N$: smoothing region
$R_N + \Delta_N \ldots R'_F - \Delta'_F$: numerical Slater
$R'_F - \Delta'_F \ldots R'_F + \Delta'_F$: smoothing region
$R'_F + \Delta'_F \ldots + \infty$: asymptotic Slater
Note: $R'_F - \Delta'_F \leq R_F - \Delta_F$
Defaults: $R_N = 7$, $\Delta_N = 0.5$, $R'_F = 10$, $\Delta'_F = 0.5$
Use `correct-b-region` and `slater-b-region` for the beta spin.

Two-component SCF (GHF)

Self-consistent two-component calculations (e.g. for spin-orbit interactions) can be carried out using the module `ridft` . The following keywords are valid:

`$soghf`
enforces two-component-SCF calculations; this option is compatible with `$rij`, `$rik` and `$dft`.

`$kramers`
switches on Kramers-restricted formalism

`$collinear`
switches on collinear two-component formalism (not rotational invariant)

`$gdiis`
enforces DIIS for complex Fock operator.

All-electron relativistic approaches (X2C, BSS, DKH)

Relativistic *all-electron* calculations can be done employing the X2C, the BSS or the DKH Hamiltonian. Implemented for modules `dscf` and `ridft`. Note that gradients are only available with X2C in the modules `grad` and `rdgrad`.

`$rx2c`
switches on X2C calculation.

`$rbss`
switches on BSS calculation.

`$rdkh` *integer*
switches on DKH calculation of order *integer*.

`$dkhparam` *integer*
selects parameterization of the DKH Hamiltonian. Valid values are 1 (=default), 2, 3, 4, and 5.

`$dkhparam 1`: Optimum parametrization (OPT)

`$dkhparam 2`: Exponential parametrization (EXP)

`$dkhparam 3`: Square-root parametrization (SQR)

`$dkhparam 4`: McWeeny parametrization (MCW)

`$dkhparam 5`: Cayley parametrization (CAY)

Note in particular that the parametrization does not affect the Hamiltonian up to 4th order. Therefore, as long as you run calculations with DKH Hamiltonians below 5th order you may use any symbol for the parametrization as they would all yield the same results. Higher-order DKH Hamiltonians depend slightly on the chosen paramterization of the unitary transformations applied in order to decouple the Dirac Hamiltonian, but this effect can be neglected. For details on the different parametrizations of the unitary transformations see [368].

`$rlocal`
switches on local approach. Default is DLU (see `$rlocpara`).

`$rlocpara` *integer*
selects parameterization of the local approximation. Valid values are 0 or 1. Default is the DLU (0). 1 refers to the DLH approximation. For details on the different parametrizations see [142]. Gradients are restricted to the DLU approximation.

`$finnuc`
switches on finite nucleus model based on a Gaussian charge distribution with parameters taken from [369]. In case of magnetic properties, the vector potential is also based on finite nuclei.

`$snso`
selects screened-nuclear-spin-orbit (SNSO) approach for the spin-dependent integrals in two-component calculations.

`$snsopara` *integer*
selects parametrization of SNSO. Valid values are 0 and 1. 0 refers to the original set of parameters originally used in low-order DKH theory [145], while 1 refers to the modified parameters [146, 147] for exact decoupling methods. Default is 1.

`$pcc`
switches on relativistic picture-change correction for the calculation of expectation values.

All of these keywords are compatible with `$soghf` employing the RI-J approximation.

23.2.11 Keywords for Point Charge Embedding

`$point_charges`
Specification of position and magnitude of point charges to be included in the Hamiltonian. Each point charge is defined in the format

```
<x> <y> <z> <q>
```

with `<x>`, `<y>`, `<z>` being the coordinates and `<q>` its charge,e.g.

```
$point_charges thr=<real> selfenergy nocheck list
   2. 2. 2. 5.
   5. 0. 0. 2.5
```

In addition the following optional arguments may be given:

`thr=`*real*
: distance threshold for discarding redundant point charges, default value 10^{-6}.

`selfenergy`
: if given, the selfenergy of the point charge array will will be included in the energy and the gradient

`nocheck`
: switches off the check for redundant point charges and the default symmetrization. This option can significantly speed up the point charge treatment if *many* of them are involved - use only if the point charges are well distributed and symmetry is C_1, e.g. when they stem from proper MM runs

`list`
: print all point charges in the output (default is to print the point charges only if less than 100 charges given)

23.2.12 Keywords for Periodic Electrostatic Embedded Cluster Method

The Periodic Electrostatic Embedded Cluster Method (PEECM) functionality provides electronic embedding of a finite, quantum mechanical cluster in a periodic, infinite array of point charges. It is implemented within HF and DFT energy and gradient `TURBOMOLE` modules: `dscf`, `grad`, `ridft`, `rdgrad`, and `escf`. Unlike embedding within a finite set of point charges the PEEC method always yields the correct electrostatic (Madelung) potential independent of the electrostatic moments of the point charges field. It is also significantly faster than the traditional finite point charges embedding.

The basic PEECM settings are defined in the `$embed` block. It can be redirected to an external file using `$embed file=<file_name>`.

Following keywords are used for the PEECM calculation setup:

`periodic`

Specifies the number of periodic directions. Allowed values for `number` are 3 for a bulk three-dimensional system, 2 for a two-dimensional surface slab, and 1 for a one-dimensional system. Default value is 3.

`cell`

Unit cell parameters in a form of six real values $|\mathbf{a}|$, $|\mathbf{b}|$, $|\mathbf{c}|$, α, β, γ, where $|\mathbf{a}|$, $|\mathbf{b}|$, $|\mathbf{c}|$ are lengths of the appropriate cell vectors, α is the angle between vectors $\mathbf{b}$ and $\mathbf{c}$, β is the angle between vectors $\mathbf{a}$ and $\mathbf{c}$, and γ is the angle between vectors $\mathbf{a}$ and $\mathbf{b}$. Default are atomic units and degrees. You can specify unit cell parameters in Å and degrees using `cell ang`.

```
content
  label x y z
end
```

Content of the unit cell, where `label` is the label of the point charge Content of the unit cell, where `label` is the label of the point charge type and `x y z` are corresponding Cartesian or fractional crystal coordinates. Defaults are Cartesian coordinates and atomic units. You can specify Cartesian coordinates in Å using `content ang` and fractional coordinates using `content frac`. Note that Cartesian coordinates assume that the cell vector **a** is aligned along the x axis, and the vector **b** on the xy plane.

```
cluster
  label x y z
end
```

Atomic coordinates of the piece of the crystal to be replaced by the QM cluster and surrounding isolation shell (ECPs and explicit point charges), where `label` is the point charge label and `x y z` are corresponding Cartesian or fractional crystal coordinates. Defaults are Cartesian coordinates and atomic units. You can specify Cartesian coordinates in Å using `cluster ang` and fractional coordinates using `cluster frac`.

```
charges
  label charge
end
```

Values of point charges (for each atom-type) , where `label` is the point charge label and `charge` specifies charge in atomic units.

```
ch_list
  label charge
end
```

Values of point charges (for each atom), where `label` is the point charge label and `charge` specifies charge in atomic units.

Note that `charges` and `ch_list` are mutually exclusive. An integer number n can also be appended to `charges` or `ch_list` to set the tolerance for charge neutrality violation to 10^{-n} (default $n = 5$).

Additionally, the following keywords control the accuracy of PEECM calculation:

`lmaxmom`
: Maximum order of the multipole expansions in periodic fast multipole method (PFMM). Default value is 25.

`potval`
: Electrostatic potential at the lattice points resulting from periodic point charges field will be output if this keyword is present. Default is not to output.

`wsicl`
: Well-separateness criterion for PFMM. Default is 3.0.

`epsilon`
: Minimum accuracy for lattice sums in PFMM. Default is 1.0d-8.

23.2.13 Keywords for COSMO

The Conductor-like Screening Model (COSMO) is a continuum solvation model, where the solute molecule forms a cavity within the dielectric continuum of permittivity epsilon that represents the solvent. A brief description of the method is given in chapter 19.2. The model is currently implemented for SCF energy and gradient calculations (`dscf`/`ridft` and `grad`/`rdgrad`), MP2 energy calculations (`rimp2` and `mpgrad`) and MP2 gradients (`rimp2`), and response calculations with `escf`. The `ricc2` implementation is described in section 19.2.5.

For simple HF or DFT single point calculations or optimizations with standard settings, we recommend to add the `$cosmo` keyword to the `control` file and to skip the rest of this section.

Please note: due to improvements in the **A** matrix and cavity setup the COSMO energies and gradients may differ from older versions (5.7 and older). The `use_old_amat` option can be used to calculate energies (not gradients) using the old cavity algorithm of TURBOMOLE 5.7.
The basic COSMO settings are defined in the `$cosmo` and the `$cosmo_atoms` block. Example with default values:

```
$cosmo
  epsilon=infinity
  refind= 1.3
  nppa= 1082
  nspa=   92
  disex= 10.0000
  rsolv= 1.30
  routf= 0.85
  cavity closed
  ampran= 0.1D-04
  phsran= 0.0
# the following options are not used by default
```

```
solvent=not set
allocate_nps= 140
use_old_amat
use_contcav
no_oc
```

`epsilon=` *real*
: defines a finite permittivity used for scaling of the screening charges. If the option `ion` is added to the same input line, the scaling factor for ions $f(\varepsilon) = \frac{\varepsilon-1}{\varepsilon+x}$ with $x = 0$ will be used. Alternatively the x value can be set by adding `ion=`x with x as a real value.

`refind=` *real*
: refractive index used for the calculation of vertical excitations and num. frequencies (the default 1.3 will be used if not set explicitly)

`solvent=` *label*
: requests to use the values for `epsilon` and `refind` for a given solvent named *label* from the `cosmosolvent` list. E.g., for water, the values `epsilon`=80.10 and `refind`=1.3334 are set. Note that additional settings for `epsilon` and `refind` (see above) after `solvent` will overwrite these values.

`allocate_nps=`*integer*
: skips the COSMO segment statistics run and allocates memory for the given number of segments.

`no_oc`
: skips the outlying charge correction.

`use_old_amat`
: uses **A** matrix setup of `TURBOMOLE` 5.7

`use_contcav`
: in case of disjunct cavities only the largest contiguous cavity will be used and the smaller one(s) neglected. This makes sense if an unwanted inner cavity has been constructed e.g. in the case of fullerenes. Default is to use all cavities.

`point_charges`
: Allows the COSMO response to point charges. By default it is disables, so that point charges interacts only with the molecule. Note that when

point charges are used together with COSMO, the COSMO surface is still constructed considering only the atoms.

All other parameters affect the generation of the surface and the construction of the **A** matrix:

`nppa=` *integer*
number of basis grid points per atom
(allowed values: $i = 10 \times 3^k \times 4^l + 2 = 12, 32, 42, 92...$)

`nspa=` *integer*
number of segments per atom
(allowed values: $i = 10 \times 3^k \times 4^l + 2 = 12, 32, 42, 92...$)

`disex=` *real*
distance threshold for A matrix elements (Ångstrom)

`rsolv=` *real*
distance to outer solvent sphere for cavity construction (Ångstrom)

`routf=` *real*
factor for outer cavity construction in the outlying charge correction

`cavity closed`
pave intersection seams with segments

`cavity open`
leave untidy seams between atoms

`ampran=` *real*
amplitude of the cavity de-symmetrization

`phsran=` *real*
phase of the cavity de-symmetrization

If the `$cosmo` keyword is given without further specifications the default parameter are used (recommended). For the generation of the cavity, COSMO also requires the definition of atomic radii. User defined values can be provided in Ångstrom units in the data group `$cosmo_atoms`, e.g. for a water molecule:

```
$cosmo_atoms
# radii in Angstrom units
o  1                                                    \
   radius=  1.7200
h  2-3                                                  \
   radius=  1.3000
```

If this section is missing in the `control` file, the default values defined in the `radii.cosmo` file (located in `$TURBODIR/parameter`) are used. A user defined value supersedes this defaults. `$cosmo` and `$cosmo_atoms` can be set interactively with the COSMO input program `cosmoprep` after the usual generation of the `TURBOMOLE` input.

The COSMO energies and total charges are listed in the result section. E.g.:

```
SCREENING CHARGE:
  cosmo      :  -0.003925
  correction :   0.003644
  total      :  -0.000282
ENERGIES [a.u.]:
  Total energy            =      -76.0296831863
  Total energy + OC corr. =      -76.0297567835
  Dielectric energy       =       -0.0118029468
  Diel. energy + OC corr. =       -0.0118765440
  The following value is included for downward compatibility
  Total energy corrected  =      -76.0297199849
```

The dielectric energy of the system is already included in the total energy. `OC corr` denotes the outlying charge correction. The last energy entry gives the total outlying charge corrected energy in the old definition used in `TURBOMOLE` 5.7 and older versions. The COSMO result file, which contains the segment information, energies, and settings, can be set using: `$cosmo_out file=` *filename.cosmo*

Isodensity Cavity: This option can be used in HF/DFT single point calculations only. The `$cosmo_isodens` section defines the settings for the density based cavity setup (see also chapter 19.2). If the `$cosmo_isodens` keyword is given without suboptions, a scaled iosodensity cavity with default settings will be created. Possible options are:

`$cosmo_isodens`

activates the density based cavity setup. The default values of `nspa` and `nsph` are changed to 162 and 92, respectively. This values are superseded by the user defined `nspa` value of the `$cosmo` section. By default the scaled density method is used. The atom type dependent density values are read from the `radii.cosmo` file (located in `$TURBODIR/parameter`).

`dx=`*real*

spacing of the marching tetrahedron grid in Å (default: 0.3Å).

`all_dens=`*real*

use one isodensity value for all atom types (value in a.u.)

The outlying charge correction will be performed with a radii based outer cavity. Therfore, and for the smoothing of the density changes in the intersection areas of the scaled density method, radii are needed as for the standard COSMO cavity. **Please note:** The isodensity cavity will be constructed only once at the beginning of the SCF calculation. The density constructed from the initial mos will be used (file `mos` or `alpha/beta` in case of unrestricted calculations). Because the mos of an initial guess do not provide a good density for the cavity construction, it is necessary to provide mos of a converged SCF calculation (e.g. a COSMO calculation with standard cavity). We recommend the following three steps: perform a standard COSMO calculation, add the isodensity options afterwards, and start the calculation a second time.

Radii based Isosurface Cavity: The `$cosmo_isorad` section defines the radii defined isosurface cavity construction. The option uses the algorithm of the isodensity cavity construction but the objective function used depends on the cosmo radii instead of the electron density. The default values of `nspa` and `nsph` are changed to 162 and 92, respectively. This values are superseded by the user defined `nspa` value of the `$cosmo` section. The resulting surface exhibits smoother intersection seams and the segment areas are less diverse than the ones of the standard radii bases cavity construction.

`$cosmo_isorad`

`dx=`*real*

spacing of the marching tetrahedron grid in Å (default: 0.3Å).

COSMO in MP2 Calculations: The iterative COSMO PTED scheme (see chapter 19.2) can be used with the `mp2cosmo` script. Options are explained in the help message (`mp2cosmo -h`). Both MP2 modules `rimp2` and `mpgrad` can be utilized. The `control` file can be prepared by a normal COSMO SCF input followed by a `rimp2` or `mpgrad` input. The PTE gradients can be switched on by using the

`$cosmo_correlated`

keyword (`rimp2` only). Again a normal SCF COSMO input followed by a `rimp2` input has to be generated. The `$cosmo_correlated` keyword forces `dscf` to keep

the COSMO information needed for the following MP2 calculation and toggles on the COSMO gradient contribution.

COSMO in Numerical Frequency Calculations: `NumForce` can handle two types of COSMO frequency calculations. The first uses the normal relaxed COSMO energy and gradient. It can be performed with a standard `dscf` or `ridft` COSMO input without further settings. This is the right method to calculate a Hessian for optimizations. The second type, which uses the approach described in chapter 19.2, is implemented for `ridft` only. The input is the same as in the first case but `NumForce` has to be called with the `-cosmo` option. If no solvent refractive index `refind=`REAL is given in the `$cosmo` section of the control file the program uses the default (1.3).

COSMO in vertical excitations and polarizabilities: COSMO is implemented in `escf` and will be switched on automatically by the COSMO keywords of the underlying SCF calculation. The refractive index, used for the fast term screening of the vertical excitations, needs to be defined in the cosmo section of control file (`refind=`REAL).

COSMO in CC2 and ADC(2) calculations: For the calculation of ground-state energy at COSMO-CC2, vertival excitation energy at COSMO-CC2 and COSMO-ADC(2) and excited-state analytic gradient at COSMO-ADC(2), the post-SCF reaction-field scheme will be switched on by the `$reaction_field` keyword as:

```
$reaction_field
  post-SCF
  ccs-like
```

DCCOSMO-RS: The DCOSMO-RS model (see chapter 19.2) has been implemented for restricted and unrestricted DFT and HF energy calculations and gradients (programs: `dscf`/`ridft` and `grad`/`rdgrad`). In addition to the COSMO settings defined at the beginning of this section, the `$dcosmo_rs` keyword has to be set.

`$dcosmo_rs file=`*filename.pot*

activates the DCOSMO-RS method. The file defined in this option contains the DCOSMO-RS σ-potential and related data (examples can be found in the defaut potentials in the `$TURBODIR/parameter` directory).

If the potential file cannot be found in the local directory of the calculation, it will be searched in the `$TURBODIR/parameter` directory. The following σ-potential files

for pure solvents at 25 °C are implemented in the current TURBOMOLE distribution (see parameter subdirectory):

Water:	`h2o_25.pot`
Ethanol:	`ethanol_25.pot`
Methanol:	`methanol_25.pot`
Tetrahydrofurane:	`thf_25.pot`
Acetone:	`propanone_25.pot`
Chloroform:	`chcl3_25.pot`
Tetrachloromethane:	`ccl4_25.pot`
Acetonitrile:	`acetonitrile_25.pot`
Nitromethane:	`nitromethane_25.pot`
Dimethylsulfoxide:	`dimethylsulfoxide_25.pot`
Diethylether:	`diethylether_25.pot`
Hexane:	`hexane_25.pot`
Cyclohexane:	`cyclohexane_25.pot`
Benzene:	`benzene_25.pot`
Toluene:	`toluene_25.pot`
Aniline:	`aniline_25.pot`

The DCOSMO-RS energies and total charges are listed in the COSMO section of the output:

```
 SCREENING CHARGE:
   cosmo      :  -0.012321
   correction :   0.011808
   total      :  -0.000513
(correction on the COSMO level)
 ENERGIES [a.u.]:
   Total energy            =      -76.4841708454
   Outlying charge corr. (COSMO)    =    -0.0006542315
   Outlying charge corr. (DCOSMO-RS)=    -0.0011042856
   Combinatorial contribution of the solute =    -0.0017627889
   (at inf. dil. in the mixture/pure solvent. Not included in the total energy above)
```

The outlying charge correction cannot be defined straight forward like in the normal COSMO model. Therefore, the output shows two corrections that can be added to the `Total energy`. The first one is the correction on the COSMO level (`COSMO`) and the second is the difference of the DCOSMO-RS dielectric energy calculated form the corrected and the uncorrected COSMO charges, respectively (`DCOSMO-RS`). The

charges are corrected on the COSMO level only. The `Total energy` includes the $E_{diel,RS}$ defined in section 19.2.4. Additionally the combinatorial contribution at infinute dilution of the COSMO-RS model is given in the output. The use of this energy makes sense if the molecule under consideration is different than the used solvent or not component of the solvent mixture, respectively. To be consistent one should only compare energies containing the same contributions, i.e. same outlying charge correction and with or without combinatorial contribution. **Please note**: the COSMO-RS contribution of the DCOSMO-RS energy depends on the reference state and the COSMO-RS parameterization (used in the calculation of the chosen COSMO-RS potential). Therefore, the DCOSMO-RS energies should not be used in a comparision with the gas phase energy, i.e. the calculation of solvation energies.

23.2.14 Keywords for Module `riper`

`riper` shares most of the relevant keywords of the `dscf` and `ridft` modules. The `$dft` data group (see 23.2.10) and auxiliary basis sets defined using the keyword `$jbas` are always required.

For periodic calculations two additional keywords are necessary:

`$periodic` n

Specifies the number of periodic directions: $n = 3$ for a 3D periodic bulk solid, $n = 2$ for a 2D periodic surface slab and $n = 1$ for a 1D periodic system. The default value is 0 for a molecular system.

`$cell`

Specifies the unit cell parameters. The number of cell parameters depends on the periodicity of the system:

For 3D periodic systems six unit cell parameters $|\mathbf{a}|$, $|\mathbf{b}|$, $|\mathbf{c}|$, α, β and γ need to be provided. Here, $|\mathbf{a}|$, $|\mathbf{b}|$ and $|\mathbf{c}|$ are lengths of the appropriate cell vectors, α is the angle between vectors $\mathbf{b}$ and $\mathbf{c}$, β is the angle between vectors $\mathbf{a}$ and $\mathbf{c}$, and γ is the angle between vectors $\mathbf{a}$ and $\mathbf{b}$. `riper` assumes that the cell vectors $\mathbf{a}$ and $\mathbf{b}$ are aligned along the x axis and on the xy plane, respectively.

For 2D periodic systems three surface cell parameters $|\mathbf{a}|$, $|\mathbf{b}|$ and γ have to be provided. Here, $|\mathbf{a}|$ and $|\mathbf{b}|$ are lengths of the appropriate cell vectors and γ is the angle between $\mathbf{a}$ and $\mathbf{b}$. `riper` assumes that the cell vectors $\mathbf{a}$ and $\mathbf{b}$ are aligned along the x axis and on the xy plane, respectively.

For 1D periodic systems only one parameter specifying the length of the unit cell has to be provided. `riper` assumes that periodic direction is along the x

axis.

`$lattice`

Alternatively, lattice vectors can be provided. The number of cell parameters depends on the periodicity of the system:

For 3D periodic systems three (three-dimensional) lattice vectors need to be provided.

For 2D periodic systems two (two-dimensional) lattice vectors have to be provided. `riper` assumes that the lattice vectors are aligned on the xy plane.

For 1D periodic systems only one parameter specifying the length of the lattice vector has to be provided. `riper` assumes that periodic direction is along the x axis.

Optionally, for periodic systems a **k** points mesh can be specified:

`$kpoints`

`nkpoints` n_1 n_2 n_3

Specifies components along each reciprocal lattice vector of a Γ centered mesh of **k** points. In 3D periodic systems each **k** point is defined by its components k_1, k_2 and k_3 along the reciprocal lattice vectors $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ as

$$\mathbf{k} = k_1\mathbf{b}_1 + k_2\mathbf{b}_2 + k_3\mathbf{b}_3 \; . \tag{23.1}$$

For 2D periodic systems $k_3 = 0$. In case of 1D periodicity $k_3 = 0$ and $k_2 = 0$. The three components k_j $(j = 1, 2, 3)$ of **k** are given as

$$k_j = \frac{i}{n_j} \text{ with } i = -\frac{n_j - 1}{2}, -\frac{n_j - 1}{2} + 1, \ldots, \frac{n_j - 1}{2} - 1, \frac{n_j - 1}{2} \; . \tag{23.2}$$

with n_j $(j = 1, 2, 3)$ as integer numbers. For 3D periodic systems n_1, n_2 and n_3 have to be specified. The component n_3 can be omitted for 2D periodic systems. In case of 1D periodicity only the n_1 has to be specified.

In addition, the options `kptlines` and `recipr` within the `$kpoints` group can be used to obtain band structure plots:

`kptlines` Specifies the number of reciprocal space lines for band structure plots.

`recipr` Specifies lines in the reciprocal space using three real numbers defining the start point of the line and three real numbers defining its end point. Finally, the number of **k** points along the line is given as an integer number.

In the following example band energies are calculated along four lines, as specified by the keyword `kptlines 4`. Each line definition starts in a new line with the keyword `recipr`, followed by three real numbers defining the start point of the line and three real numbers defining its end point. Finally, the number of **k** points along the line is given as an integer number. Thus, the first line starts at the point `(0.500 0.500 0.500)`, ends at `(0.000 0.000 0.000)` and contains 40 **k** points.

```
$kpoints
 kptlines 4
 recipr 0.500 0.500 0.500 0.000 0.000 0.000 40
 recipr 0.000 0.000 0.000 0.500 0.500 0.000 40
 recipr 0.500 0.500 0.000 0.746 0.373 0.373 40
 recipr 0.746 0.373 0.373 0.000 0.000 0.000 40
```

The calculated band structure is written to the file `bands.xyz`. Each line of the file contains five real numbers: the coordinates k_1, k_2 and k_3 of the **k** point, its length $|k|$ and the corresponding band energy $\epsilon_{n\mathbf{k}}$.

Keywords within the section `$riper` can be used to control precision and parameters of algorithms implemented in `riper`. If the `$riper` group is absent, the following default values are used:

```
$riper
  # general keywords
   thrints    1.0d-12
   lenonly    off
   lchgprj    on  (for periodic systems)
   lchgprj    off (for molecular systems)
   northol    5
   pqmatdiag  off
   pqsingtol  1.0d-8
  # CFMM control options
   lmaxmom    20
   nctrgt     10 (for periodic systems)
   nctrgt     1  (for molecular systems)
   wsicl      3.0
   epsbext    1.0d-9
   locmult    on  (for periodic systems)
```

```
 locmult    off (for molecular systems)
 locmomadd  2
# LMIDF control options
 lpcg       on
 lcfmmpcg   on
 lmxmompcg  20
 pcgtol     1.0d-9
 pcgtyp     sp
# Gaussian smearing options
 sigma      0.0d0
 desnue     0.0d0
```

The following options are available:

```
# general keywords
```

`thrints` Threshold for integrals neglect and for differential overlap when screening basis functions pairs. Probably never needs to be changed.

`lenonly` Flag for energy calculation only, no gradients.

`lchgprj` If set to `on` charge projection of the auxiliary electron density [176] is performed for molecular systems during calculation of the Coulomb term. The charge projection constraints the charge of auxiliary density exactly to the number of electrons in the system. It is required for periodic systems, otherwise the Coulomb energy would be infinitely large. For molecular systems charge projection leads to a slight increase of the RI fitting error. It may be useful in some cases but we have so far not identified any.

`northol` Forces orthonormalization of orbital coefficients every `northol` SCF iteration.

`pqmatdiag` If set to `on` full diagonalization of the Coulomb metric matrix [176] is performed and used to solve density fitting equations. When diffuse auxiliary basis functions are used the default Cholesky decomposition of the Coulomb metric matrix may fail due to small negative eigenvalues. In this case the slower method based on a full diagonalization of the metric matrix is necessary.

`pqsingtol` If `pqmatdiag` is used `pqsingtol` sets threshold for neglect of small eigenvalues of the Coulomb metric matrix.

```
# CFMM control options
```

`lmaxmom` Maximum l-moment of multipole expansions used for calculation of the Coulomb term. The default value hardly ever needs to be changed.

`nctrgt` Target number of charge distributions per lowest level box of the octree [172]. The default value hardly ever needs to be changed.

`wsicl` Sets the well-separateness criterion [172]. Octree boxes with centers separated more than sum of their lengths times $0.5\times$`wsicl` are considered as well-separated. The default hardly ever needs to be changed. `wsicl` makes sense only for values ≥ 2.0. For `wsicl` ≤ 3.0 increasing `lmaxmom` may be necessary for reasonable accuracy.

`epsbext` Precision parameter used to determine basis function extents [172].

`locmult` If set to `on`, an additional acceleration method employing local multipole expansions is used. For periodic systems this leads to a significant speedup of calculations, especially for small unit cells and/or diffuse basis functions. Default value is `off` and `on` for molecular and periodic systems, respectively.

`locmomadd` For `locmult` set to `on` the order of local multipole expansions is increased by `locmomadd`. The default value probably never needs to be changed.

`# LMIDF control options`

`lpcg` If set to `on` the low-memory iterative density fitting (LMIDF) scheme is used for solving the RI equations [175] using the preconditioned conjugate gradient (PCG) method. It is implemented for molecular systems only. Default value is `off`.

`lcfmmpcg` If `lpcg` is used, `lcfmmpcg` specifies whether the CFMM is applied for evaluation of the matrix-vector products needed for the PCG solver. Not employing CFMM speeds up the calculations but significantly increases memory demand since the full Coulomb metric matrix has to be stored. Default value is `on`.

`lmxmompcg` Maximum l-moment of multipole expansions for calculations of Coulomb interactions within the PCG algorithm. It should be set to the same or larger value than `lmaxmom`.

`pcgtol` Sets the threshold parameter controlling accuracy of the PCG solver (see [175] for details). Default value is $1.0\cdot 10^{-9}$. For lower-precision calculations it can be set to $1.0\cdot 10^{-8}$ but values larger

than $1.0 \cdot 10^{-7}$ are not allowed as these lead to large errors in Coulomb energies and occasionally to SCF convergence problems.

`pcgtyp` *char*

char = `at`, `ss` or `sp`

Specifies the type of preconditioner used in the PCG algorithm. Three types of preconditioners are implemented and are defined explicitly in Sec. 7. The `sp` preconditioner is a default one performing consistently the best among the preconditioners considered. The `at` preconditioner is less efficient in decreasing the number of CG iterations needed for convergence. However, it has negligible memory requirements and more favorable scaling properties, albeit with a large prefactor. The `ss` preconditioner represents a middle ground between the `sp` and `at` preconditioners both in terms of the efficiency and memory requirements.

`# Gaussian smearing options`

`sigma` Width of the Gaussian smearing in hartree. See ref. [178] for more information. Note, that [178] uses eV as the unit for the width of the Gaussian smearing.

`desnue` Specifying `desnue` along with `sigma` forces occupancy leading to the number of unpaired electrons equal to `desnue`.

Keywords within the section `$pointvalper` can be used to evaluate quantities (electron density/molecular orbitals) for visualization on grid points:

```
$pointvalper fmt=plt
     dens
     orbs 2
     k 3 2 1 a 1 i
     k 0 0 0 b 2 r
     nimg 2 2 1
     npts 100 100 100
     eps 5.0
     ngrdpbx 50
     full
```

The following options are available:

`fmt`	Specifies output format. Currently `.plt`, `.cub`, `.xyz` and `.upt` extensions are supported (see 7.3.8).
`dens`	If present, total (and spin for UHF) density is calculated.
`orbs`	Specifies the number of plotted orbitals. For further details see subsection 7.3.8.
`nimg`	Number of unit cell images `n1`, `n2` and `n3` in the periodic directions **a**, **b** and **c**, respectively, for which plot data is generated.
`npts`	Number of grid points `n1`, `n2` and `n3` along each periodic direction. If not specified, value 100 is used for each `n`.
`eps`	Specifies the distance `real` in bohr around the system for which plot grid is generated in aperiodic directions. Default value is 5 bohr.
`ngrdpbx`	Number of grid points stored in one octree box during density calculations. Default value is 50. For very large systems or high resolution it may be necessary to increase this parameter to avoid memory allocation problems.
`full`	Only valid for `plt` output format. This format uses orthogonal grids. Therefore, for non-orthogonal unit cells grid data is generated for a rectangular box that contains the supercell (unit cell and its periodic images). By default, the values at grid points outside of the supercell are set to zero. For strongly non-orthogonal systems this may lead to large files. The option `full` switches off the zeroing of values on grid points outside the supercell.

To calculate a simulated density of states (DOS) keyword `$dosper` is necessary:

```
$dosper width=real emin=real emax=real scal=real npt=integer
```

The following options are available:

`width`	The width of each Gaussian, default value is 0.01 a.u.
`emin,enmax`	Lower/upper bounds for energy in DOS calculation.
`scal`	Scaling factor for DOS (total and s-, p-, ... contributions).
`npt`	Resolution (number of points).

Keywords within the section `$rttddft` can be used to specify the parameters for RT-TDDFT. Example:

```
$rttddft
  magnus 2
  scf off
  time 1000.0d0
  tstep 0.1d0
  min energy = 0.013d0
  max energy = 0.5d0
  energy step 0.001d0
```

The options are explained below.

`magnus` Can take values 2 or 4. "2" for second order Magnus expansion and "4" for fourth order Magnus expansion. Default value is 2.

`scf` if `on`, then SCF procedure is used for the time integration. If `off` then Predictor-Corrector scheme is used instead.

`iterlim` Max SCF cycles if `scf` is `on`. Default value is 15.

`time` Specifies the evolution time in au. (1 au= 0.02419 fs)

`tstep` The time step for the time evolution in au. 0.1 au is usually a good starting point.

`print step` Specifies the number of steps n after which the dipole moments and energies are printed out if requested. Default value is 100. That means the quantities are printed out at every 100 steps. To have all the information for post processing, a value of 1 is recommended.

`damping` Only valid for absorption spectrum calculation. It is the factor gamma in the equation to calculate the complex polarizability tensor. Default value is 0.004 au. Recommended values in the range of 0.003 au to 0.005 au.

`min energy` Only valid for absorption spectrum calculation. Specifies the minimum value of the energy range used to perform the Fourier transform from time to frequency space. Units: au. Default value is 0.15 au.

`max energy` Only valid for absorption spectrum calculation. Specifies the maximum value of the energy range used to perform the Fourier

transform from time to frequency space. Units: au. Default value is 0.625 au.

`energy step` Only valid for absorption spectrum calculation. Specifies the step value or energy interval `dE` at which to sample the energy values for Fourier transform and absorption spectrum plotting. Units: au. Default value is 0.005 au.

Keywords within the section `$electric field` can be used to specify the parameters for electric field. To specify a Gaussian electric field pulse use

```
$electric field
  amplitude x=2.0E-5 y=2.0E-5 z=2.0E-5
  gaussian tzero=3.0 width=0.2
```

Options are explained below.

`amplitude x y z` Cartesian components of the amplitude of the Gaussian pulse (in Electric field au).

`gaussian` Sets the electric field type to Gaussian.

`tzero` Position of Gaussian pulse maximum (in time au).

`width` Width of Gaussian pulse (in time au).

23.2.15 Keywords for Modules `grad` and `rdgrad`

Many of the `dscf` and `ridft` keywords are also used by `grad` and `rdgrad`.

`$drvopt`

This keyword and corresponding options are required in gradient calculations only in special circumstances. Just `$drvopt` is fine, no options needed to compute derivatives of the energy with respect to nuclear coordinates within the method specified: SCF, DFT, RIDFT.

If running a DFT gradient calculation, it is possible to include the derivatives of the quadrature weights, to get more accurate results. In normal cases however those effects are marginal. An exception is numerical calculation of frequencies by `NumForce`, where it is strongly recommended to use the weight derivatives option. The biggest

deviations from the uncorrected results are to be expected if doing gradient calculations for elements heavier than Kr using all electron basis sets and very small grids. To use the weight derivatives option, add

```
weight derivatives
```

in `$dft`.

The option

```
point charges
```

in `$drvopt` switches on the evaluation of derivatives with respect to coordinates of point charges. The gradients are written to the file `$point_charge_gradients` old gradients will be overwritten.

23.2.16 Keywords for Module `aoforce`

This module calculates analytically harmonic vibrational frequencies within the HF- or (RI)DFT-methods for closed-shell and spin-unrestricted open-shell-systems. Broken occupation numbers would lead to results without any physical meaning. Note, that RI is only used partially, which means that the resulting Hessian is only a (very good) approximation to exact second derivatives of the RIDFT-energy expression. Apart from a standard force constant calculation which predicts all (allowed and forbidden) vibrational transitions, it is also possible to specify certain irreps for which the calculation has to be done exclusively or to select only a small number of lowest eigenvalues (and eigenvectors) that are generated at reduced computational cost.

General keywords

`$drvopt`

is the keyword for non-default options of gradient and second derivative calculations. Possibilities in case of the module `aoforce` are:

`frequency analysis only`

`analysis only`

to read a complete Hessian from the input file `$hessian` and perform only the frequency analysis

`analysis [only] intcoord [print` *printlevel*`]`

to perform an analysis of normal modes in terms of internal coordinates. Details about this option and the effect of the printlevel (default is 0)

are given in Section 15. The effect of the keyword `only` is the same as described above.

`$maxcor 50`

fixes the amount of core memory to be used for dynamically allocated arrays (here 50 MiB), about 70% of available memory should be fine, because `$maxcor` specifies only the memory used to store derivatives of density and Fock matrices as well as right hande side vectors for the CPHF equations. For further details see subsection 23.2.3.

`$forceconv 7`

sets the convergence criterion for the CPHF-equations to a residual norm of 1.0d-7. Normally the default value of 1.0d-5 already provides an accuracy of vibrational frequencies of 0.01 cm^{-1} with respect to the values obtained for the convergence limit.

`$forceiterlimit 10`

fixes the maximum number of Davidson iterations for the solution of the CPHF-equations to a value of ten. Normal calculations should not need more than eight iterations, but as a precaution the default value is 25.

`$nosalc`

forces the program in case of molecules with C_1 symmetry not to use $3N-6(5)$ symmetry adapted but all $3N$ cartesian nuclear displacement vectors. This option may lead to a moderate speed-up for molecules notedly larger than 1000 basis functions and 100 atoms.

`$noproj`

forces the program not to project out translations and rotations when forming a basis of symmetry adapted molecular displacements. This option may be needed if a Hessian is required, that contains translation- and rotation-contributions, e.g. for coupling the system with low cost methods. Output of the unprojected hessian is done on `$nprhessian`; format is the same as for conventional `$hessian`. Output of the corresponding eigenvalues and eigenvectors is done analogously on `$nprvibrational spectrum` and `$nprvibrational normal modes`.

`$nomw`

causes the program to diagonalize a not mass weighted hessian. Output is on `$nprhessian`, `$nprvibrational spectrum` and `$nprvibrational normal modes`, because projection of rotations is not possible in this case.

`$isosub`

This keyword allows to trace back the effects of isotopic substitution on vibrational frequencies. The atom(s) for which isotopic substitution is to be investigated are specified in subsequent lines of the form (atom index) (mass in special isotope), e.g.

```
$isosub
 3 2.001
 5 13
```

The interpolation then takes place between the mass(es) specified in `$atoms` (or the default mass(es), if none specified) and the mass(es) in `$isosub`. Take care of symmetry equivalent atoms, otherwise symmetry analysis will fail. This feature can not be used in a lowest eigenvalue search (keyword `$les`).

`$isopts 6`

Sets the number of points for interpolation between the two isotopes compared by the `$isosub` option to six. Default value is 21.

Keywords for the treatment of only selected nuclear displacement vectors:

`$ironly`

CPHF-iteration is done only for distortions, that are IR active.

`$ramanonly`

CPHF-iteration is done only for distortions, that are Raman active.

`$vcd`

Calculation of VCD rotational strenghts after preceding `mpshift`run.

`$les`

This causes a lowest Hessian eigenvalue search to be performed instead of a complete force constant calculation. The lowest eigenvalue search consists of the calculation of a guess-Hessian and macro-iterations to find the solution vector(s) for the lowest eigenvalue(s). In each macro-iteration the CPHF-equations are solved for the present search vector(s). `$les all 1` means that one lowest eigenvalue for each irrep will be determined, other numbers of lowest eigenvalues per irrep are admissible too.

Different numbers of lowest eigenvalues for different irreps are requested by e.g.

```
$les
 a1 3
```

```
a2 all
b2 1
```

The convergence criterion of the Davidson iterations for the solution of the CPHF-equations as well as the maximal residual norm for the lowest Hessian eigenvalue in the macro-iteration are specified by `$forceconv` as explained above.

The maximum number of macro-iterations is specified by `$lesiterlimit x`

with the default `x`=25. The maximum number of iterations for each solution of the CPHF-equations is again determined by `$forceiterlimit` as shown above.

The convergence of the macro-iterations is strongly influenced by the size of the starting search-subspace. Generally all guess-Hessian eigenvectors corresponding to imaginary frequencies and at least two real ones are used as starting search-subspace. However it proved to be necessary to use even more vectors in the case of guess-Hessians with very large conditioning numbers.
`$hesscond 8.0d-5`
means that all eigenvalues with the quotient (eigenvalue)/(max. eigenvalue) lower than 0.00008 are added to the starting search-subspace. Default is `1.0d-4`.

`$hotfcht`

Triggers the generation of input files for hotFCHT (program to calculate Franck-Condon-factors by R. Berger and co-workers). See 15.5.

`$sijuai_out`

Save the derivative of the density matrix for subsequent use in the module `evib`. See 16

`$dipgrad`

This keyword is the output of a default `aoforce` run and contains the dipole derivatives w.r.t. atomic Cartesian coordinates. The result is given as 3x3 tensors for each atom (first three lines are for atom 1, next three lines for atom 2, etc.). In each 3x3 matrix per atom the columns are the dipole moment direction and the rows are three atomic cartesian coordinate directions:

```
$dipgrad      cartesian dipole gradients
```

$$
\begin{array}{lll}
d(\mu_x)/d(x_1) & d(\mu_y)/d(x_1) & d(\mu_z)/d(x_1) \\
d(\mu_x)/d(y_1) & d(\mu_y)/d(y_1) & d(\mu_z)/d(y_1) \\
d(\mu_x)/d(z_1) & d(\mu_y)/d(z_1) & d(\mu_z)/d(z_1) \\
d(\mu_x)/d(x_2) & d(\mu_y)/d(x_2) & d(\mu_z)/d(x_2) \\
d(\mu_x)/d(y_2) & d(\mu_y)/d(y_2) & d(\mu_z)/d(y_2) \\
\ldots &&
\end{array}
$$

```
$end
```

Force constant calculations on the DFT level prove to be numerically reliable only with large integration grids or if one includes the effects of quadrature weights. This is done by default—to prevent this, insert

```
no weight derivatives
```

in `$dft`.

23.2.17 Keywords for Module `evib`

`$dfdxi textout`

can be used to generate text output of the matrix elements of the derivative of the Fock-operator. For bigger systems this can however generate very large output files. See 16

23.2.18 Keywords for Module `escf`

TDHF and TDDFT calculations

To perform an `escf` calculation converged molecular orbitals from a HF, DFT or RIDFT calculation are needed. The HF, DFT or RIDFT method is chosen according to the `$dft` or `$ridft` keywords, specified above. It is recommended to use well-converged orbitals, specifying `$scfconv 7` and `$denconv 1d-7` for the ground-state calculation. The input for an `escf` calculation can be conveniently generated using the `ex` menu in `define`, see Section 4.

During an `escf` run, a system-independent formatted logfile will be constructed for each IRREP. It can be re-used in subsequent calculations (restart or extension of eigenspace or of `$rpaconv`). An `escf` run can be interrupted by typing "touch stop" in the working directory.

In an `escf` run one of the following properties can be calculated: (please note the 'or' in the text, do only one thing at a time.)

1. RPA and time-dependent DFT singlet or triplet or spin-unrestricted excitation energies (HF+RI(DFT))

`$scfinstab rpas` or

`$scfinstab rpat` or

`$scfinstab urpa`

2. TDA (for HF: CI singles) singlet or triplet or spin-unrestricted or spin-flip excitation energies (HF+RI(DFT))

`$scfinstab ciss` or

`$scfinstab cist` or

`$scfinstab ucis` or

`$scfinstab spinflip`

3. Two-component TDDFT/TDA excitation energies of Kramers-restricted closed-shell systems

`$scfinstab soghf` or

`$scfinstab tdasoghf`

4. Eigenvalues of singlet or triplet or non-real stability matrices (HF+RI(DFT), RHF) or complex stability matrices (HF+RI(DFT), Kramers-GHF)

`$scfinstab singlet` or

`$scfinstab triplet` or

`$scfinstab non-real` or

`$scfinstab complex`

5. Static polarizability and rotatory dispersion tensors (HF+(RI)DFT, RHF+UHF)

`$scfinstab polly`

6. Dynamic polarizability and rotatory dispersion tensors (HF+(RI)DFT, RHF+UHF)

`$scfinstab dynpol` *unit*
list of frequencies

where *unit* can be eV, nm, rcm; default is a.u. (Hartree). For example, to calculate dynamic polarizabilities at 590 nm and 400 i nm (i is the imaginary unit):

```
$scfinstab dynpol nm
  590
  400 i
```

The number and symmetry labels of the excited states to be calculated is controlled by the data group `$soes`. Example:

```
$soes
b1g  17
eu   23
t2g  all
```

will yield the 17 lowest excitations in IRREP b1g, the 23 lowest excitations in IRREP eu, and all excitations in IRREP t2g. Specify `$soes all`n; to calculate the n first excitations in all IRREPS. If n is not specified, all excitations in all IRREPS will be obtained. Both static and dynamic polarizabilites can be calculated with (local) all-electron relativistic methods including the picture-change correction of the corresponding dipole moment in one and two-component calculations. The picture-change correction is invoked by the keyword `$pcc` and the two-component framework by `$soghf`.

Adding `$damped_response` will trigger a damped response calculations at the give frequencies. Example:

```
$damped_response 0.25 eV
```

The damping factor must additionaly be specified in eV, cm-1 or nm.

7. Two-photon absorption

`$scfinstab twophoton rpas` or

`$scfinstab twophoton urpa` or

`$scfinstab twophoton ciss` or

`$scfinstab twophoton ucis`

8. Nuclear spin-spin coupling constants

The data group `$ncoupling` can be set using the `ncoup` section in `define`. A sample input might look like this:

```
$ncoupling
  fc sd pso dso nofcsdcross     or
  simple                        or
  fromfile
  reduced
  thr=0.1
$nucsel 1,3,5-8
$nucsel2 "c","h"
```

You can specify the contributions to be calculated by indicating the appropriate abbreviation. If both `fc` and `sd` are specified, the FC/SD cross term is calculated, unless manually switched off.

The `simple` method only calculates the FC and FC/SD cross terms, thus yielding the most important contributions to the isotropic and anisotropic part. Be warned that this might yield qualitatively wrong results in some cases! This method is quite fast because only response equations for the FC term are solved. However, this means that it is incompatible with `$nucsel2`.

If the option `fromfile` is set, SSCCs from the data group `$coupling_reduced` from a previous calculation are used. This option can be used to obtain SSCC for a different isotope without redoing the expensive part of the calculation.

If none of the keywords mentioned above is given, all terms (equivalent to `fc sd pso dso`) are calculated.

In a two-component calculation, `fc`, `sd`, and `pso` shall always be used together and `nofcsdcross` is not permissible. The `simple` method is likewise not allowed. If the DSO term is to be computed by picture-change correction, it needs to be activated here and `$pcc` needs to be set.

By default, the output is multiplied by the gyromagnetic ratios, yielding J in Hertz. With `reduced`, the reduced coupling constants K are printed in units of $10^{19}\,\mathrm{T}^2/\mathrm{J}$. The content of `$coupling_reduced` is not influenced by this setting.

In the final output, couplings where both the isotropic and the anisotropic part are smaller than the threshold `thr` (in Hertz or $10^{19}\,\mathrm{T}^2/\mathrm{J}$, default: 0.1) will not be printed.

You can specify for which atoms coupling constants shall be calculated by the data groups `$nucsel` and `$nucsel2`. As shown above, the syntax is either a list of atomic indices or of element symbols. The default is to calculate all atoms. Response equations (the most expensive step) are solved for the atoms in `$nucsel` and right-hand side integrals are calculated for atoms in any of the data groups. This means

that a coupling tensor is obtained if at least one of the partner is in `$nucsel`. Notice that the very same data group `$nucsel` is used by the module `mpshift`.

It is furthermore recommended to set `$rpaconv` to at least 6.

The file referenced in `$coupling_reduced` (default file name: `coupling.reduced`) is suitable for automated post-processing. It consists of the atomic indices, J^{iso}, $\gamma_K \cdot \gamma_L$, and the nine elements of the matrix $\tilde{K} = J/\gamma_K\gamma_L$ (see eqs. 8.24 and 8.25 for an explanation of these quantities). The gyromagnetic ratios γ_K are given in $10^7\,\mathrm{rad\,s^{-1}\,T^{-1}}$. All couplings with $K \geq L$ (ignoring `$nucsel` and `thr`) are printed.

general keywords

`$rpacor` *n*

The maximum amount of core memory to be allocated for the storage of trial vectors is restricted to *n* MB. If the memory needed exceeds the threshold given by `$rpacor`, a multiple pass algorithm will be used. However, especially for large cases, this will increase computation time significantly. The default is 200 MB.

`$spectrum` *unit*

The calculated excitation energies and corresponding oscillator strengths are appended to a file named 'spectrum'. Possible values of `unit` are eV, nm and cm^{-1} or rcm. If no unit is specified, excitation energies are given in a.u.

`$cdspectrum` *unit*

The calculated excitation energies and corresponding rotatory strengths are appended to a file named 'cdspectrum'. `unit` can have the same values as in `$spectrum`.

`$start vector generation` *e*

Flag for generation of UHF start MOs in a triplet instability calculation. The option will become effective only if there are triplet instabilities in the totally symmetric IRREP. The optional real number *e* specifies the approximate second order energy change in a.u. (default: 0.1).

`$velocity gauge`

Enables calculation of dipole polarizability/rotatory dispersion in the velocity gauge. Active only for pure DFT (no HF exchange).

`$sum rules` *unit*
list of frequencies
: Enable calculation of oscillator and rotatory strength sum rules at frequencies specified by *list of frequencies* in unit *unit* (see `$scfinstab dynpol`). Note that the sums will be taken only over the states specified in `$soes`.

`$rpaconv` n
: The vectors are considered as converged if the Euclidean residual norm is less than 10^{-n}. Larger values of n lead to higher accuracy. The default is a residual norm less than 10^{-5}. For SSCCs, it is recommended to increase n to 6.

`$escfiterlimit` n
: Sets the upper limit for the number of Davidson Iterations to n. Default is $n = 25$.

`$escfnoxc`
: Disable the DFT XC kernel contribution (on the grid). Required for the TDDFT-as/TDDFT-ris approaches, see Section 8.4.16.

GW Keywords

`$gw`

The main keyword that switches on a *GW* calculation using full spectral representations. This keyword will perform a standard G_0W_0 calculation with default values for the other flags.

`$rigw`

The main keyword that switches on a *GW* calculation using analytic continuation of the self-energy from imaginary to real space.

There are several options which can be added to the $gw or $rigw keyword. The recommended settings for a quick G_0W_0 run are:

Full quasiparticle spectrum using spectral representations:

```
$gw
  rpa
  eta 0.001
```

HOMO-LUMO gap using analytic continuation:

```
$rigw
  rpa
  ips+1
  gap
```

Orbital 3-6 using contour deformation and iterative solution of the QP equation:

```
$rigw
  rpa
  eta 0.001
  contour start=3 end=6
  qpeiter 10
```

With the optional entries:

```
    rpa
```

Default: false (not set). If added as option pure rpa response function is calculated. If not added, the TDDFT response function is calculated and used to screen the coulomb interaction. In combination with $rigw this keyword is mandatory, with $gw it should also always be set and only be deactivated by experts. The gw menu in `define` always sets this as default.

```
  qpeiter <integer>
```

Default: 0. Switches between linearized quasiparticle equation for 0 and iterate quasiparticle equation for > 0. eta is changed from 0.2 to the supplied value to obtain smooth convergence during the iteration. Only with $gw.

```
    gw0
```

Default: false (not set). A GW_0 calculation is performed instead of G_0W_0. The number of GW_0 iterations performed is set by qpeiter.

```
    evgw
```

Default: false (not set). An eigenvalue-only selfconsistent GW calculation is performed instead of G_0W_0.

```
    scgw
```

Default: false (not set). A quasiparticle selfconsistent GW calculation is performed instead of G_0W_0. Only with $gw and not compatible with two-component calculations.

```
offpq <real>
```

Default: 0.03. Infinitesimal complex energy shift in Fock-matrix contribution of self-energy in scGW calculation. Only in combination with scgw keyword, ignored otherwise.

```
fdamp <real>
```

Default: 0.3. Damping of new Fock-matrix in scGW calculation. Only in combination with scgw keyword, ignored otherwise.

```
unlimitz
```

Default: false (not set). In linearized G_0W_0 the linearization Factor Z is allowed to take any value instead of values between 0.5 - 1.0. Ignored if calculation is not a linearized G_0W_0 calculation.

```
fixz
```

Default: false (not set). In linearized G_0W_0 the linearization Factor Z is fixed to 1.0. Ignored if calculation is not a linearized G_0W_0 calculation.

```
csf <integer>
```

Default: false (not set). Calculate self-energy on an energy grid for the specified orbital. Results are saved on file.

```
nl <integer>
```

Default: n_{occ}+5. Number of orbitals to calculate gw for. It is set to the number of occupied orbitals + 5 if set smaller than the number of occupied orbitals.

```
eta <real>
```

Default: 0.001. Infinitesimal complex energy shift η. Negative value switches to calculating at that value but extrapolating to 0 in linear approximation. Note that η^2 is subsequently used internally.

```
output <filename>
```

Default: qpenergies.dat. Output filename for the quasiparticle energies.

For $rigw there are some special keywords that control the cost and efficiency of the module. The defaults chosen by define (or `escf`itself) are suitable for most systems. They were selected rather to be rather tight to yield reliable results without much knowledge of the system.

```
gap
```

Default: not set. Only the HOMO and LUMO quasiparticle states are calculated, all other orbitals are shifted by the HOMO-LUMO gap. In conjunction with the "ips+<integer>" keyword the N highest occupied and N lowest unoccupied states are calculated, making $rigw calculations feasible for many systems.

```
ips+ <integer>
```

Default: not set. When $rigw is set then the the quasiparticle states of all occupied + the N lowest virtual orbitals are calculated. Only with $rigw, ignored with $gw.

```
contour start=<integer> end=<integer> spin=<integer>
```

Default: not set. start, end and spin are optional and may be defined in arbitrary order. start defines the starting point for the orbital range and end the ending of the orbital range to be treated with RI-CD-*GW*. If neither start nor end are defined then the definition from the ips+ and gap keywords is used. In open shell systems spin=1 calculates the self-energy only for alpha shells, spin=2 for alpha and beta shells. For closed shell or any 2c calculation spin=1 is always the correct option and does not need to be set by the user.

```
npade <integer>
```

Default: 128. Number of Pade approximants used in the $rigw module for RI-AC-*GW*. Recommended to be the same as npoints in a dual-grid ansatz. Due to numerical instabilities not more than 256 Pade approximants should be used!

`npoints <integer>`

Default: 128. Number of imaginary frequency integration points used in the $rigw module for RI-AC-$GW$, RI-AC-$GW$ and RPA energies. Recommended to be the same as npade in a dual-grid ansatz in RI-AC-GW.

`rpoints <integer>`

Default: 16. Number of pseudo Fermi-levels in RI-AC-GW calculation. Not used if "contour" keyword is present.

`rshift <real>`

Default: 0.3 eV. Initial shift for first Fermi level when RI-AC-GW is used in eV. For insulators 0.3 is recommended, for lower gap systems ($<$ 3-4 eV HOMO-LUMO gap) 0.2 is recommended. Not used if "contour" keyword is present.

`width <real>`

Default: rshift. stepwidth between subsequent Fermi levels when RI-AC-GW is used in eV. Recommended to be set such that $e_{HOMO}+rshift+rpoints*width < e_{LUMO}$. Usually values from 0.05-0.20 eV are reasonable if the given condition is fulfilled. Not used if "contour" keyword is present.

`fscdgrd`

Default: not set. Approximative RI-CD-GW variant, employing a frequency-sampling strategy for the calculation of the required residues.

`thrs1 <real>`

Default: 1.361 eV. Threshold for selection of frequency grid based on absolute differences in frequency-sampled RI-CD-GW. Used only if `fscdgrd` is set.

`thrs2 <real>`

Default: 0.001 (0.1%). Threshold for selection of frequency grid based on relative energy differences in frequency-sampled RI-CD-GW. Used only if `fscdgrd` is set.

BSE Keywords

`$bse`

The main keyword that switches on a BSE calculation. Provided that the response function is calculated setting this keyword will perform a standard BSE calculation with default values for the other flags.

`$cbse`

The main keyword that switches on a correlation augmented BSE (=cBSE) calculation. A cBSE calculation with default values for the other flags will be performed.

The following optional entries can be added to the `$bse` or `$cbse` keyword:

```
noqpa
noqpw
```

Default: Not set. If set, the matrices **A** and **W**, respectively, are constructed using KS/HF orbital energies instead of GW quasi-particles energies. if $cbse is used noqpw is set automatically.

```
file <filename>
```

Default: `qpenergies.dat`. This option defines the file name of GW quasi-particle energies which are read in.

```
iterative
```

Default: Not set. Compute the auxiliary matrix for the static screened interaction iteratively instead of by Cholesky decomposition.

```
thrconv <real>
```

Default: 1.0d-12. Threshold for convergence in case of the iterative computation of the static screened interaction.

```
iterlim <integer>
```

Default: 100. Maximum number of iterations in case of the iterative computation of the static screened interaction.

`$keep_fxc`

Default: not set. Appears **outside** of the $bse data group. It adds the correlation part of the underlying XC Kernel to the BSE response. Automatically set if $cbse is set.

23.2.19 Keywords for Module `rirpa`

The keyword `$rirpa` allows to specify the following options,

`npoints` n
: Number of frequency quadrature points, n (default is 60).

`gausslegendre` *`mapping`*
: Use Gauss–Legendre instead of Clenshaw–Curtis quadrature. *`mapping`* defines the mapping function to determine the quadrature abscissae and weights, and may be `clenshaw` (which combines Gauss–Legendre with Clenshaw–Curtis mapping) or `rational`. If *`mapping`* is omitted, Clenshaw–Curtis mapping is the default. Gauss–Legendre quadrature usually converges more rapidly than Clenshaw–Curtis only, especially for difficult cases such as small gap and open shell systems.

`maxitergrid` n
: Maximum number of nested Clenshaw–Curtis quadrature iterations. During each iteration, the number of quadrature points doubles. The default is 1; namely, nested quadrature rule is turned off.

`gridtol` ϵ
: Convergence criterion for the energy difference between successive nested quadrature iterations. Because of the exponentially convergent Clenshaw–Curtis rule, the quadrature error should be on the order of ϵ^2 once the nested rule converges. The default is `1d-4`.

`sigmafunc` *`parametrization`*
: Switch on σ-functional corrections to RPA. Available parametrizations are `pbe_s1`, `pbe_s2`, `pbe_w1`, `pbe0_s1`, `pbe0_s2`, `pbe0_w1`, `b3-lyp_w1`, `tpss_w1`; the default is `pbe_s1`. Implemented for energy and gradient; see [269].

`riaxk`
: Performs perturbative corrections to RPA such as AXK, ACSOSEX, or bare second-order exchange; see [264]. The implementation is full parallelized using OpenMP.

`acsosex`

By default, the `riaxk` option triggers an RI-AXK calculation. Adding the `acsosex` option triggers an RI-ACSOSEX calculation instead.

`acsox`

trigers an bare second-order exchange correction instead of RI-AXK.

`o4riaxk`

employs the AO based $\mathcal{O}(N^4 \ln N)$ RI-AXK algorithm instead of the default MO based $\mathcal{O}(N^5 \ln N)$ algorithm. The AO based algorithm only becomes more efficient than the MO based algorithm for very large systems with small basis sets.

`axktol` ϵ

basis shell quadruple screening threshold for the AO based RI-AXK algorithm. ϵ is defined according to equation (27) in [264]. The default value of ϵ is $10^{-\texttt{scfconv}}$.

`nohxx`

HF energy calculation is skipped, (HXX = Hartree + eXact (Fock) eXchange). The HXX energy computation in the `rirpa` module is not parallelized for now. For parallel RI-RPA or RI-AXK energy calculations on large systems, it is recommended to use the `nohxx` option for `rirpa`, and then compute the HXX energy separately using the parallel version of `ridft` or `dscf` (by removing the `$dft` keyword block and setting `$scfiterlimit 1` in the control file).

`rpaprof`

Generates profiling output.

`rpagrad`

Switches on the gradients calculation for RI-RPA.

`drimp2`

Computes gradients in the direct RI-MP2 limit.

`iter` n

Number of GKS iterations, n (default is 0).

`ldiis`

Turns on DIIS algorithm for faster convergence of GKS iterations (default is off).

`eigshift` r

Adds a shift of r Hartree to the Kohn–Sham gap to aid in the convergence of difficult small-gap systems (default is 0.0 Hartree).

`output` $filename$

Prints a condensed version of GKS-spRPA relevant output to the $filename$ (default filename is gksrpa.dat).

23.2.20 Keywords for Module `egrad`

`egrad` uses the same general keywords as `escf` and `grad`, see Sections 23.2.15 and 23.2.18.

The state to be optimized is by default the highest excited state specified in `$soes`. Note that only one IRREP can be treated at the same time in contrast to `escf` calculations. When the desired excited state is nearly degenerate with another state of the same symmetry it may be necessary to include higher states in the initial calculation of the excitation energy and vector in order to avoid root flipping. This is accomplished by means of the additional keyword

`$exopt` n

which explicitly enforces that n-th excited state is optimized. n must not be larger than the number of states specified in `$soes`.

`$nacme`

flag to compute Cartesian non-adiabatic coupling vectors between the excited state of interest and the ground state [370]. This option requires the use of `weight derivatives` in section `dft`. It is only implemented for C_1 symmetry.

23.2.21 Keywords for Module `mpgrad`

If an MP2 run is to be performed after the SCF run, the SCF run has to be done with at least

1) density convergence `$denconv 1.d-7`
2) energy convergence `$scfconv 6`

`$maxcor` n

The data group `$maxcor` adjusts the maximum size of core memory (n in MB) which will be allocated during the MP2 run. Recommendation: 3/4 of the actual main memory at most. For further details see subsection 23.2.3.

`$mp2energy`
: Calculation of MP2 gradient is omitted, only MP2 energy is calculated.

`$freeze`
: Freeze orbitals in the calculation of the MP2 correlation energy. For details see Section 23.2.4.

All essential data groups for `mpgrad` may be generated by the preparation tool `mp2prep`, apart from `$maxcor` (see above) these are the following:

`$traloop` n
: specifies the number of loops (or 'passes') over occupied orbitals, n, performed in the `mpgrad` run: the more passes the smaller file space requirements—but CPU time will go up.

```
$mointunit
 type=intermed unit=61       size=0        file=halfint
 type=1111     unit=62       size=0        file=moint#0
 type=1112     unit=63       size=0        file=moint#1
 type=1122     unit=64       size=0        file=moint#j
 type=1212     unit=65       size=0        file=moint#k
 type=1212a    unit=70       size=0        file=moint#a
 type=gamma#1  unit=71       size=0        file=gamma#1
 type=gamma#2  unit=72       size=0        file=gamma#2
 type=1212u    unit=73       size=0        file=moint#u
 type=1112u    unit=74       size=0        file=moint#v
 type=gamma#1u unit=75       size=0        file=gamma#1u
```

The data group `$mointunit` specifies:

- which scratch files are needed,
- where they are located (path name) and
- (after a statistics run, see below) an estimated file size.

`$statistics mpgrad`
: statistics run (estimation of disc space needed) for the adjustment of the file sizes will be performed.

`$mp2pair`
: calculation of MP2 pair correlation energies.

23.2.22 Keywords for Module `ricc2`

Note that beside the keywords listed below the outcome of the `ricc2` program also depends on the settings of most thresholds that influence the integral screening (e.g. `$denconv`, `$scfconv`, `$scftol`) and for the solution of Z vector equation with 4-index integrals (for relaxed properties and gradients) on the settings for integrals storage in semi-direct SCF runs (i.e. `$thime`, `$thize`, `$scfintunit`). For the explanation of these keywords see Section 23.2.10.

`$cbas file=auxbasis`
: Auxiliary basis set for RI approximation.

`$freeze`
: Freeze orbitals in the calculation of correlation and excitation energies. For details see Section 23.2.4.

`$core_excitations`
: Molecular orbitals out which core excitations are allowed. The syntax for this data group is the same as for `$freeze`.

`$printlevel` 1
: Print level. The default value is 1.

`$tmpdir /work/thisjob`
: Specify a directory for large intermediate files (typically three-index coulomb integrals and similar intermediates), which is different from the directory where the `ricc2` program is started.

`$maxcor` n *unit reference*
: The data group `$maxcor` adjusts the maximum size of core memory which will be allocated during the `ricc2` calculation. `$maxcor` has a large influence on computation times. It is recommended to set `$maxcor` to ca. 75–85% of the available physical core memory. For further details see subsection 23.2.3.

`$spectrum` *unit*
: The calculated excitation energies and corresponding oscillator strengths are appended to a file named 'spectrum'. Possible values of `unit` are eV, nm and cm^{-1} or rcm. If no unit is specified, excitation energies are given in a.u.

`$cdspectrum` *unit*
: The calculated excitation energies and corresponding rotatory strengths are appended to a file named 'cdspectrum'. `unit` can have the same values as in `$spectrum`.

```
$laplace
```

```
conv = 5
```

The purpose of this data group is twofold: It activates the Laplace-transformed implementation of SOS-MP2 in the `ricc2` module (if the `sos` option has been specified in `$ricc2`) and it provides the options to specify the technical details for the numerical Laplace-transformation.

`conv`

Threshold for the numerical integration used for the Laplace transformation of orbital energy denominators. The grid points for the numerical integration are determined such that is the remaining root mean squared error (RMSE) of the Laplace transformation is $< 10^{-\mathrm{conv}}$. By default the threshold is set to the value of `conv` given in `$ricc2` (see below).

```
$ricc2
```

```
ccs
cis
mp2     d1diag
cis(d)  energy only
cis(di)
adc(2)
cc2
restart on/off
hard_restart on/off
conv    = 8
oconv   = 7
lindep  = 14
maxiter = 25
mxdiis  = 10
maxred  = 100
shift   = 0.d0
iprint  = 1
fmtprop = f15.8
geoopt model=cc2 state=(a" 2)
scs  cos=1.2d0   css=0.3333d0
sos
intcorr
```

specifies the *ab initio* models (methods) for ground and excited states and the most important parameters and thresholds for the solution of the cluster equations, linear response equations or eigenvalue problems. If more than one model is given, the corresponding calculations are performed successively. Note: The CCS ground state energy is identical with the SCF reference energy, CCS excitation energies are identical to CIS excitation energies. The MP2 results is equivalent to the result from the `rimp2` module. `cis(di)` denotes the iterative CIS(D) variant CIS(D_∞).

`mp2 d1diag`

Request the calculation of the D_1 diagnostic in MP2 energy calculations (ignored in MP2 gradient calculations). Note that the evaluation of the D_1 diagnostic increases the computational costs of the RI-MP2 energy calculation roughly by a factor of 3.

`cis(d) energy only`

If the `energy only` flag is given after the `cis(d)` keyword, it is assumed that only excitation energies are requested. This switches on some shortcuts to avoid the computation of intermediates needed e.g. for the generation of improved start vectors for CC2.

`restart on/off`

If the `restart` flag is not disabled, the program will try to restart from previous solution vectors on file if it finds any. If the `restart` flag is set to `off` no restart from previous solution vectors will be done. The flag is by default set to `on`.

`hard_restart on/off`

If the `hard_restart` flag is set, the program will try to reuse integrals and intermediates from a previous calculation. This requires that the `restart.cc` file has been kept, which contains check sums and some other information needed. The `hard_restart` flag is switched on by default, if the `restart.cc` file is present and no geometry optimization is done (i.e. the option `geoopt` is not set).

`conv` The `conv` parameter gives the convergence threshold for the ground state energy for the iterative coupled-cluster methods as $10^{-\mathrm{conv}}$. The default value is taken from the data group `$deneps`.

`oconv`

The `oconv` parameter gives an additional threshold for the residual of the cluster equations (vector function). If this parameter is given, the iterations for the cluster equations are not stopped before the norm of

the residual is $< 10^{-\text{oconv}}$. By default the threshold is set to `oconv` =`conv`−1, or 10× `deneps` if no input for `conv` is given.

`lindep`

If the norm of a vector is smaller than $10^{-\text{lindep}}$, the vector is assumed to be zero. This threshold is also used to test if a set of vectors is linear dependent. The default threshold is 10^{-14}.

`maxiter`

gives the maximum number of iterations for the solution of the cluster equations, eigenvalue problems or response equations (per default set to 25, or if found in the `control` file, to the value of `$scfiterlimit`).

`mxdiis`

is the maximum number of vectors used in the DIIS procedures for ground state or excitation energies (default: 10).

`maxred`

the maximum dimension of the reduced space in the solution of linear equations (default: 100).

`iprint`

print level, by default set to 1 or (if given) the value of the `$printlevel` data group.

`fmtprop`

Fortran print format used to print several results (in particular one-electron properties and transition moments) to standard output.

`geoopt`

specify wavefunction and electronic state for which a geometry optimization is intended. For this model the gradient will be calculated and the energy and gradient will be written onto the data groups `$energy` and `$grad`. Required for geometry optimizations using the `jobex` script. Note, that in the present version gradients are only available for ground states at the MP2 and CC2 and for excited states at the CC2 level and not for ROHF based open-shell calculations. Not set by default. The default model is CC2, the default electronic state the ground state. To obtain gradients for the lowest excited state (of those included in the excitation energy calculation, but else of *arbitrary* multiplicity and symmetry) the short cut `s1` can be used. `x` is treated as synonym for the ground state.

`scs`

the opposite–spin scaling factor `cos` and the same–spin scaling factor `css`

can be chosen. If `scs` is set without further input, the SCS parameters cos=6/5 and css=1/3 are applied. This keyword can presently only be used in connection with MP2.

`sos`

the SOS parameters cos=1.3 and css=0.0 are applied. This keyword can presently only be used in connection with MP2.

`intcorr`

calculates the second-order corrections to the CCSD(T) energy from the interference-corrected MP2-F12 (INT-MP2-F12) if `$rir12` is switched on. It can be combined either with the `mp2` or the `ccsd(t)` methods. In the latter case, the CCSD(T)-INT-F12 energy is printed. The `intcorr all` keyword writes to the output all pair energies.

`$rir12`

`ansatz`
`r12model`
`comaprox`
`cabs`
`examp`
`r12orb`
`pairenergy`
`corrfac`
`cabsingles`
`f12metric`

`ansatz` *`char`*

char=`1`, `2*` or `2`

The `ansatz` flag determines which ansatz is used to calculate the RI-MP2-F12 ground state energy.

(Ansatz 2 is used if `ansatz` is absent.)

`r12model` *`char`*

char=`A`, `A'` or `B`

The `r12model` flag determines which approximation model is used to calculate the RI-MP2-F12 ground state energy.

(Ansatz B is used if `r12model` is absent.)

`comaprox` *`char`*

char=`F+K` or `T+V`

The comaprox flag determines the method used to approximate the commutator integrals $[T, f_{12}]$.
(Approximation T+V is used if comaprox is absent.)

cabs *char val*

char=svd or cho
The cabs flag determines the method used to orthogonalize the orbitals of the CABS basis. *val* is the threshold below which CABS orbitals are removed from the calculation.
(svd 1.0d-08 is used if cabs is absent.)

examp *char*

char=noinv, fixed or inv with flip or noflip
The examp flag determines which methods are used to determine the F12 amplitudes. For inv the amplitudes are optimized using the orbital-invariant method. For fixed and noinv only the diagonal amplitudes are non-zero and are either predetermined using the coalescence conditions (fixed), or optimized (noinv—not orbital invariant). If *char*=inv, the F12 energy contribution is computed using all three methods. For open-shell calculations noflip supresses the use of spin-flipped geminal functions.
(The fixed flip method is used if examp is absent.)

pairenergy *char*

char=off or on
If *char*=off (default), the print out of the standard and F12 contributions to the pair energies is suppressed. The summary of the RI-MP2-F12 correlation energies is always printed out.

corrfac *char*

char=LCG or R12
The corrfac flag determines which correlation factor is used for the geminal basis. LCG requires the data group $lcg, which contains the information regarding exponents and coefficients of the linear combination of Gaussians.

cabsingles *char*

char=off or on or *
The cabsingles flag determines whether or not the single excitations into the CABS basis are computed. Neglect of the EBC (virtual-CABS) Fock matrix elements is activated by *.
The default is to always compute the CABS singles correction if the

CABS Fock matrix elements are anyway available as a byproduct of the F12 calculation (*i.e.*, for ansatz `2` or r12model `B` or comaprox `F+K`).

`r12orb` *`char`*

char=`hf`, `rohf`, `boys` `pipek` or `arb`

The `r12orb` flag controls which orbitals are used for the F12 geminal basis functions. With `hf` the (semi)-canonical Hartree–Fock orbitals are used (default). For ROHF-based UMP2 calculations `rohf` orbitals can be used, which also implies that the $freeze data group options refer to ROHF rather than semi-canonical orbitals. For closed-shell species, localised orbitals can be used with either the Boys or Pipek-Mezey method. For the non-(semi)-canonical options, the `r12orb` `noinv` F12 energy correction is evaluated using active occupied orbitals transformed to the same basis as the orbitals in the geminal function. The option `arb` uses the semi-canonicalised reference orbitals for the F12 correction, but makes no assumption about whether or not these orbitals are converged HF orbitals and is appropriate for computing F12 corrections using e.g. DFT or Brueckner orbitals (Note, that the data group `$non-canonical MOs` should be set in this case).

`no_f12metric`

`f12metric`

If `no_f12metric` is selected the coulomb metric is used in the density fitting scheme to calculate the four index integrals over the operators f_{12}, $f_{12}g_{12}$, f_{12}^2 and $f_{12}r_{12}$. If `f12metric` is selected the operator's own metric is used. The default for the `ricc2` program is `no_f12metric`, while the `pnoccsd` program can only be used with `f12metric`, where it is therefore the default.

```
$excitations

     irrep=au  multiplicity=1 nexc=4  npre=6  nstart=8 ncore=1
     irrep=bg  multiplicity=3 nexc=2  npre=4  nstart=5
     spectrum states=all operators=diplen,dipvel
     tmexc    istates=all fstates=all operators=diplen,dipvel
     exprop   states=all operators=qudlen
     twophoton states=all operators=(diplen,diplen) freq=0.1d0
     momdrv   states=all operators(diplen,soc) freq=0.0d0
     xgrad    states=(ag{3} 1)
     conv    = 6
```

```
thrdiis = 2
thrpreopt = 3
firstside right
bothsides off
oldnorm   off
maxiter = 25
mxdiis  = 10
maxred  = 100
iprint  = 1
```

In this data group you have to give additional input for calculations on excited states:

`irrep`
: the irreducible representation with the additional suboptions:

  `multiplicity` spin multiplicity (1 for singlet, 3 for triplet); default: singlet, not used for UHF.

  `nexc` the number of excited states to be calculated within this irrep and for this multiplicity (mandatory suboption).

  `npre` the number of roots used in preoptimization steps (default: `npre` = `nexc`).

  `nstart` the number of start vectors generated or read from file (default: `nstart` = `npre`).

  `ncore` the number of core holes for the excited states requested in this input line, possible values are 0 and 1, default: 0.

`spectrum`
: This flag switches on the calculation of oscillator strengths for excited state—ground state transitions. Setting the parameter `states=all` is mandatory for the calculation of transition properties in the present version. The `operators` flag can be followed by a list of operators (see below) for which the transition properties will be calculated. Default is to compute the oscillator strengths for all components of the dipole operator.

`tmexc`
: This flag switches on the calculation of oscillator strengths for excited state—excited state transitions. Specifying the initial and final states

via `istates=all` and `fstates=all` is mandatory for the calculation of transition properties in the present version. The `operators` flag can be followed by a list of operators (see below) for which the transition properties will be calculated. Default is to compute the oscillator strengths for all components of the dipole operator.

`twophoton`

request the calculation of two-photon transition moments between ground and excited states. It recognizes the optional suboptions `states` for specifying the states for which the two-photon transition moments should be computed, `operators` for specifying a list of pairs of one-electron transition operators, and `freq` for a list of frequencies for one of the photons. If these suboptions are specified, two-photon transition moments are computed per default for all requested singlet states, dipole operators (in the length representation) and photon energies which are equal 1/2 of the transition energies.

`momdrv`

request the calculation of derivatives of transition moments between ground and excited states. It recognizes the optional suboptions `states` for specifying the states for which the two-photon transition moments should be computed, `operators` for specifying a list of pairs of one-electron transition operators. For phosphorescence lifetimes `operators` should be set to `(diplen,soc)` and `freq` to `0.0d0`.

`exprop`

requests the calculation of first-order properties for excited states. For the `states` option see `spectrum` option above; for details for the `operators` input see below.

`xgrad`

request calculation of the gradient for the total energy of an excited state. If no state is specified, the gradient will be calculated for the lowest excited state included in the calculation of excitation energies. The simultaneous calculation of gradients for several state is possible.

`conv` convergence threshold for norm of residual vectors in eigenvalue problems is set to $10^{-\texttt{conv}}$. If not given, a default value is used, which is chosen as $\max(10^{-\texttt{conv}}, 10^{-\texttt{oconv}}, 10^{-6})$, where `conv` refers to the values given in the data group `$ricc2`.

`thrpreopt`

convergence threshold used for preoptimization of CC2 eigenvectors is

set to $10^{-\texttt{preopt}}$ (default: 3).

`thrdiis`

threshold ($10^{-\texttt{thrdiis}}$) for residual norm below which DIIS extrapolation is switched on in the modified Davidson algorithm for the non-linear CC2 eigenvalue problem (default: 2).

`firstside right`

With this option one can select whether the right or left eigenvalue problem is solved for excitation energies, or solve first, if both eigenvectors are needed. It can be set to `right` (default) or `left` (for test purposes only).

`bothsides`

The `bothsides` flag (`on`,`off`) enforces the calculation of both, the left and the right eigenvectors (for test purposes only).

`oldnrom`

The `oldnorm` flag (`on`,`off`) switches the program to the old normalization of the eigenvectors and $\%T_1$ and $\%T_2$ diagnostics which were identical with those used in the CC response code of the Dalton program.

`maxiter, mxdiis, maxred, iprint`

Overwrites the values set in `$ricc2` in the routines for the calculations of excitation energies and eigenvectors.

`roothome`

Declares that the requested excitations are specified with their target vectors in the `$roothome` section.

`$response`

```
fop unrelaxed operators=diplen
sop operators=(diplen,diplen) freq=0.077d0,0.089d0
top operators=(diplen,diplen,angmom) freq=0.077d0,-0.077d0
cpp=off gamma=4.5563d-3
gradient
conv = 6
zconv = 6
semicano
nosemicano
thrsemi = 3
```

In this data group you have to give additional input for the calculation of ground state properties and the solution of response equations:

`fop` This flag switches on the calculation of ground state first-order properties (expectation values). The `operators` flag can be followed by a list of operators (see below) for which the first-order properties will be calculated. Default is to compute the components of the dipole and the quadrupole moment. The option `unrelaxed` suppress the calculation of orbital-relaxed first-order properties, which require solution the CPHF-like Z-vector equations. Default is the calculation of unrelaxed and orbital-relaxed first-order properties. The `unrelaxed` option will be ignored, if the calculation of gradients is requested (see `gradient` option below and `geoopt` in data group `$ricc2`).

`sop` requests the calculation of ground state second-order properties as e.g. dipole polarizabilities. The `operators` flag has to be followed by a comma seperated pair of operators. (If more pairs are needed they have to be given with additional `sop` commands.) Default is to compute all symmetry-allowed elements of the dipole-dipole polarizability. With the `freq` flag one can specify a list of frequencies (default is to compute static polarizabilities). The `relaxed` flag switched from the unrelaxed approach, which is used by default, to the orbital-relaxed approach. Note that the orbital-relaxed approach can only be used in the static limit (`freq=0.0d0`). For further restrictions for the computation of second-order properties check Chapter 10.5. In combination with damped response (CPP), the flag `abscpp` can be used to request the operator combinations needed to compute absorption spectra within the length gauge of the dipole operator, and `ecdcpp` to request the operator combinations for ECD spectra within the velocity gauge of the dipole operator. Instead of `freq` the flag `frqscan` can be used to specify an evenly spaced grid of frequency values, e.g. `frqscan=0.1,0.2,0.01`. The three arguments define the start and end point and the step width of the grid.

`top` requests the calculation of ground state third-order properties as e.g. first hyperpolarizability and MCD. The flags for the `sop` keyword also apply to `top`. For further information on the properties available, see 10.6.

`cpp=on/off`
requests the calculation of damped second- or third-order properties with damping factor `gamma` in a.u. . Must be combined with the `sop` or `top` keyword.

`gradient`
require calculation of geometric gradients. In difference to the `geoopt`

keyword in the data group `$ricc2` this can be used to compute gradients for several methods within a loop over models; but gradients and energies will not be written to the data groups `$grad` and `$energy` as needed for geometry optimizations. Note, that in the present version gradients are only available for MP2 and CC2 and only for a closed-shell RHF reference.

`conv` convergence threshold for norm of residual vectors in linear response equations is set to $10^{-\texttt{conv}}$. If not given in the `$response` data group, a default value is used, which is chosen as $\max(10^{-\texttt{conv}}$, $10^{-\texttt{oconv}}, 10^{-6})$, where `conv` and `oconv` refer to the values given in the data group `$ricc2`.

`zconv`

convergence threshold for the norm of the residual vector in the solution of the Z vector equations will be set to $10^{-\texttt{zconv}}$.

`semicano`

use semi-canonical formulation for the calculation of (transition) one-electron densities. Switched on by default. The semi-canonical formulation is usually computationally more efficient than the non-canonical formulation. Exceptions are systems with many nearly degenerate pairs of occupied orbitals, which have to be treated in a non-canonical way anyway. (See also explanation for `thrsemi` below).

`nosemicano`

use non-canonical formulation for the calculation of (transition) one-electron densities. Default is to use the semi-canonical formulation.

`thrsemi`

the threshold for the selection of nearly degenerate pairs of occupied orbitals which (if contributing to the density) have to be treated in a non-canonical fashion will be set to $10^{-\texttt{thrsemi}}$. If set to tight the semi-canonical algorithm will become inefficient, if the threshold is to large the algorithm will become numerical unstable

`zpreopt`

threshold for preoptimizating the so-called Z vector (i.e. the Lagrangian multipliers for orbital coefficients) with a preceding RI-CPHF calculation with the cbas auxiliary basis. The RI-CPHF equations will be converged to a residual error $< 10^{-\texttt{zpreopt}}$. Default is `zpreopt=4`. This preoptimization can reduce significantly the computational costs for the solution of the Z vector equations for large basis sets, in particular if they

contain diffuse basis functions. For calculations on large molecules with small or medium sized basis sets the preoptimization becomes inefficient compared to the large effects of integral screening for the conventional CPHF equations and should be disabled. This option is automatically disabled for `ricc2` calculations based on foregoing RI-JK Hartree-Fock calculation.

`nozpreopt`

disable the preoptimization of the Z vector by a preceding RI-CPHF calculation with the cbas basis set. (Note that the preoptimization is automatically deactivated if the `ricc2` calculation is based on a foregoing RI-JK Hartree-Fock calculation.)

`$roothome`

```
2
1
1   8   9   1.00
2
1   2   3   0.50
1   2   4   0.50
```

Specifies the guess vectors for the excitation calculations. It follows the syntax

```
<Number of roots>
<Number of components for root #1>
<Spin (1 - alpha, 2- beta)> <#occ (from) MO> <#vrt (to) MO> <coeff.>
...
...
<Number of components for root #2>
...
...
```

The MO indices should correspond to increasing SCF orbital energy numbering. The data group `$excitations` with the `roothome` keyword should also be present in the control file.

Common options for keywords in the data groups `$ricc2`, `$response`, and `$excitations`:

`operators=diplen,dipvel`

input of operator labels for first-order properties, transition moments, etc. Currently implemented operators/labels are

`overlap` overlap (charge) operator: the integrals evaluated in the AO basis are $\langle\mu|\nu\rangle$

`diplen` dipole operator in length gauge: $\langle\mu|r_i^O|\nu\rangle$ with $i = x, y, z$; the index O indicates dependency on the origin (for expectation values of charged molecules), which in the present version is fixed to $(0, 0, 0)$ (all three components, individual components can be specified with the labels `xdiplen`, `ydiplen`, `zdiplen`).

`dipvel` dipole operator in velocity gauge: $\langle\mu|\nabla_i|\nu\rangle$
(all three components, individual components can be specified with the labels `xdipvel`, `ydipvel`, `zdipvel`).

`qudlen` quadrupole operator $\langle\mu|r_i^O r_j^O|\nu\rangle$
(all six components, individual components can be specified with the labels `xxqudlen`, `xyqudlen`, `xzqudlen`, `yyqudlen`, `yzqudlen`, `zzqudlen`).
If all six components are present, the program will automatically give the electronic second moment tensor (which involves only the electronic contributions) M_{ij}, the isotropic second moment $\alpha = \frac{1}{3}\mathrm{tr}M$ and the anisotropy

$$\beta = \sqrt{\frac{1}{2}\sum_{i=x}^{z}(M_{ii} - M_{i+1,i+1})^2 + 3\sum_{i=x}^{z} M_{i,i+1}^2}\,.$$

Furthermore the traceless quadrupole moment

$$\Theta_{ij} = \frac{1}{2}\langle 3r_i r_j - r^2\delta_{ij}\rangle$$

(including nuclear contributions) is given.

`angmom` angular momentum $\langle\mu|L_i^O|\nu\rangle$
(all three components, individual components can be specified with the labels `xangmom`, `yangmom`, `zangmom`).

`nef` electronic force on nuclei $\langle\mu|\frac{Z_I r_i^I}{r^I{}^3}|\nu\rangle$, where Z_I is the charge of the nucleus I and r^I is the position vector of the electron relative to the nucleus (all three components for all nuclei: the labels are `xnef001`, `ynef001`, `znef001`, `xnef002`, etc. where the number depends on the order in the `coord` file).

`states=all`

specification of states for which transition moments or first-order properties are to be calculated. The default is `all`, i.e. the calculations will be done for all excited states for which excitation energies have been calculated. Alternatively, one can select a subset of these listed in parentheses, e.g. `states=(ag{3} 1,3-5; b1u{1} 1-3; b2u4)`. This will select the triplet a_g states no. 1, 3, 4, 5 and the singlet b_{1u} states no. 1, 2, 3 and the singlet (which is default if no `{}` is found) b_{2u} state no. 4.

`istates=all fstates=all`

The specification of initial and final states for transition properties between excited states is mandatory. The syntax is analog to the `states` option, i.e. either `all` or a list of of states is required.

`$D2-diagnostic`

Calculate the double-substitution-based diagnostics D_2.

`$cc2_natocc`

Write MP2/CC2 natural occupation numbers and natural orbitals to a file.

`$cgrad 1000`

Calculate the error functional δ_{RI} for the RI approximation of $(ai|bj)$ integrals

$$\delta_{\mathrm{RI}} = \frac{1}{4}\frac{|\langle ab||ij\rangle_{exact} - \langle ab||ij\rangle_{\mathrm{RI}}|^2}{\epsilon_a - \epsilon_i + \epsilon_b - \epsilon_j}$$

and its gradients with respect to exponents and coefficients of the auxiliary basis set as specified in the data group `$cbas`. The results are written to `$egrad` scaled by the factor given with the keyword `$cgrad` and can be used to optimize auxiliary basis sets for RI-MP2 and RI-CC2 calculations (see Section 1.5).

23.2.23 Keywords for Module `ccsdf12`

The `ccsdf12` program uses a subset of the data groups of the `ricc2` program. For F12 calculations it uses in addition the data groups `$rir12`, `$lcg`, `$cabs`, and `$jkbas`. See the respective sections for further details.

The `ccsdf12` program recognizes in the `$ricc2` data group the following options to specify wavefunction methods:

`$ricc2`

```
ccsd
bccd rohf
```

```
mp3
mp4
ccsd(t)
bccd(t) rohf
shift 0.0
core  can/res/bru
no_sc
```

The options `mp3`, `mp4`, and `ccsd` request, respectively, MP3, MP4 and CCSD calculations. The option `ccsd(t)` request a CCSD calculation with the perturbative triples correction, CCSD(T), and as a side result also the CCSD[T] energy will be printed. The option `bccd` requests a BCCD calculation, which is UBCCD for open-shell systems by default. The sub-option `rohf` for BCCD or BCCD(T) requests a ROHF-BCCD calculation, where the alpha and beta Brueckner orbitals are restricted to be the same for the doubly occupied orbitals of an ROHF open shell initial reference state.

`shift`
: sets a level shift (> 0) for the amplitude update in the CC iterations, useful to improve convergence for CCSD or BCCD calculations on e.g. transition metal complexes.

`core`
: specifies the nature of the core orbitals in a frozen core calculation as either the (semi-)canonicalised reference mos (`can`) or the unmodified reference mos (`res`) or Brueckner core orbitals (`bru`) that are relaxed during a BCCD calculation.

`no_sc`
: suppresses the default semi-canonicalisation of a ROHF reference prior to a CCSD or BCCD calculation, such that the output amplitudes and pair energies correspond to the restricted orbitals. Note that a semi-canonicalisation step is always performed immediately before the evaluation of the (T) energy.

In the data group `$rir12` it recognizes `ccsdapprox` as additional option:

`ccsdapprox` *label*
: defines the approximation to CCSD-F12 which will be used if the MP2-F12 calculation is followed by a CCSD or CCSD(T) calculation. The available approximation and corresponding labels are

CCSD(F12)	`ccsd(f12)`
CCSD(F12*)	`ccsd(f12*)`
CCSD[F12]	`ccsd[f12]`
CCSD-F12b	`ccsd-f12b`
$\text{CCSD}(2^*)_{\overline{\text{F12}}}$	`ccsd(2)_/f12`
$\text{CCSD}(2)_{\overline{\text{F12}}}$	`ccsd(2)_/f12`
$\text{CCSD}_{(\text{F12}*)}$	`ccsd_(f12*)`

It is recommended that these approximations are only used in combination with ansatz 2 and the SP approach (i.e. geminal coefficients fixed by the cusp conditions). For CCSD-F12b calculations also the CCSD-F12a energies are calculated as a byproduct. By default a CCSD(F12*) calculation is carried out, because it gives the best cost/accuracy ration among the approximations. $\text{CCSD}_{(\text{F12}*)}$ is the perturbative variant of CCSD(F12*) and should be used when performing a Brueckner calculation.

23.2.24 Keywords for Module `pnoccsd`

Note that beside the keywords listed below the outcome of the `pnoccsd` program also depends on the settings of most thresholds that influence the integral screening (e.g. `$denconv`, `$scfconv`, `$scftol`). For the explanation of these keywords see Section 23.2.10.

`$cbas file=auxbasis`

Auxiliary basis set for RI approximation.

`$freeze`

Freeze orbitals in the calculation of correlation and excitation energies. For details see Section 23.2.4.

`$printlevel` 1

Print level. The default value is 1.

`$tmpdir /work/thisjob`

Specify a directory for large intermediate files (typically three-index coulomb integrals and similar intermediates), which is different from the directory where the program is started.

`$maxcor` n *unit reference*

The data group `$maxcor` adjusts the maximum size of core memory which will be allocated during the `pnoccsd` run. `$maxcor` has a large influence on

computation times. It is recommended to set `$maxcor` to ca. 75–85% of the available physical core memory. For further details see subsection 23.2.3.

`$laplace`

```
conv = 1
```

Only needed for the (T) perturbative triples correction; for other methods only for test purposes. It sets the accuracy for the numerical Laplace-transformation to $10^{-\mathrm{conv}}$ used for the (T) correction and the iterative OSV generation. For the (T0) correction and methods without triples his threshold is only used for the approximate doubles amplitudes from which the OSVs are computed when the iterative algorithm is enabled. The default value is 10^{-1} for the OSV generation and 10^{-2} for the (T) correction. If specified the input value will currently be used in both cases.

`$pnoccsd`

```
mp2
localize ibo
osvmode  paos
paos     tnos
mxrdim   800
tolpno=   1.00E-7
tolosv=   1.00E-9
tolri=    1.21E-3
tolpair=  4.64E-6
opnos on
tolcapno= 1.00E-9   3.16E-8
tolosc=   1.00E-10  3.15E-9
tolopno=  1.00E-9
toloso=   1.00E-10
conv    = 7
oconv   = 3
lindep  = 12
maxiter = 25
mxdiis  = 10
maxred  = 100
scs  cos=1.2d0   css=0.3333d0
sos
```

`mp2` specifies the *ab initio* model (method). The current release version is restriced to MP2, CCSD, CCSD(T0) and CCSD(T). MP2 is the default model.

`localize` specifies the localization method; possible choices are `boys` for Foster-Boys, `pm` for Pipek-Mezey, `ibo` for intrinsic bond orbitals (IBOs) and `none` for canonical (deprecated, only meant for testing) orbitals. Default are (since V7.3) IBOs, which are recommended if e.g. for aromatic systems the seperation of σ- and π-type orbitals is important. (Pipek-Mezey orbitals become very delocal with diffuse basis sets.)

`osvmode` switch between to algorithms for the generation of OSV coefficients. Possible choices are `full` for an $\mathcal{O}(\mathcal{N}^4)$ scaling direct diagonalization (cmp. [371]), `davidson` for an $\mathcal{O}(\mathcal{N}^3)$ scaling iterative diagonalization (cmp. [23]) and `paos` for sub-$\mathcal{O}(\mathcal{N}^2)$ scaling implementation. `paos` is the default choice and recommended but is not currently compatible with gradients, excited states or explicit correlation, for which cases `davidson` is the default.

`mxrdim` the maximal dimension of trial vectors in the iterative OSV generation (`osvmode davidson`). For PNO-MP2 the default is 800. The dimension is bounded by the number of active virtual orbitals and except for small systems a much smaller value as the number of virutals is sufficient. Smaller dimensions increase the performance, but than the iterative scheme might not converge and the program must be restarted with an adjusted dimension. For PNO-MP2-F12 the Default is 4000 since a larger reduced space is required to construct the OSX (cmp. [371]). Usually there is no need to touch this parameter.

`tolpno` specifies the PNO truncation threshold. Default value: 10^{-7}.

`tolosv` specifies the OSV truncation threshold. If no given `tolosv` is set set such that the OSV truncation error is 10 times smaller than the PNO truncation error.

`toltno` specifies the TNO truncation threshold. If no given `toltno` is set set equal to `tolpno`.

`paos` (de)-activates PAO selection at OSV, PNO and TNO steps. Possible choices are `off`, `osv`, `pno`, `tn0` and `tno`, which turn on PAO selection at the specified level and all earlier levels. If PAOs are used and no PAO selection is performed, the domains are constructed by merging the domains one level below (eg pair domains are formed by merging OSV

domains). The default is `tno`, which activates PAO selection at every step and is recommended.

`tolopao` specifies the PAO truncation threshold for OSVs, the default is linked to `tolpno`.

`tolppao` specifies the PAO truncation threshold for PNOs, the default is linked to `tolpno`.

`tolt0pao` specifies the PAO truncation threshold for T0-TNOs, the default is linked to `toltno`.

`toltpao` specifies the PAO truncation threshold for (T)-TNOs, the default is linked to `toltno`.

`tolri` specifies the threshold for selecting orbital and pair-specific auxiliary basis sets for the local RI approximation. If not given `tolri` is set to $10^{7/12} \times \sqrt{\texttt{tolpno}}$, which is the recommended value.

`tolpair` specifies the energy threshold for selecting the significant pairs. If not given `tolpair` is set to $(0.1 \times \texttt{tolpno})^{2/3}$

`toltrip` specifies the energy threshold for selecting the significant triples. If not given `toltrip` is set equal to `tolpair`.

`opnos` enables (`on`) or disables (`off`) for F12 calculations the use of OPNOs for the occupied orbital spaces in the projectors for the three-electron integrals. Default: `on`

`tolcapno` sets for F12 calculations the truncation thresholds for the complementary auxiliary PNOs for the virtual spaces (CAPNOs) in the projectors for the three-electron integrals. If not specified, default values are calculated from the threshold `tolpno`.

`tolosc` sets for F12 calculations the truncation thresholds for the orbital specific complementary auxiliary virtuals from which the CAPNOs are generated. If not specified, default values are chosen as 0.1 $\times$ the `tolcapno` thresholds, which is the recommended choice.

`tolopno` sets for F12 calculations the truncation thresholds for the selection of PNOs for the occupied orbital spaces (OPNOs) in the projectors for the three-electron integrals. If not specified, default values are calculated from the threshold `tolpno`.

`toloso` set for F12 calculations the truncation thresholds for the orbital specific auxiliary occupied orbitals from which the OPNOs are generated. If not specified, default values are chosen as 0.1 $\times$ the `tolopno` threshold, which is the recommended choice.

`conv` The `conv` parameter gives the convergence threshold for the ground state energy as $10^{-\text{conv}}$. The default threshold is 10^{-7}.

`oconv` The `oconv` parameter gives an additional threshold for the residual of the ground state equations as $< 10^{-\text{oconv}}$. The default threshold is 10^{-3}.

`lindep` If the norm of a vector is smaller than $10^{-\text{lindep}}$, the vector is assumed to be zero. This threshold is also used to test if a set of pre-PNOs is linear dependent. The default threshold is 10^{-10} if PAOs are used and 10^{-12} otherwise.

`maxiter` gives the maximum number of iterations for the solution of the cluster equations, eigenvalue problems or response equations (default: 25).

`mxdiis` is the maximum number of vectors used in the DIIS procedures for the ground state equations (default: 10).

`maxred` not used in the current release.

`scs` the opposite–spin scaling factor `cos` and the same–spin scaling factor `css` can be chosen. If `scs` is set without further input, the SCS parameters cos=6/5 and css=1/3 are applied.

`sos` the SOS parameters cos=1.3 and css=0.0 are applied.

`$rir12` for the description of this data group see Sec. 23.2.22.

23.2.25 Keywords for Module `relax`

`$optimize` *options*

define what kind of nonlinear parameters are to be optimized by `relax` and specify some control variables for parameter update.

Available options are:

`internal on/off`

optimize molecular structures in the space of internal coordinates using definitions of internal coordinates given in `$intdef` as described in Section 4.1 (default: `on`).

`redundant on/off`

optimize molecular structures in redundant internal coordinates using definitions of redundant internal coordinates given in `$redundant`. For an optimization in redundant internal coordinates option `internal` has to be switched `on` too, and option `cartesian` has to be switched `off` (default: `on`).

`Cartesian on/off`

optimize molecular structures in the space of (symmetry-distinct) Cartesian coordinates (default: `off`).

`basis on/off` *suboptions*

optimize basis set exponents (default=`off`).

Available suboptions are:

`logarithm`

exponents of uncontracted basis functions will be optimized after conversion into their logarithms (this improves the condition of the approximate force constant matrix obtained by variable metric methods and the behavior of the optimization procedure); scale factors of contracted basis functions will not be affected by the logarithm suboption

`scale`

ALL basis set exponents will be optimized as scale factors (i.e. contracted blocks and single functions will be treated in the same way); if both suboptions (scale and logarithm) are given the logarithms of the scale factors will be optimized

`global on/off`

optimize a global scaling factor for all basis set exponents (default: `off`).

NOTES:
- basis and global have to be used exclusively!
- if `$optimize` has been specified but `$forceapprox` is absent, the option `$forceinit on` is switched on by default.
- specification of the option `$interconversion on` will override `$optimize`!

`$coordinateupdate` *options*

define some variables controlling the update of coordinates.

Available options are:

`dqmax` *real*

maximum allowed total change for update of coordinates. The maximum change of individual coordinate will be limited to $dq_{max}/2$ and the collective change dq will be damped by $dq_{max}/\langle \mathrm{d}q | \mathrm{d}q \rangle$ if $\langle \mathrm{d}q | \mathrm{d}q \rangle > dq_{max}$q (default: `0.3`)

`interpolate on/off`

calculate geometry update by inter/extrapolation of geometries of the

last two cycles (the interpolate option is always switched on by default, but it is only active ANY time if steepest descent update has been chosen, i.e. `$forceupdate method=none`; otherwise it will only be activated if the DIIS update for the geometry is expected to fail)

`statistics on/`*integer*`/off`
: provide a statistics output in each optimization cycle by displaying all (the last *integer*, default setting by `define` is `5`) subsequent coordinates, gradient and energy values (default: `on`).

`$gdiishistory file=`*char*
: the presence of this keyword forces `relax` to provide informational output about the usage of DIIS for the update of the molecular geometry.

`$interconversion` *options* `default=off`
: special input related to the transformation of atomic coordinates between cartesian and internal coordinate spaces (default: `off`).

Available options are:

`maxiter=`*n*
: maximum number of iterations for the iterative conversion procedure internal → cartesian coordinates (default: 25).

`qconv`
: convergence criterion for the coordinate conversion (default: 1.d-10).

`on/off` *options*
: this switch activates special tasks: transform coordinates/gradients/ hessians between spaces of internal/cartesian coordinates using the definitions of internal coordinates given in `$intdef`:

available suboptions are:

`cartesian -> internal coordinate gradient hessian`

`cartesian <- internal coordinate`the direction of the transformation is indicated by the direction of the arrow

Note: specification of `$interconversion on` will override `$optimize`!

`$forceupdate` *method options*
: this data group defines both the method for updating the approximate force constant matrix and some control variables needed for the force constant update.

Options for **method**:

`none`	no update (steepest descent)
`ms` *suboptions*	Murtagh–Sargent update
`dfp` *suboptions*	Davidon–Fletcher–Powell update
`bfgs` *suboptions*	Broyden–Fletcher–Goldfarb–Shanno update
`dfp-bfgs` *suboptions*	combined (bfgs+dfp) update
`schlegel` *suboptions*	Schlegel update
`ahlrichs` *suboptions*	Ahlrichs update (macro option)

suboptions if `method=ms`, `dfp`, `bfgs`, `schlegel`, `ahlrichs`

`numgeo=`*integer*	number of structures used
`maxgeo=`*integer*	maximum number of geometries (= rank of the update procedure, for `ahlrichs` only)
`ingeo=`*integer*	minimum number of geometries needed to start update

if *method*=`ms`, `dfp`, `bfgs`: `maxgeo=2`, `mingeo=1` as default

additional *suboptions* if `method=ahlrichs`

`modus=` *char fmode*	for an explanation see *suboptions* `pulay` given below e.g. `ahlrichs numgeo=7 mingeo=3 maxgeo=4 modus=<g|dg> dynamic`

NOTES: if the macro option `ahlrichs` has been chosen and n=`numgeo`, *ncycl*='number of geometries available'

- if $ncycl < n$: geometry update by inter/extrapolation using the last two geometries
- if $ncycl \geq n$: diagonal update for the hessian by least mean squares fit; pulay update for the geometry (using specified *modus*, *fmode* (see `pulay` below))
- if ($ncycl \geq \max(5, n+3)$ and $\max(|\, g \,|) < 0.01$ and $\bar{g} < 0.001$) or $\mathbf{H}_{ij} \neq 0 \forall\, i \neq j$: diagonal update is replaced by multidimensional BFGS (rank n) update for the hessian

`pulay` *suboptions*

try to find an optimal linear combination of the coordinates of the `numpul` previous optimization cycles as specified by `modus` (see below).

Available suboptions are:

`numpul`=*integer*

number of geometries to be utilized

`maxpul`=*integer*

maximum number of geometries

`minpul`=*integer*

minimum number of geometries needed to start update

`modus`=*char fmode*

char=`<g|g>` or `<g|dq>` or `<dq|dq>` defines the quantity to be minimized (`dq` = internal coordinate change).
fmode specifies the force constants to be used (only if *char*=`<g|dq>` or `<dq|dq>`!)
fmode=`static`: use static force constants
fmode=`dynamic`: use updated force constants

`fail`=*real*

real defines the threshold for the quantity $g * \mathrm{d}q/|g| * |\mathrm{d}q|$ which defines the angle between gradient vector and coordinate change (default: `0.1`). If `pulay` is used in connection with a multidimensional BFGS update for the hessian than the default is *real*=`0.0`. If $\frac{g \cdot \mathrm{d}q}{|g| * |\mathrm{d}q|} > -real$ the pulay update for the geometry is expected to fail and will be ignored. For example:

```
pulay numpul=4 maxpul=4 minpul=3 modus=<dq|dq> static fail=0.2
```

options for `$forceupdate`

`diagonal`

update only the diagonal force constants (update for off-diagonals will be suppressed) (only active if *method*=`ms`, `dfp`, `bfgs`)

`offdamp` *real*

this allows to damp off-diagonal force constants by $1/real$ (compare offreset, which discards off-diagonals completely). Only values > 1.0 will be accepted. This option is active only within one `relax` run and will be disabled automatically by `relax`. This is useful in difficult cases, where

the non-diagonal update has lead to too large non-diagonal elements of the hessian.

`offreset`
: reset off-diagonal force constants to zero. This option will be active for the current optimization cycle only, i.e. it will be removed by `relax` after having discarded off-diagonals!

`allow=`*real*
: optimization cycle specification of a maximum energy change allowed (given in mHartree) which will be accepted using the actual approximate force constant matrix from `$forceapprox`; if this energy change will be exceeded, the force constants will be scaled appropriately
(The default: `0.0` means NO action)

`scale=`*real*
: scaling factor for the input hessian (default: `1.0`).

`threig=`*real*
: lower bound for eigenvalues of the approximate hessian (default: `0.005`); if any eigenvalue drops below `threig`, it will be shifted to a reasonable value defined by:

`reseig=`*real*default: texttt0.005.

`thrbig=`*real*
: upper bound for eigenvalues of the hessian; if any eigenvalue exceeds `thrbig`, it will limited to this value (default: `1000.0`).

`damping=`*real*
: damp the variable metric update for the hessian by 1/(1+ *real*) (default: `0.0`).

`$forceinit` *option*
: specify initialization of the (approximate) force constant matrix.

Available options are:

`on/off`
: this activates or deactivates initialization; if `on` has been set, `relax` will provide an initial force constant matrix as specified by one of the possible initialization options as described below and will store this matrix in data group `$forceapprox`; after initialization `relax` resets `$forceinit` to `off`!

`diag=`*suboptions*
: provide a diagonal force constant matrix with:

available suboptions are:

`real`
: this will lead to an assignment of diagonal elements (default: `1.0`)).

`default`
: this will lead to an assignment of initial force constant diagonals depending on the coordinate type.

`individual`
: Provide individual defined force constant diagonals for
  - internal coordinates (supplied in `$intdef ... fdiag=..`)
  - a global scale factor (`$global ... fdiag=..`)

  This does not work for basis set optimization. For the correct syntax of '`fdiag=..`' see descriptions of `$intdef`, `$global`

`carthess`
: read a cartesian (e.g. analytical) hessian from `$hessian` and use it as a start force constant matrix; if `$optimize internal` has been set: use its transform in internal coordinate space. If large molecules are to be optimized, it may be necessary (large core memory requirements!) to deactivate the numerical evaluation of the derivative of the B-matrix with respect to cartesian coordinates, which is needed to transform $\mathbf{H}(\mathbf{cart}) \rightarrow \mathbf{H}(\mathrm{int})$ exactly by specifying `no dbdx`.

`$last SCF energy change` = *real*

`$last MP2 energy change` = *real*

These keywords depend on the optimization task to be processed and are updated by the corresponding program (i. g. SCF energy).

`$m-matrix` *options*

This data block contains non-default specifications for the m-matrix diagonals. This is of use if some Cartesian atomic coordinates shall be kept fixed during optimization.

Available options are:

integer real real real
: atomic index followed by diagonal elements of the m-matrix for this atom

`$scratch files`

The scratch file `ftmp` allocated by `relax` can be placed anywhere in your file systems instead of the working directory by referencing its path name in this

data group as follows:

```
$scratch files
 relax   ftmp          path/file
```

The first column specifies the program, the second column the scratch file and the third column the path name of the file to be used as scratch file.

Input Data Blocks Needed by RELAX

`$intdef` or `$redundant`

Definitions of internal coordinates and, optionally, values of internal coordinates (`val=...`, given in a.u. or degrees) or force constant diagonal elements (`fdiag=...`).

`$grad`

Cartesian coordinates and gradients calculated in subsequent optimization cycles. Entries are accumulated by one of the gradient programs (`grad`, `mpgrad`, `rimp2`, `ricc2`, `egrad`, etc.).

`$egrad`

Basis set exponents scale factors and their gradients as calculated in subsequent optimization cycles. Entries are accumulated by one of the gradient programs.

`$globgrad`

Global scale factors and gradients as calculated in subsequent optimization cycles. Entries are accumulated by the `grad` or `aoforce` program.

`$corrgrad`

Allows to augment internal SCF gradients by approximate increments obtained from treatments (e.g. correlation or relativistic) on higher level. See the example below.

```
$corrgrad
#  coordinate    increment
     1            0.0600
     8           -0.0850
```

`$forceapprox` *options*

Approximate force constant matrix (as needed for geometry optimization tasks). The storage format may be specified by the
available options:

`format=`*format*

the default format is `format=(8f10.5)`, but other 10-digit f10.x formats (e.g. x=4,6,..) are possible and will be used, after being manually specified within `$forceapprox`. See the example below:

```
$forceapprox  format=(8f10.4)
    0.9124
    -.0108    0.3347
    0.2101    0.0299    1.3347
    0.0076    0.1088    0.0778    0.6515
```

`$hessian (projected)`

this data block contains the analytical Cartesian force constant matrix (with translational and rotational combinations projected out) as output by the `aoforce` program and may be used to supply a high quality force constant matrix `$forceapprox` for geometry optimizations (specifying `$forceinit on carthess`, or `$interconversion cartesian -> internal hessian`).

RELAX Output Data Groups

`$coord`

either updated Cartesian coordinates if a successful coordinate update has been performed, or Cartesian coordinates for input internal coordinates if only a conversion from internal to Cartesian coordinates has been performed.

`$basis`

updated basis set exponents, basis sets contraction coefficients or scaling factors, if `$optimize basis on` has been specified.

`$global`

updated global scaling factor for all basis set exponents, if `$optimize global on` has been specified.

`$forceapprox`

an approximate force constant matrix to be used in quasi-Newton type geometry optimizations; this matrix will be improved in subsequent optimization cycles if one of the variable-metric methods (`$forceupdate`) has been chosen. See 5.3.13 and 23.2.25.

`$forcestatic`

a static (i.e. never updated) approximate force constant matrix to be used in

DIIS-type geometry optimizations. It will be initialized by `relax` specifying: `$forceupdate pulay ...modus=<dq|dq> static`.

The next data groups are output by `relax` (depending on the optimization subject) in order to control the convergence of optimization procedures driven by the shell script `jobex`.

`$maximum norm of cartesian gradient =` *real*

`$maximum norm of internal gradient =` *real*

`$maximum norm of basis set gradient =` *real*

real is the absolute value of the maximum component of the corresponding gradient.

Other Input/Output data used by RELAX

In order to save the effort for conversion of accumulated geometry and gradient data (as needed for the force constant update or the DIIS update of the geometry) to the optimization space, within which the geometry has to be optimized, one may specify the keyword

`$oldgrad`

Then the `relax` program accumulates all subsequent coordinates and gradient as used in optimization in this data group (or a referenced file). This overrides the input of old coordinate and gradient data from data blocks `$grad`, `$egrad`, ... as accumulated by the `grad` program.

degrees

23.2.26 Keywords for Module `statpt`

```
$statpt
  itrvec      0
  hssupdate   bfgs
  hssfreq     0
  keeptmode
  hssidiag    0.5
  radmax      0.3
  radmin      1.0d-4
```

```
  tradius     0.3
  threchange  1.0d-6
  thrmaxdispl 1.0d-3
  thrmaxgrad  1.0d-3
  thrrmsdispl 5.0d-4
  thrrmsgrad  5.0d-4
```

Only non-default values are written in the `control` file except:

```
$statpt
  itrvec 0
```

Following options are available:

`itrvec`
: Index of the Hessian eigenvector to follow for transition structure search (transition vector). Eigenpairs are sorted in ascending order, i.e. with increasing eigenvalues and start with index 1. The eigenpairs corresponding to translations and rotations are shifted to the end. For minimization the value 0 has to be specified.

`hssupdate`
: Method of hessian update. For minimization default is *BFGS*, for TS search default is *Powell* and *none* is for no update.

`hssfreq`
: Recompute the full Hessian every N'th step during a transition state search. The default is zero and the Hessian is read in or computed in the first step only. If the standard Hessian update methods fail, it can help to use this keyword. Warning: This will make the calculation much more time demanding!

`keeptmode`
: Freezing transition vector index.

`hssidiag`
: diagonal hessian elements for diagonal Hessian guess (default: 0.5).

`radmax`
: Maximum allowed value for trust radius (default: 0.3).

`radmin`
: Minimum allowed value for trust radius (default: 1.0d-4).

`tradius`
Initial value for trust radius (default tradius: radmax = 0.3).

Convergence criteria

`threchange` threshold for energy change (default: 1.0d-6).

`thrmaxdispl` threshold for maximal displacement element (default: 1.0d-3).

`thrmaxgrad` threshold for maximal gradient element (default: 1.0d-3).

`thrrmsdispl` threshold for RMS of displacement (RMS = root mean square) (default = 5.0d-4)

`thrrmsgrad` threshold for RMS of gradient (default: 5.0d-4).

All values are in atomic units.

23.2.27 Keywords for Module `moloch`

`$properties` specifies the global tasks for program `moloch` by virtue of the following options

```
$properties
    trace                          off
    moments                        active
    potential                      off
    cowan-griffin                  off
    localization                   off
    population analyses            off
    plot                           off
    firstorder                     off
    fit                            off
```

a missing option or a option followed by the flag `off` will not be taken into account. The flag `active` may be omitted. For most of these options (with the only exceptions of `trace` and `cowan-griffin`), there are additional data groups allowing for more detailed specifications, as explained below.

`moments`

if `moment` is active you need

```
$moments
  0th 1st 2nd 3rd
  point .0 .0 .0
```

to compute the 0th, 1st, 2nd and 3rd moment at the reference point 0 0 0.

`potential`

if `potential` is active you need

```
$points #1
  pot fld fldgrd shld
  point .0 .0 .0
```

to compute the electrostatic potential (`pot`) and/or electrostatic field (`fld`) and/or electrostatic field gradient (`fldgrd`) and/or the zeroth order contribution to the diamagnetic shielding (`shld`) at reference point 0 0 0.

`localization`

if `localization` is active you need `$boys` to perform a boys-localization of orbitals with orbital energies $\geq$ `threshola`d=-2 Hartrees; localize with respect to `locxyz`=x, y and z and write resulting orbitals to `lmofile`= 'lmo'. At the most `sweeps`=10000 orbital rotations are performed. Non-defaults may be specified using the following suboptions:

`lmofile=` *filename*

`locxyz` *dir1 dir2 dir3*

`threshold` *real*

`sweeps` *integer*

`population analyses`

if `population analyses` is active you need

```
$mulliken
 spdf molap netto irpspd irpmol mommul
```

to perform a Mulliken population analysis. The options specify the output data:

`spdf` print molecular orbital contributions to atomic *s*, *p*, *d*,...-populations

`molap` print molecular orbital contributions to overlap populations

`netto` print atomic net populations

`irpspd` print contributions of (irreducible) representations to atomic s,p,d,...-populations

`irpmol` print contributions of (irreducible) representations to overlap populations

or

`$loewdin`

to perform a Löwdin population analysis (options are invalid here). A Löwdin population analysis is based on decomposing $\sqrt{\mathbf{S}}\mathbf{D}\sqrt{\mathbf{S}}$ instead of $\mathbf{DS}$ in case of a Mulliken PA.

or

```
$paboon
 momao maodump maofile=mao all
```

to perform a population analysis based on occupation numbers (the options are not necessary and produce some output data concerning the modified atomic orbitals):

`momao` print MO contributions to occupation numbers of modified atomic orbitals (MAOs).

`maodump` print all MAOs on standard output

`maofile=mao all`
print all MAOs to file `mao`.

This kind of population analysis basically aims at so-called shared electron numbers (SEN) between two or more atoms. By default 2-, 3- and 4-center contributions to the total density are plotted if they are larger than 0.01 electrons. Thresholds may be individually chosen, as well as the possibility to compute SENs for molecular orbitals: `$shared electron numbers`

```
 orbitals
 2-center  threshold =  real
 3-center  threshold =  real
 4-center  threshold =  real
```

Results of this kind of PA depend on the choice of MAOs. By default, all MAOs with eigenvalues of the atomic density matrices larger than 0.1 will be

taken into account. This is a reasonable minimal basis set for most molecules. If modified atomic orbitals shall not be selected according to this criterion, the data group `$mao` selection has to be specified

`$mao selection threshold =`*real*;

The default criterion for the selection of MAOs is the occupation number, for which a global threshold can be specified within the same line as the keyword `$maoselection`. If the global criterion or threshold is not desirable for some atoms, lines of the following syntax have to be added for each atom type of these.

`atom` *symb list* `nmao=`*i* `method=`*meth* `threshold=`*r*

The parameters in this definition have the following meaning:

`symb` atom symbol

`list` list of all atoms for which this definition should apply. The syntax for this list is as usual in `TURBOMOLE`, e.g. `2,3,8-10,12`

`nmao=`*i*
: means number of MAOs to be included

`method=`*meth*
: means selection criterion for MAOs. *meth* can be `occ` (default), `eig`, or `man` *string*, where `occ` denotes selection of MAOs by occupation numbers, `eig` selection by eigenvalues and `man` allows manual selection. In the latter case the string (max. 8 characters) appended to `man` serves as nickname for the definition of the MAOs to be chosen. This nickname is expected to appear as the leftmost word in a line somewhere within data group `$mao selection` and is followed by the indices of the modified atomic orbitals which are to be selected.

`threshold=`*r*
: means the threshold to be applied for the selection criteria `occ` or `eig` (default: 0.1).

Example:

```
$mao selection  threshold= 0.09
   atom c 1,3-5  nmao= 5  method= eig  threshold= 0.1
   atom o 2      nmao= 3  method= man olabel
 olabel  3-5
```

`plot`

option `plot` is out of fashion; to plot quantities on a grid, rather use `$pointval` in connection with `dscf`, `ridft`, `rimp2` or `egrad`, as described below. If nevertheless `plot` is active you need

```
$grid     #1
 mo 4a1g
  origin       .000000       .000000       .000000
 vector1     1.000000       .000000       .000000
 vector2       .000000     1.000000       .000000
 grid1 range    -5.000000     5.000000 points   100
 grid2 range    -5.000000     5.000000 points   100
 outfile = 4a1g
```

to obtain two-dimensional plot data of mo 4a1g (the plane is specified by origin and two vectors with grid range and number of grid points) which is written to file 4a1g. Several plots may be obtained (#1, #2 etc.) at the same time. Use tool 'konto' to visualize the plot.

Note: This is the old-fashioned way to plot MOs and densities. A new—and easier—one is to use `$pointval`, as described below.

`fit`

if `fit` is active you need

```
$vdw_fit
shell     number_of_gridpoints     distance_from_vdW_surface
refine    value_of_potential
```

`shell` Each line refers to all atoms, the line specifies a spherical layer of grid points around the atoms. The number of points and their distance from the van der Waals surface [Bohr] are given (the default is 1.0).

`refine` one line only, smoothing of the layers of grid points around the molecule: the real number is used to define isopotential surfaces on which the points of the layers have to lie.

```
$vdw_radii
element_symbol    van_d_waals_radius
```

One line per element has to be specified, it contains the name of the element and the van der Waals radius in [Bohr].

23.2.28 Keywords for wave function analysis and generation of plotting data

Properties of RHF, UHF and (two-component) GHF wave functions as well as those of SCF+MP2 densities or such from excited state DFT-calculations can be directly analyzed within the respective programs (`dscf`, `ridft`, `mpgrad`, `rimp2` and `egrad`). In case of spin-unrestricted calculations results are given for total densities ($D^\alpha + D^\beta$) and spin densities ($D^\alpha - D^\beta$). If not explicitly noted otherwise, in the following "D" is the SCF density, D(SCF), in case of `dscf` and `ridft`, the MP2-corrected density, D(SCF)+D(MP2), for `mpgrad` and `rimp2` and the entire density of the excited state in case of `egrad`. For modules `dscf` and `ridft` the analysis of properties may be directly started by calling `dscf -proper` (or `ridft -proper`). In case of `mpgrad` and `rimp2` this is possible only, if the MP2 density has already been generated, i.e. after a complete run of `mpgrad` or `rimp2`.

Functionalities of analyses are driven by the following keywords.

`$mvd`

leads to calculation of relativistic corrections for the SCF total density in case of `dscf` and `ridft`, for the SCF+MP2 density in case of `rimp2` and `mpgrad` and for that of the calculated excited state in case of `egrad`. Quantities calculated are expectation values $< p^2 >, < p^4 >$ and the Darwin term ($\sum 1/Z_A * \rho(R_A)$).

`$moments`

yields calculation of electrostatic moments arising from nuclear charges and total electron densities. Also without setting this keyword moments up to quadrupole are calculated, with respect to reference point (0,0,0). Supported extensions:

```
$moments <i>
x1 y1 z1
x2 y2 z2
.
.
```

By integer i; the maximum order of moments is specified, maximum and default is i=3 (octopole moments), real numbers x, y, z allow for the specification of one or more reference points.

`$pop`

drives the options for population analyses. By default a Mulliken PA in the

basis of Cartesian atomic orbitals (CAOs) is performed for the total density ($D^\alpha + D^\beta$) leading to Mulliken (gross) charges and, in case of spin-unrestricted calculations also for the spin density ($D^\alpha - D^\beta$) leading to Mulliken (gross) numbers for unpaired electrons. Besides total numbers also contributions from *s*-, *p*-, ... functions are listed separately.

Two-component wavefunctions (only module `ridft` and only if `$soghf` is set): In two-component calculations instead of S_z $|(S_x, S_y, S_z)|$ is written to the output. Additionally a vector-file named `spinvec.txt` is written, which includes the resulting spin vector for each atom in the system (also the direction).

The following modifications and extensions are supported, if the respective commands are written in the same line as `$pop`:

`lall` Additional information about p_x, p_y, p_z (and analogous for d and f functions) is displayed (lengthy output).

`atoms` *list of atoms*
Contributions are plotted only if arising from atoms selected by list.

`thrpl`=*real*
Contributions smaller than `thrpl` are not displayed (default: 0.01).

`overlap`Mulliken atomic overlap matrix is displayed.

`netto`Mulliken net populations (diagonal elements of Mulliken overlap matrix) are calculated.

`mosum` *list of MOs*
Summed Mulliken contributions for a group of molecular orbitals defined by numbers referring to the numbering obtained e.g. from the tool `eiger`. Note that occupancy of MOs is ignored, i.e. all orbitals are treated as occupied.

`mo` *list of MOs*
Mulliken contributions for single MOs defined by numbers (independent of whether they are occupied or not). If this option is valid, one may additionally set

`dos width`=*real* `points`=*integer*
to calculate a (simulated) density of states by broadening the discrete energy levels with Gaussians and superimposing them. The width of each Gaussian may be set by input (default: 0.01 a.u.). The resolution (number of points) may be chosen automatically (default values are usually sufficient to generate a satisfactory plot) or specified by hand. The output files (`dos` in case of RHF wave

functions, and `dos_a+b`, `dos_a-b`, `dos_alpha`, `dos_beta`; for UHF cases) contain energies (first column), resulting DOS for the respective energy (second column) as well as *s*-, *p*-, *d*-contributions for the respective energy (following columns).

Example:

```
$pop mo 23-33 dos atoms 2,3,7-8
```

leads to Mulliken PA (CAO-basis) for each of the eleven MOs `23-33`, regarding only contributions from atoms `2-3` and `7-8` (results are written to standard output) and generation of file(s) with the respective simulated density of states.

`$pop nbo`

to perform a natural population analyses [335]. The possible options (specified in the same line) are

`idbgl=`*integer*	Debug level.
`AO`	must be provided, the CAO case is not implemented.
`tw=`*real*	Threshold t_w to circumvent numerical difficulties in computing O_w (default: tw=1.d-6).
`idbgl=`*integer*	Debug level (default: idbgl=0).
`ab`	For UHF cases: Print alpha and beta density results.
`short`	Print only natural electron configuration and summary.

Example:

```
$pop nbo AO ab short atoms 1,2,6
```

leads to a natural population analysis (AO-basis) with printing the results of alpha and beta densities (only the electron configuration and the summary) for the atoms 1,2 and 6.

To change the NMB set for atoms, one has to add a $nbonmb-block in the *control* file. Example:

```
$nbonmb
  ni s:4 p:2 d:1
  o  s:2 p:1
```

leads to a NMB set for Ni of 4 s-, 2 p- and 1d-functions and for O of 2 s- and 1 p-functions.

`$pop paboon`

to perform a population analyses based on occupation numbers [336] yielding "shared electron numbers (SENs)" and multicenter contributions. For this method always the total density is used, i.e. the sum of alpha and beta densities in case of UHF, the SCF+MP2-density in case of MP2 and the GHF total density for (two-component-)GHF.

The results of such an analysis may depend on the choice of the number of modified atomic orbitals ("MAOs"), which can be specified by an additional line; without further specification their number is calculated by the method "mix", see below. Note: One should carefully read the information concerning MAOs given in the output before looking at the numbers for atomic charges and shared electron numbers.

`$mao selection` *options*

to specify how MAOs are selected per atom.

Available options are:

a) for the way of sorting MAOs of each atom:

`eig`

MAOs are sorted according to their eigenvalue (those with largest EW finally are chosen). This is the default.

`occ`

MAOs are sorted according to their occupation; note that the number of all occupation is NOT the number of electrons in the system. This option is kept rather for historical reasons.

b) for the determination of the number of MAOs:

`fix`

A fixed number of MAOs is taken for each atom; usually this is the number of shells up to the complete valence shell, e.g. 5 for B-Ne, 6 for Na-Mg, etc. Exceptions are Elements Sc (Y, La), Ti (Zr, Hf), V (Nb, Ta) for which not all five d-shells are included, but only 2, 3 or 4, respectively. This modification leads to better agreement with partial charges calculated by an ESP-fit.

`thr <real>`

All MAOs with an eigenvalue larger than `<real>` are chosen; default is `<real>=0.1`. This and the following two options are not valid in connection with `occ`.

`max`

Maximum of numbers calculated from `fix` and `thr=0.1` is taken.

`mix`

2:1 mixture of `fix` and `thr=0.1`. This choice gives best agreement (statistical) with charges from an ESP-fit. It is the default choice.

c) for additional information about MAOs:

`info`

Eigenvalues and occupations for each MAO are written to output.

`dump`

Entire information about each MAO is written to output. Lengthy.

Further for each atom the number of MAOs and the sorting mode can be specified individually in lines below this keyword. Example:

```
atom 1,3-4 eig 7
```

leads to choice of the 7 MAOs with largest eigenvalue at atoms 1, 3-4.

`$localize`

enables the generation of localized molecular orbitals (LMOs) using Boys localization. By default, all occupied orbitals are included, localized orbitals are written (by default in the AO basis) to file(s) `lmo` in case of RHF and `lalp` and `lbet` in case of UHF orbitals. Note, that LMOs usually break the molecular symmetry; so, even for symmetric cases the AO (not the SAO) basis is used for the output. The localized orbitals are sorted with respect to the corresponding diagonal element of the Fock matrix in the LMO basis. In order to characterize these orbitals, dominant contributions of (canonical) MOs are written to standard output as well as results of a Mulliken PA for each LMO (for plotting of LMOs see option `$pointval`).

The keyword allows for following options (to be written in the same line):

`mo` *list of MOs*

Include only selected MOs (e.g. valence MOs) in localization procedure (numbering as available from `Eiger`).

`sweeps=`*integer*

maximum number of orbital rotations to get LMOs; default value is 10000 (sometimes not enough, in particular for highly delocalised systems).

`thrcont=`*real*

lower threshold for displaying MO and Mulliken contributions (default: 0.1).

`CAO`

LMOs are written to file in the CAO basis (instead of AO).

`pipmez` or `pm` or `pipek-mezey`

Pipek-Mezey localization is used.

`boys`

Foster-Boys localization is used.

`ibo`

Intrinsic Bond Orbitals are used.

`center`

Compute LMO centers, i.e. the diagonal matrix elements of the position operator $\vec{r}_i = \langle i|\vec{r}|i\rangle$

`spread`

Compute LMO spreads defined as $\sigma_i = \sqrt{\langle i|(\vec{r}-\vec{r}_i)^2|i\rangle} = \sqrt{\langle i|\vec{r}^2|i\rangle^2 - \vec{r}_i^2}$, where $\vec{r}_i$ is the LMO center

`spatial_sorting`

sort LMOs such that LMOs with spatially close centers are also close in index space, by default the LMOs are sorted according to the orbital energy expectation values

`timing`

print timings for localization procedures

`$wfn`

triggers the generations of a wfn file. It can be used in dscf/ridft single-point calculations or in ricc2/egrad gradient calculations.

`$esp_fit`

fits point charges at the positions of nuclei to electrostatic potential arising from electric charge distribution (also possible for two-component calculations, for UHF cases also for spin density). For this purpose the ("real") electrostatic potential is calculated at spherical shells of grid points around the atoms. By default, Bragg-Slater radii, r_{BS}, are taken as shell radii, for each atom the number of points is given by $1000 \cdot r_{\mathrm{BS}}^2$, the total number of points is the sum of points for each atom reduced by the number of points of overlapping spheres. Non-default shells (one or more) can be specified as follows:

`$esp_fit`
`shell` *i1 s1*
`shell` *i2 s2*
⋮

Integer numbers i define the number of points for the respective shell, real numbers s constants added to radii (default corresponds to one shell with s=1.0).

A parameterization very close to that by Kollman (U.C. Singh, P.A. Kollman, J. Comput. Chem. 5(2), 129-145 (1984)) may be obtained by

```
$esp_fit kollman
```

Here five shells are placed around each atom with r=1.4*$r_{vdW} + k$, k=0pm, 20pm, 40pm, 60pm, 80pm, and r_{vdW} are the van-der-Waals radii of the atoms.

`$pointval`

drives the calculation of space-dependent molecular quantities at 3D grids, planes, lines or single points. Without further specifications the values of densities are plotted on a three-dimensional grid adapted to the molecular size. Data are deposed to output files (suffix `plt`) that can be visualized directly with the gOpenMol program. In case of RHF-dscf/ridft calculations you get the total density on file `td.plt`, for UHF-dscf/ridft calculations one gets both values for the total density ($D^\alpha + D^\beta$) on `td.plt` and the "spin density" ($D^\alpha - D^\beta$) on `sd.plt`. For mpgrad/rimp2 calculations one gets in the RHF case the total density (D(SCF+MP2)) on `td.plt` and the MP2 contribution on `mp2d.plt` and in the UHF case one obtains the total density ($D^\alpha(SCF + MP2)+D^\beta(SCF+MP2)$) on `td.plt`, the "spin density" ($D^\alpha(SCF+MP2)-D^\beta(SCF + MP2)$) on `td.plt`, and the respective MP2 contributions on files `mp2d.plt` and `mp2sd.plt`. For `egrad` it is similar, just replace in the filenames `mp2` by `e`.

Integration of density (if absolute value greater than `eps`) within a sphere (origin x, y, z, radius r) is performed for

`$pointval integrate` *x y z r eps*

By default the origin is at (0,0,0), the radius is chosen large enough to include the whole 3D box and all contributions are regarded (`eps=0`).

Data different from total and spin densities are generated by following (combinable) settings (to be written in the same line as statement `$pointval`):

`pot` leads to calculation of electrostatic potential arising from electron densities, nuclei and—if present—constant electric fields and point charges. The densities used for calculation of potentials are the same as above; the

respective filenames are generated from those of densities by replacement of the "d" (for density) by a "p" (for potential). By "`pot eonly`" only the electronic contribution to the electrostatic potential is calculated.

`fld` calculation of electric field. Note, that for 3D default output format (`.plt`, see below) only norm is displayed. Densities used are the same as above, filenames are generated from those of densities by replacement of "d" (for density) by "f" (for field).

`mo` *list of MO numbers*

calculation of amplitudes of MOs specified by numbers referring to the numbering obtained e.g. from the tool `eiger` in the same format. The respective filenames are self-explanatory and displayed in the output. Note, that also in MP2 and excited state calculations the HF/DFT ground state orbitals are plotted (and not natural MP2/excited orbitals).

Two-component cases: The density of the spinors specified by numbers referring to the numbering obtained e.g. from the file `EIGS` are visualized. By setting the keyword `minco` also the amplitudes of the spinor-parts are calculated, whose weights (the probability of finding the electron in this part) lie above the threshold.

`lmo` *list of LMO numbers*

calculation of amplitudes of LMOs (previously generated by `$localize`) ordered by the corresponding diagonal element of the Fock matrix in the LMO basis.

`nmo` *list of NMO numbers*

calculation of amplitudes of NMOs (previously generated by
`$natural orbitals file=natural` and
`$natural orbital occupation file=natural`

`elf` calculation of the electron localization function (ELF). [343]

`dens` has to be set, if additionally to one of the above quantities also the density is to be computed.

`xc` calculation of the Kohn-Sham exchange-correlation potential. It is only valid for DFT calculations and it works for all exchange-correlation functionals, including LHF. Note that for hybrid functionals, only the Kohn-Sham part of the potential will be computed (the HF part is a non-local-operator and can't be plotted). For GGA functional the full potential will be computed (local and non-local terms)

For line plots the output file is `tx.vec` . For UHF calculations the output files are `tx.vec` (alpha-spin potential) and `sx.vec` (beta-spin potential).

For a line plot the file has three columns: 1: total potential 2: local term (or Slater-potential for LHF) 3: non-local terms or Correction term for LHF

Output formats may be specified by e.g. `fmt=xyz` if written to the same line as `$pointval`. Supported are:

`xyz` in case of scalars (density, (L)MO amplitudes, electrostatic potential) this format is: $(x, y, z, f(x, y, z))$. In case of vectors components of the vector and its norm are displayed. This format is valid for all types of grid (3D, plane, line, points, see below), it is the default format in case of calculation of values at single points. Output file suffix is `.xyz`.

`plt` only for 3D, default in this case. Data are written to binary files that can be directly read by gOpenMol. Note, that this output is restricted to scalar quantities; thus in case of vectors (E-field) only the norm is plotted. Output file suffix is `.plt`.

`map` only for 3D. Data are written to ASCII files that can be imported by e.g. gOpenMol. Note, that this output is restricted to scalar quantities; thus in case of vectors (E-field) only the norm is plotted. Output file suffix is `.map`.

`txt` a format compatible with gOpenMol for visualization of vectors v. The format is x, y, z, v_x, v_y, v_z.

`vec` for planes and lines (default in these cases). In case of a line specified by $\alpha \cdot \vec{v}$ (see below) output is $\alpha, f(x, y, z)$ for scalars, for vectors components and norm are displayed. vectors. Analogously, in case of planes it is $\alpha, \beta, f(x, y, z)$. The output (file suffix `.vec`) may be visualized by plotting programs suited for two-dimensional plots. A command file (termed `gnuset`) to get a contour plot by gnuplot is automatically generated.

`cub` only for 3D, writes out data in Cube format which can be imported by many external visualization programs.

For 3D grids non-default boundarys, basis vector directions, origin and resolutions may be specified as follows:

```
$pointval
grid1 vector 0 3 0 range -2,2 points 200
grid2 vector 0 0 -7 range -1,4 points 300
grid3 vector 1 0 0 range -1,1 points 300
origin 1 1 1
```

Grid vectors (automatically normalized) now are $(0,1,0)$,$(0,0,-1)$,$(1,0,0)$, the grid is centered at $(1,1,1)$, and e.g. for the first direction 200 points are distributed between -2 and 2.

In particular for 2D plots it is often useful to shift and rotate the grid plane such that some particular atoms are located in the plot plane. This can be achieved with the `orient` option, which accepts as additional input a list of atoms, e.g. the atoms 4-5 and 9 in the following example:

```
$pointval
  grid1 vector 0 3 0 range -2,2 points 200
  grid2 vector 0 0 -7 range -1,4 points 300
  orient 4-5,9
  rotate 25 3
```

This will shift the origin of the plot into the center of mass of the specified set of atoms and align the grid axes with their principal axes: If two or more atoms are specified which lie on one line grid axis 1 is rotated into this line. If the specified subset consists of three atoms located in a plane the grid axis 1 and 2 are rotated into this plane. If more than three atoms are specified the grid axis 1 and 2 are rotated into a plane which fitted to the position of all specified atoms. With the `rotate` option the grid can be rotated around a grid axis. It accepts as input the rotation angle the index (1/2/3) of the grid axis around which the grid will be rotated.

Grids of lower dimensionality may be specified (in the same line as `$pointval`) by typing either `geo=plane` or `geo=line` or `geo=point` The way to use is best explained by some examples:

```
$pointval geo=plane
grid1 vector 0 1 0 range -2,2 points 200
grid2 vector 0 0 1 range -1,4 points 300
origin 1 1 1
```

Values are calculated at a plane spanned by vectors (0,1,0) and (0,0,1) centered at (1,1,1).

```
$pointval geo=line
grid1 vector 0 1 0 range -2,2 points 50
origin 0 0 1
```

Values are calculated at a line in direction (0,1,0) centered at (0,0,1). Output format as above.

```
$pointval geo=point
7 5 3
0 0 7
```

Values are calculated at the two points $(7.0, 5.0, 3.0)$ and $(0.0, 0.0, 7.0)$.

Plane-averaged density can be computed by

```
$pointval dens averagez fmt=vec
grid1 vector 1 0 0 range -10,10 points 100
grid2 vector 0 1 0 range -10,10 points 100
grid3 vector 0 0 1 range -20,20 points 200
origin 0 0 0
```

The generated file `td.vec` will contain the quantity

$$\rho(z) = \int\int \mathrm{d}x\mathrm{d}y\, \rho(x, y, z) \tag{23.3}$$

23.2.29 Keywords for Module `frog`

The *ab initio* molecular dynamics (MD) program `frog` needs a command file named `mdmaster`. The interactive `Mdprep` program manages the generation of `mdmaster` and associated files. It is always a good idea to let `Mdprep` check over `mdmaster` before starting an MD run. `Mdprep` has online-help for all menus.

In this implementation of *ab initio* MD, time is divided into steps of equal duration Δt. Every step, the energy and its gradient are calculated and these are used by the `frog` to work out the new coordinates for the next step along the dynamical trajectory. Both the accuracy of the trajectory and the total computation time thus depend crucially on the time step chosen in `Mdprep`. A bad choice of timestep will result in integration errors and cause fluctuations and drift in the total energy. As a general rule of thumb, a timestep Δt should be chosen which is no longer than one tenth of the shortest vibrational period of the system to be simulated.

Note that `Mdprep` will transform velocities so that the total linear and angular momentum is zero. (Actually, for the Leapfrog algorithm, initial velocities are $\Delta t/2$ before the starting time).

The following keywords are vital for `frog`:

`$nsteps` 75

Number of MD time steps to be carried out. `$nsteps` is decreased by 1 every time `frog` is run and JOBEX -md stops when `$nsteps` reaches 0.

`$natoms` 9

Number of atoms in system.

`$current file=mdlog.aa`

The file containing the current position, velocity, time and timestep, that is, the **input** configuration. During an MD run the `$current` information is generally kept at the end of the `$log` file.

`$log file=mdlog.ZZ`

The file to which the trajectory should be logged, i.e. the **output**: t=time (a.u.);
atomic positions x,y,z (Bohr) and symbols at t;
timestep (au) Δt;
atomic symbols and velocities x,y,z (au) at $t - (\Delta t/2)$;
kinetic energy (H) interpolated at t, *ab initio* potential energy (H) calculated at t, and pressure recorded at the barrier surface (atomic units, 1 au = 29.421 TPa) during the corresponding timestep;
ab initio potential energy gradients x,y,z (H/Bohr) at t.
This file can be manipulated with `log2?` tools after the MD run (Section 1.5).

`$turbomole file=control`

Where to look for `TURBOMOLE` keywords `$grad` etc.

`$md_status`

The status of the MD run is a record of the action carried out during the previous MD step, along with the duration of that step. The format matches that of `$md_action` below.

Canonical dynamics is supported using the Nosé-Hoover thermostat. This option can be enabled in `Mdprep` or by the following syntax:

```
$md_status
  canonical T=500 t=100
  from t= -25.0000000000       until t=  0.00000000000
```

Here, T specifies the temperature of the thermostat in K (500 K in the example) and t specifies the thermostat relaxation time in a.u. (100 a.u. in the example). It is advisable to choose the thermostat relaxation 2-10 times larger

than the time step. Note that user-defined actions are presently not supported in canonical dynamics mode.

These are optional keywords:

`$seed -123`
: Integer random number seed

`$title`
: Arbitrary title

```
$log_history
 100                mdlog.P
 71                 mdlog.Q

$ke_control
  length        50
  response      1
```

To determine the trends in kinetic energy and total energy (average values and overall drifts) it is necessary to read the history of energy statistics over the recent MD steps. The number of MD steps recorded so far in each log file are therefore kept in the `$log_history` entry: this is updated by the program each step. The length of records needed for reliable statistics and the number of steps over which changes are made to kinetic energy (`response`) are specified in `$ke_control`.

```
$barrier angstroms
  type        elps
  limits      5.0 10.0 7.5
  constant    2.0
  springlen   1.0
  temperature 300.0
```

`$barrier` specifies a virtual cavity for simulating condensed phases. The optional flag, `angstroms`, can be used to indicate that data will be entered in Ångstrøms rather than Bohr.

`type`
: can be one of `orth`, `elps`, or `none`, for orthorhombic, ellipsoidal, or no barrier (the default) respectively.

`limits`
: are the +x,y,z sizes of the cavity. In this case, an ellipsoid with a major axis of 20 Å along y, semi-major of 15 Å on z, and minor of 10 Å on x.

`constant`
: is the Hooke's Law force constant in atomic units of force (H/Bohr) per length unit. Here, it is 2.0 H/Bohr/Ångstrøm, a bastard combination of units.

`springlen`
: is the effective limit to the restorative force of the barrier. For this system, an atom at 5 Å into the barrier will feel the same force as at 1.0 Å.

`temperature`
: denotes the temperature of the cavity walls in Kelvin. If the system quasi-temperature is below this setpoint, particles will be accelerated on their return to the interior. Alternately, they will be retarded if the system is too warm. A `temperature` of 0.0 K will turn off wall temperature control, returning molecules to the system with the same momentum as when they encountered the barrier.

```
$constraints angstroms
  tolerance   0.05
  adjpercyc   0.25
  type H O 0.9 1.2
  type F C 0.0 1.7
  type H C -1.0 1.2
  2 1 0.0
  3 1 1.54
  4 1 -1.0
```

`$constraints`
: specifies and/or automatically generates atomic distance constraints. The optional flag, `angstroms`, can be used to indicate that data will be entered in Ångstrøms rather than Bohr.

`tolerance`
: is the convergence criterion for application of constraints. All distances must be within +/- `tolerance` of the specified constraint. Additionally,

the RMS deviation of all constrained distances must be below 2/3 of `tolerance`.

`adjpercyc`
: is the fraction of the total distance correction to be applied on each constraint iteration.

`type X A` *const rmax*
: commands `frog` to find the closest `A` atom to each atom `X` that is closer than `rmax` and apply `const`. The first `type` line above examines each `H` atom and looks for any `O` atoms within 1.2 Å. The shortest distance, if any, is then fixed at 0.9 Å. Similarly, the second `type` line binds each `F` to the closest `C` within 1.7 Å, but since `const`=0.0, that distance is fixed at the current value. The third `type` line attaches `H` atoms to the appropriate nearby `C`, but at the current average `H-C` distance multiplied by the absolute value of `const`.

Explicitly specified constraints are listed by atom index and supercede auto-generated constraints. A positive third number fixes the constraint at that value, while zero fixes the constraint at the current distance, and a negative number unsets the constraint.

The output of `frog` contains the full list of constrained atom pairs and their current constraints in explicit format.

User-defined instructions allow the user to tell `frog` to change some aspect of the MD run at some point in time `t=`*real number*. The same format is used for `$md_status` above. Here is an example:

```
$md_action
     fix total energy from t=2000.0
     anneal from t=2500.0
     free from t=3000.0
```

In this example, starting from the time 2000.0 a.u., velocities are to be scaled every step to keep average total energy constant. Then, from 2500.0 a.u., gradual cooling at the default rate (annealing) is to occur until the time 3000.0 a.u., when free Newtonian dynamics will resume.

Here are all the possible instructions:

```
$md_action
```

```
fix temperature from t=<real>
fix total energy from t=<real>
```

These commands cause velocities to be scaled so as to keep the average kinetic energy (i.e. quasi-temperature) or the average total energy approximately constant. This is only possible once enough information about run history is available to give reliable statistics. (Keywords `$log_history`, `$ke_control`).

`$md_action`

```
set temperature at t=<real> to x=<real> K
set total energy at t=<real> to x=<real> H
set kinetic energy at t=<real> to x=<real> H
set position file=<filename> at t=<real>
set velocity file=<filename> at t=<real>
set velocity at t=<real> random
set velocity at t=<real> zero
```

At some time during the *ab initio* MD run the user can specify a new value for one of the dynamical variables. The old value is discarded. Single values are given by `x=`*real number*. Vectors must be read in `frog` format from `file=`*file*.

`$md_action`

```
anneal from t=<real>
anneal from t=<real> x=<real>
quench from t=<real>
quench from t=<real> x=<real> file=<file>
relax at t=<real>
```

In Simulated Annealing MD, the temperature of a run is lowered so as to find minimum-energy structures. Temperature may be lowered gradually by a small factor each step (`anneal`; default factor 0.905 over 100 steps) or lowered rapidly by reversing all uphill motion (`quench`; default factor -0.8 each step). The cooling factors may be changed from the default using `x=`. Another option allows the quenching part of the run to be logged to a separate file. Alternatively, a standard non-dynamical geometry optimization can be carried out in a subdirectory (`relax`).

`$md_action`

```
free from t=<real>
```

Finally, this instruction turns off any previous action and resumes free dynamics. This is the default status of an MD run.

`$surface_hopping`

This keyword allows to carry out Tully-type *fewest switches surface hopping* (SH) [372]. This option is only available in combination with TDDFT. For the TDDFT surface hopping see Tapavicza et al. 2007 [373]; for the current implementation see Tapavicza et al. 2011 [374]. In the current implementation the surface hopping algorithm only allows switches between the first excited singlet state and the ground state. However, total energies of higher excited states can be computed during the MD simulation. The proper functioning of SH has only been tested for the option

```
$md_action
 fix total energy    from t=  0.00000000000
```

To carry out SH dynamics simulations, the keyword `$surface_hopping` has to be added to the control and mdmaster file. In addition several keywords are required in the control file:

```
$currentstate
  1
```

`$currentstate`

keyword needed to ensure dynamics starting in S_1

`$nacme`

needed to compute non-adiabatic couplings; this keyword requires the use of `weight derivatives` in section `dft`

The excitations, excited state gradients and nonadiabatic coupling vectors are generated from a TDDFT calculation and stores by `egrad`. The gradients and the nonadiabatic vectors are stored under the data group and the excitations under . These data groups are automatically generated in an MD simulation.

`$sh_coeffs file=sh_coeffs`

collects amplitudes of the adiabatic states along the trajectory . This keyword is automatically generated in the first MD step, with the entire electronic population on the current active state. However, the user may specify the initial coefficients to be something different by manually creating the file.

`$population_el`

stores the electronic populations at each time step. This is generated by `frog` and the user need not specify.

`$prob_fssh`

stores the probability of hopping at each time step. This is generated by `frog` and the user need not specify.

`$active`

stores the active state for surface hopping at each time step. This is generated by `frog` and the user need not specify if is specified.

Special caution has to be taken to control problems related to conical intersections [375,376]. At geometries where conical intersections between the ground and excited state are present, DFT often exhibits singlet instabilities, which leads to imaginary excitation energies in linear response TDDFT; in this case the MD run is terminated. This problem can be circumvented by the use of the Tamm-Dancoff approximation (TDA) to TDDFT (see 8). In addition an optional keyword for the md_master file can be used:

`$gap_threshold <real>`

enforces a switch to the ground state in case the S_1-S_0 energy gap drops below `<real>` eV. As default a switch to S_0 is enforced if the S_1 TDDFT-TDA excitation energy becomes negative.

Often times if a switch is enforced due to a negative TDA excitation energy the potential energy surface is discontinuous and limited numerical precision of the nuclear forces may lead to a loss of total energy conservation. In this case the nuclear velocities are rescaled to obtain a conserved total energy.

23.2.30 Keywords for Module `mpshift`

In order to control the program execution, you can use the following keywords within the `control` file:

`$csmp2`

Switches on the calculation of the MP2 NMR shieldings. The required SCF shielding step will be performed in the same run. This flag will be set by the script `mp2prep`.

`$traloop` n

specifies the number of loops (or 'passes') over occupied orbitals n when doing an MP2 calculation: the more passes the smaller file space requirements—but CPU time will go up. This flag will be set by the script `mp2prep`.

`$mointunit`

Scratch file settings for an MP2 calculation. Please refer to Section 23.2.21 for a description of the syntax. This flag will be set by the script `mp2prep`.

`$shiftconv` *integer*

Residuum convergence threshold for the solution of the CPHF equations. Default is 7.

`$maxcor`

Controls the memory usage for the batching part of the Davidson solver.

`$oldcphf`

Uses the old CPHF solver, which was the default up to Version 7.4, the calculation can be restarted at any stage with this solver.

`$csconv` *real*

Sets the convergence threshold for the shielding constant of succeeding CPHF iterations. The unit is ppm and the default value is 0.01. Only used with the old solver.

`$csconvatom` *integer*

This selects the atom number for convergence check after each cphf iteration. After this convergence is reached all other atoms are checked, too (default: 1). Only used with the old solver.

`$thime`, `$thize`, `$scftol`, `$scfintunit`, `$scfmo`

have the save meaning as in `dscf` (see Section 23.2.10);
Since mpshift works 'semi-direct' it uses the same integral storage.

`$scratch files`

The scratch files allocated by mpshift can be placed anywhere in your file systems instead of the working directory by referencing their pathnames in this data group. All possible scratch files are listed in the following example:

```
$scratch files
    mpshift   csssmat        path1/file1
    mpshift   cshsmat        path2/file2
    mpshift   csdgsmat       path3/file3
    mpshift   csusmat        path4/file4
    mpshift   dens           path5/file5
    mpshift   fock           path6/file6
    mpshift   dfock          path7/file7
```

```
mpshift   idvds1          path8/file8
mpshift   idvds2          path9/file9
mpshift   idvds3          path10/file10
mpshift   jdvds1          path11/file11
mpshift   jdvds2          path12/file12
mpshift   jdvds3          path13/file13
mpshift   cshmmat         path14/file14
```

`$trast` , `$trand` *traloop-number*

stands for traloop start and traloop end. Each loop or pass in MP2 chemical shift calculations can be done individually by providing the keywords `$trast` and `$trand`. This can be used to do a simple parallelization of the run:
Create separate inputs for each traloop. Add

```
$trast <number>
$trand <number>
```

in the `control` files, *number* goes from 1 to the number of `$traloops`. Each calculation will create a restart file called `restart.mpshift`. To collect all steps and to do the remaining work, copy all restart files to one directory and rename them to `restart.mpshift.`*number*, add `$trast -1` and `$trand` *number_ of_ traloops* to the `control` file and start `mpshift`.

`$vcd`

$U_{ai}^{B_\beta}$ will be written to file for a subsequent calculation of VCD rotational strengths by `aoforce`.

`$nucsel`

Selection of nuclei whose NMR shielding tensor has to be computed. The nuclei can be specified by the element symbol (i.e. `$nucsel` "c","h") or by the corresponding number as set by the coordinates (i.e. `$nucsel` 1,3,5-8). Without setting `$nucsel`, the shielding tensors of all nuclei are calculated. You can reuse this keyword for nuclear spin-spin coupling constants.

`$csmaxiter`

Set the maximum number of CPHF iterations, default 30.

`$nics`

Calculation of nucleus-independent chemical shifts. The coordinates have to be listed just as the molecular coordinates of the molecule. A sample input for NH_3 would look like this:

```
$nics
    0.00000000000000    0.00000000000000    0.12917062194577
    1.75612466223131    0.00000000000000   -0.59831613718919
   -0.87806233111566    1.52084856970468   -0.59831613718919
   -0.87806233111566   -1.52084856970468   -0.59831613718919
    0.00000000000000    0.00000000000000    0.00000000000000
$coord
    0.00000000000000      0.00000000000000      0.12917062194577      n
    1.75612466223131      0.00000000000000     -0.59831613718919      h
   -0.87806233111566      1.52084856970468     -0.59831613718919      h
   -0.87806233111566     -1.52084856970468     -0.59831613718919      h
```

`$gimic`

Prepare input for a GIMIC calcuation, see https://github.com/qmcurrents/gimic/. The density and the perturbed density matrix are written to disk. GIMIC allows to calculate the magnetically induced current density to estimate the degree of aromaticity and to study electron delocalization pathways.

`$pnmr` *options*

This keyword allows for the specification of options in open-shell calculations. By default, Fermi contact, spin dipole, and the paramagnetic spin–orbit interactions are included in the hyperfine part of the shielding tensor and the contributions are evaluated at 298K. The g-tensor is also calculated based on spin–orbit perturbation theory. In X2C, the proper picture-change correction is applied throughout. For systems with a spin $S > 1/2$, the ZFS tensor can be calculated as well, but is not used for the evaluation of the shielding tensor. Available options are

`fc`	Only Fermi contact term will be included in the hyperfine shielding
`sd`	Only spin dipole term will be included in the hyperfine shielding
`psoso`	Only spin–orbital paramagnetic spin–orbit term will be included in the hyperfine shielding
`none`	All hyperfine contributions will be excluded and only the orbital contribution to the shielding tensor will be calculated. This option should be used if you are only interested in the ring current or the magnetically induced current density (GIMIC).
`hfc-only`	Only the hyperfine coupling constant will be calculated, the orbital contribution to the shielding tensor is skipped and the g-tensor is

not calculated.

`g-only` Only the g-tensor will be calculated, the orbital contribution to the shielding tensor is skipped and the hyperfine coupling tensor is not calculated.

`gelec` Use isotropic g-factor of the free electron instead of calculating the g-tensor.

`g=`*real* Use a specific isotropic g-factor instead of calculating the g-tensor.

`zfs` In addition to the default settings, the ZFS tensor is calculated, but is NOT considered for the calculation of the paramagnetic shift. The UNO approach is used for the SD contribution to the ZFS tensor.

`zfs-only` Only the ZFS tensor will be calculated, the orbital contribution is skipped, as well as the HFC and g-tensor.

`zfs-dip` Only the SD contribution to the ZFS tensor will be calculated.

`zfs-soc` Only the SO contribution to the ZFS tensor will be calculated.

`direct` Only in combination with `zfs`, `zfs-only` or `zfs-dip`. The direct approach for the SD contribution to the ZFS tensor is used.

The options fc, sd, and psoso can be combined with the option temp= *integer/real*. For example,

```
$pnmr fc temp=100
```

includes only the Fermi contact term, evaluated at 100K, in the hyperfine shielding. Note that only temperatures above 10K are allowed. The terms included in the hyperfine part of the shielding and the temperature at which they were evaluated are stated in the output of `mpshift`. We strongly recommend to use the current-dependent generalization for meta-GGAs and local hybrid functionals (`$curswitchengage`), especially for open-shell calculations [125, 280, 281].

23.2.31 Keywords for Parallel Runs

On all systems the parallel input preparation is done automatically. Details for the parallel installation are given in Section 3.4.1. The following keywords are necessary for all parallel runs:

`$parallel_platform` *architecture*

Currently the following parallel platforms are supported:

`SMP` for systems with very fast communication; all CPUs are used for the linear algebra part. Synonyms for `SMP` are:
`HP V-Class`, `SP3-SMP` and `HP S/X-Class`

`MPP` for systems with fast communication like Fast-Ethernet, the number of CPUs that will be taken for linear algebra part depends on the size of the matrices. Synonyms for `MPP` are:
`SP3` and `linuxcluster`

`cluster` for systems with slow communication, the linear algebra part will be done on one single node. Synonyms for `cluster` are:
`HP Cluster` and every platform that is not known by `TURBOMOLE`

`SGI` similar to `SMP`, but here the server task is treated differently: the MPI implementation on the SGIs would cause this task to request too much CPU time otherwise.

If you want to run `mpgrad`, `$traloop` has to be equal to or a multiple of the number of parallel workers.

For very large parallel runs it may be impossible to allocate the scratch files in the working directory. In this case the `$scratch files` option can be specified; an example for a `dscf` run is given below. The scratch directory must be accessible from all nodes.

```
$scratch files
   dscf  dens       /home/dfs/cd00/cd03_dens
   dscf  fock       /home/dfs/cd00/cd03_fock
   dscf  dfock      /home/dfs/cd00/cd03_dfock
   dscf  ddens      /home/dfs/cd00/cd03_ddens
   dscf  xsv        /home/dfs/cd00/cd03_xsv
   dscf  pulay      /home/dfs/cd00/cd03_pulay
   dscf  statistics /home/dfs/cd00/cd03_statistics
   dscf  errvec     /home/dfs/cd00/cd03_errvec
   dscf  oldfock    /home/dfs/cd00/cd03_oldfock
   dscf  oneint     /home/dfs/cd00/cd03_oneint
```

For all programs employing density functional theory (DFT) (i.e. dscf/gradand ridft/rdgrad) $pardft can be specified:

```
$pardft
       tasksize=1000
       memdiv=0
```

The tasksize is the approximate number of points in one DFT task (default: 1000) and memdiv says whether the nodes are dedicated exclusively to your job (memdiv=1) or not (default: memdiv=0).

For dscf and grad runs you need a parallel statistics file which has to be generated in advance. The filename is specified with

```
$2e-ints_shell_statistics    file=DSCF-par-stat
```

or

```
$2e-ints'_shell_statistics    file=GRAD-par-stat
```

respectively.

The statistics files have to be generated with a single node dscf or grad run. For a dscf statistics run one uses the keywords:

```
$statistics  dscf parallel
$2e-ints_shell_statistics    file=DSCF-par-stat
$parallel_parameters
       maxtask=400
       maxdisk=0
       dynamic_fraction=0.300000
```

and for a grad statistics run:

```
$statistics  grad parallel
$2e-ints'_shell_statistics    file=GRAD-par-stat
$parallel_parameters
       maxtask=400
```

maxtask is the maximum number of two-electron integral tasks,
maxdisk defines the maximum task size with respect to mass storage (MBytes) and dynamic_fraction is the fraction of two-electron integral tasks which will be allocated dynamically.

In the parallel version of ridft, the first client reads in the keyword $ricore from the control file and uses the given memory for the additional RI matrices and for

RI-integral storage. All other clients use the same amount of memory as the first client does, although they do not need to store any of those matrices. This leads to a better usage of the available memory per node. But in the case of a big number of auxiliary basis functions, the RI matrices may become bigger than the specified `$ricore` and all clients will use as much memory as those matrices would allocate even if that amount is much larger than the given memory. To omit this behavior one can use:

`$ricore_slave` *integer*

specifying the number of MBs that shall be used on each client.

For parallel `jobex` runs one has to specify all the parallel keywords needed for the different parts of the geometry optimization, i.e. those for `dscf` and `grad`, or those for `ridft` and `rdgrad`, or those for `dscf` and `mpgrad`.

Chapter 24

Sample `control` files

24.1 Introduction

The file `control` is the input file for `TURBOMOLE` which directly or by cross references provides the information necessary for all kinds of runs and tasks. `control` is usually generated by `define`, the input generator. The following sample `control` files cover a variety of methods and systems. The keywords themselves are explained in Chapter 23.

24.2 NH_3 Input for a RHF Calculation

Main File control

```
$title
NH3 c3v SVP
$symmetry c3v
$coord    file=coord
$atoms
   basis =def-SVP
$pople   AO
$basis    file=basis
$scfmo    file=mos
$closed shells
 a1      1-3                                    ( 2 )
 e       1                                      ( 2 )
$scfiterlimit       30
$scfconv        7
$energy    file=energy
$grad    file=grad
$end
```

File coord

```
$coord
     .00000000000000       .00000000000000       .54561506935122      n
    -.87806233111566      1.52084856970468      -.18187168978374      h
    -.87806233111566     -1.52084856970468      -.18187168978374      h
    1.75612466223131       .00000000000000      -.18187168978374      h
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    4    1           val=   1.90084
   2 k  1.0000000000000 bend    4    3    1      val= 106.27756
        1.0000000000000 bend    3    2    1
        1.0000000000000 bend    2    4    1
$end
```

File basis

```
$basis
*
n def-SVP
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
```

```
*
   5  s
  1712.8415853      -.53934125305E-02
  257.64812677      -.40221581118E-01
  58.458245853      -.17931144990
  16.198367905      -.46376317823
  5.0052600809      -.44171422662
   1  s
  .58731856571       1.0000000000
   1  s
  .18764592253       1.0000000000
   3  p
  13.571470233      -.40072398852E-01
  2.9257372874      -.21807045028
  .79927750754      -.51294466049
   1  p
  .21954348034       1.0000000000
   1  d
  1.0000000000       1.0000000000
*
h def-SVP
# h     (7s) / [3s]     {511}
*
   3  s
  13.010701000       .19682158000E-01
  1.9622572000       .13796524000
  .44453796000       .47831935000
   1  s
  .12194962000       1.0000000000
   1  p
  .80000000000       1.0000000000
*
$end
```

File mos

```
$scfmo    expanded   format(4d20.14)
     1  a1     eigenvalue=-.15633041862301D+02   nsaos=10
 .98699003163455D+00-.47221435341751D-01 .55873125006179D-02-.48016374887169D-02
 .26746008768233D-02 .20823779196149D-03 .14270460008808D-01 .90849517503597D-02
 .58676121352806D-03 .29091871198884D-03
     2  a1     eigenvalue=-.99896275238736D+00   nsaos=10
 .26412162337482D+00 .51846472345768D+00 .37623729061179D+00-.77139882704089D-02
-.47252329287316D-02-.21494050853221D-02 .11795673774658D+00 .83316086019184D-01
```

```
-.11229203933488D-01-.27038186251429D-02
     3  a1     eigenvalue=-.57101279949392D+00   nsaos=10
-.35584199011701D-01-.96938258881594D-01-.70254605702716D-01 .65569041318341D+00
-.44746149963029D+00 .40094287741992D-03 .51691151834284D-01 .47722350097160D-01
 .19189122068531D-02 .56638497851180D-03
     1  e      eigenvalue=-.64374209294851D+00   nsaos=9
-.49313475446075D+00 .33757893447603D+00-.76142296567409D-04-.74524664248740D-04
-.26407572210452D+00-.22619038902975D+00-.50035170531670D-05-.12199166245418D-03
 .63021657663245D-04
$end
```

24.3 NO_2 input for an unrestricted DFT calculation

Main File control

```
$title
NO2 c2v UKS SVP
$symmetry c2v
$coord    file=coord
$atoms
   basis =def-SVP
$basis    file=basis
$uhfmo_alpha none   file=alpha
$uhfmo_beta none  file=beta
# none : hamilton core guess will be made
# files alpha and beta will be generated by the program
$uhf
$alpha shells
 a1      1-6                                    ( 1 )
 a2      1                                      ( 1 )
 b1      1-4                                    ( 1 )
 b2      1                                      ( 1 )
$beta shells
 a1      1-5                                    ( 1 )
 a2      1                                      ( 1 )
 b1      1-4                                    ( 1 )
 b2      1                                      ( 1 )
$scfiterlimit       30
$scfconv        7
$energy    file=energy
$grad    file=grad
$dft
 functional b-p
 gridsize m3
$end
```

File coord

```
$coord
     .00000000000000       .00000000000000     -1.00494155217173      n
    1.85766051386774       .00000000000000       .50247077608587      o
   -1.85766051386774       .00000000000000       .50247077608587      o
$end
```

24.4 $TaCl_5$ Input for an RI-DFT Calculation with ECPs

Main File control

```
$title
$symmetry d3h
$coord    file=coord
$atoms
ta 1                                                        \
   jbas=ta def-SVP                                          \
   basis =ta def-SVP                                        \
   ecp   =ta def-ecp
cl 2-6                                                      \
   jbas=cl def-SVP                                          \
   basis =cl def-SVP
$basis    file=basis
$ecp    file=basis
$scfmo   none   file=mos
# none : hamilton core guess will be made
# file mos will be generated by the program
$scfiterlimit       30
$scfconv        6
$energy    file=energy
$grad    file=grad
$dft
   functional b-p
   gridsize   m3
$ricore    20
$ridft
$jbas file=auxbasis
$closed shells
 a1'     1-11                                    ( 2 )
 a2'     1-2                                     ( 2 )
 e'      1-10                                    ( 2 )
 a2"     1-8                                     ( 2 )
 e"      1-4                                     ( 2 )
$end
```

File coord

```
$coord
     .00000000000000       .00000000000000       .00000000000000      ta
    2.19392179448315     -3.79998401587749       .00000000000000      cl
```

```
    2.19392179448315      3.79998401587749       .00000000000000      cl
   -4.38784358896629       .00000000000000       .00000000000000      cl
     .00000000000000       .00000000000000      4.46615918865523      cl
     .00000000000000       .00000000000000     -4.46615918865523      cl
$end
```

File basis

```
$basis
*
ta def-SVP
# ta    (7s6p5d) / [6s3p2d]     {211111/411/41}
*
   2  s
  14.400000000      .99343296745
  12.000000000     -1.6510077975
   1  s
  5.0701477302      1.0000000000
   1  s
  .86033356487      1.0000000000
   1  s
  .37158938894      1.0000000000
   1  s
  .10745336254      1.0000000000
   1  s
  .39142776556E-01  1.0000000000
   4  p
  7.4188720000      .26979695152
  5.6984100000     -.46968874449
  1.1777211960      .50905100155
  .54478533555      .52298161137
   1  p
  .22309270117      1.0000000000
   1  p
  .43100000000E-01  1.0000000000
   4  d
  3.9738796278     -.52799310714E-01
  1.4528884813      .18558319471
  .61042908544      .42959071631
  .24216276510      .43497228232
   1  d
  .87909318337E-01  1.0000000000
*
```

```
cl def-SVP
# cl    (7s5p) / [6s2p]     {211111/41}
*
   5  s
  10449.827566      .19708362484E-02
  1571.7365221      .14754727977E-01
  357.12065523      .66679112875E-01
  100.25185935      .17228924084
  30.812727554      .15883786100
   3  s
  51.923789434     -.10009298909
  5.7045760975      .60841752753
  2.3508376809      .54352153355
   1  s
  .44605124672      1.0000000000
   1  s
  .16848856190      1.0000000000
   5  p
  307.66790569     -.87801484118E-02
  72.102015515     -.63563355471E-01
  22.532680262     -.24016428276
  7.8991765444     -.47798866557
  2.8767268321     -.38515850005
   1  p
  .77459363955      1.0000000000
   1  p
  .21037699698      1.0000000000
   1  d
  .65000000000      1.0000000000
*
$ecp
*
ta def-ecp
*
  ncore =    60           lmax =     3
#        coefficient   r^n          exponent
f
          12.0179609    2           2.0178811
s-f
        1345.8806470    2          14.5464077
          36.7668062    2           7.2732038
         -12.0179609    2           2.0178811
p-f
```

```
         378.4253015    2          9.9355653
          22.2930909    2          4.9677824
         -12.0179609    2          2.0178811
d-f
         104.8839557    2          6.3473769
           8.7558481    2          3.1736885
         -12.0179609    2          2.0178811
*
$end
```

24.5 Basisset optimization for Nitrogen

Main File control

```
$title
 Basisset-optimization for nitrogen SV(P)
$symmetry oh
#--- uncomment following line to clean the basis-file after optimization ----
#$dump basis set
$coord    file=coord
$user-defined bonds    file=coord
$pople   AO
$basis    file=basis
$scfmo none    file=mos
$roothaan         1
      a = 1      b = 2
$scfiterlimit       60
$scfconv       10
$scfdiis   start=0.5
$scforbitalshift  closedshell=.4
$drvopt
   cartesian  off
#---- optimize basis! -> basis on ----
   basis      on
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
$optimize
   internal   off
   cartesian  off
   global     off
#---- optimize basis! -> basis on logarithm ----
   basis      on logarithm
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.6
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=gradient
#---- optimize basis! -> $egrad file=egradient ----
$egrad    file=egradient
```

```
$forceapprox    file=forceapprox
$atoms
n  1                                                                          \
   basis =n def-SV(P)
$closed shells
 a1g     1-2                                    ( 2 )
$open shells type=1
 t1u     1                                      ( 1 )
$end
```

File coord

```
$coord
    0.00000000000000      0.00000000000000      0.00000000000000      n
$end
```

File basis

```
*
n def-SV(P)
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
# use expopt to optimize exponents and contopt to optimize contractions
*
   5  s    expopt  contopt
  1712.8415853      0.53934125305E-02
  257.64812677      0.40221581118E-01
  58.458245853      0.17931144990
  16.198367905      0.46376317823
  5.0052600809      0.44171422662
   1  s  expopt
 0.58731856571       1.0000000000
   1  s  expopt
 0.18764592253       1.0000000000
   3  p    expopt  contopt
  13.571470233      0.40072398852E-01
  2.9257372874      0.21807045028
 0.79927750754      0.51294466049
   1  p  expopt
 0.21954348034       1.0000000000
#  1  d
# 1.0000000000       1.0000000000
*
```

File mos

```
$scfmo    scfconv=10   format(4d20.14)
# SCF energy is      -54.3329250250 a.u.  (virial theorem =  2.000000001)
#
     1  a1g    eigenvalue=-.15623888057347D+02   nsaos=3
-.99166890864040D+00-.28420294406651D-010.91519592317893D-02
     2  a1g    eigenvalue=-.92524548524703D+00   nsaos=3
0.30506869715453D+00-.65051761026701D+00-.44610487551870D+00
     3  a1g    eigenvalue=0.74881229854801D+00   nsaos=3
0.30759302935434D+00-.16295969601691D+010.16126161147521D+01
     1  t1u    eigenvalue=-.56865046629517D+00   nsaos=2
0.67926397018841D+000.46005039868410D+00
     2  t1u    eigenvalue=0.96169069264790D+00   nsaos=2
-.95675659621171D+000.10794148212163D+01
$end
```

24.6 ROHF of Two Open Shells

Extracts from `control` for O_2 in D_{3d} Symmetry

```
# HF-SCF/SVP


# Reference: triplet-sigma in D3d
# This is a Roothaan case (as is D-infinity-h).
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d3d
$closed shells
 a1g     1-3                        (2)
 a2u     1-2                        (2)
 eu      1                          (2)
$open shells type=1
 eg      1                          (1)
$roothaan         1
      a = 1      b = 2
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4774402753    149.4799190239   -298.9573592992


# Reference: singlet-delta in D3d
# This is a Roothaan case (as is D-infinity-h).
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d3d
$closed shells
 a1g     1-3                        (2)
 a2u     1-2                        (2)
 eu      1                          (2)
$open shells type=1
 eg      1                          (1)
$roothaan       1
      a = 1/2   b = 0
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4297623470    149.4298692899   -298.8596316369
```

Extracts from `control` for O_2 in D_{2h} Symmetry

```
# HF-SCF/SVP


# Triplet-sigma in D2h
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d2h
$closed shells
 ag      1-3                        ( 2 )
 b1u     1-2                        ( 2 )
 b2u     1                          ( 2 )
 b3u     1                          ( 2 )
$open shells type=1
 b2g     1                          ( 1 )
 b3g     1                          ( 1 )
$roothaan          1
      a = 1    b = 2
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4774402750    149.4798706643   -298.9573109393


# Singlet-delta in D2h : xx-yy component
# where x = b2g and y = b3g.  In D-infinity-h, b2g and b3g combine to eg.
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d2h
$closed shells
 ag      1-3                        ( 2 )
 b1u     1-2                        ( 2 )
 b2u     1                          ( 2 )
 b3u     1                          ( 2 )
$open shells type=1
 b2g     1                          ( 1 )
 b3g     1                          ( 1 )
$roothaan          2
$rohf
1b2g-1b3g    a = 0      b = 2
```

```
1b2g-1b2g    a = 1      b = 0
1b3g-1b3g    a = 1      b = 0
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4297623516    149.4298351805   -298.8595975321


# Singlet-delta in D2h : xy+yx component
# (an example of the general type: [xy]-singlet)
# where in D2h x = b2g and y = b3g are of different symmetry.
# In D-infinity-h, b2g and b3g combine to eg; see the reference
# calculation in D3d above.
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d2h
$closed shells
 ag      1-3                        ( 2 )
 b1u     1-2                        ( 2 )
 b2u     1                          ( 2 )
 b3u     1                          ( 2 )
$open shells type=1
 b2g     1                          ( 1 )
 b3g     1                          ( 1 )
$roothaan          2
$rohf
1b2g-1b3g    a = 1      b = -2
1b2g-1b2g    a = 0      b =  0
1b3g-1b3g    a = 0      b =  0
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4297623501    149.4298391833   -298.8596015334
```

Chapter 25

The Perl-based Test Suite Structure

25.1 General

Testing the TURBOMOLE modules for correctness and speed is the first task once the coding is completed. It is subject to automatization and thus requires a structure which is as simple and flexible as possible. In the Perl-based test suite this is implemented by a Perl script `TTEST` which performs all the testing and benchmarking tasks and resides in the central `scripts` directory of the TURBOMOLE installation. The test examples are located in subdirectories of the `TURBOTEST` directory, grouped according to the modules modules to be tested and a rough short/long classification. The benchmark suite shows the same directory structure and is rooted in the `TURBOBENCH` directory.

The central idea of the Perl-based test suite is that only the specific information about an individual test example is included in its local directory along with the input and reference files. This information is stored in the criteria file `CRIT` which contains the program calls, test criteria, and specific reference timings. Running the test script creates a new test subdirectory, usually called like `TESTDIR.i786-pc-linux-gnu`, where the TURBOMOLE programs are run and the results are summarized in the protocol file `TESTPROTOKOLL`.

25.2 Running the tests

Starting a single test example is simple. Change to the test example of your choice and call the `TTEST` script without arguments. The test is started in a subdirectory named `TESTDIR`.*sysname*, where *sysname* is the current platform name as returned by the `Sysname` script. The tested executable, a short description, and the test summary are output to the screen. Detailed information about the performed commands and results of all test criteria are found in the `TESTPROTOKOLL` file in the test subdirectory.

The default location for the binaries and scripts used for testing is the `$TURBODIR` directory. If you like to test some other, e.g., your local version of the TURBOMOLE binaries or scripts, you can specify the loading paths by the `-l` or `-ls` options for the binaries and scripts, respectively,

```
TTEST -l /usr/local/TURBOMOLE/bin/i786-pc-linux-gnu \
   -ls /usr/local/TURBOMOLE/scripts.
```

A specific executable can be chosen by the `-x` option,

```
TTEST -x /usr/local/TURBOMOLE/bin/i786-pc-linux-gnu/dscf.
```

If a test output is already present, e.g., in the `TESTDIR` directory, you may wish to check the results. This is accomplished by calling `TTEST` in check mode,

```
TTEST --check TESTDIR,
```

which compares the results in `TESTDIR` with the reference and writes the results to the `CHECKPROTOKOLL` file in the test directory.

Testing parts of the `TURBOTEST` directory structure or the entire test suite at once is performed by calling the `TTEST` script from the appropriate place. The test script works recursively, executing all test examples underneath its starting directory. This requires that the test examples be arranged in a `TURBOTEST`-like directory structure,

```
progname/short|long/example  (e.g., dscf/short/H2O.SCF.E1),
```

and the `TURBOTEST` directory contain a `DEFCRIT` file with general test suite settings. If `TTEST` is started in the central `TURBOTEST` without any options, all available test examples are executed. By giving the list of module names (for full list, check `TTEST -help`) as argument to the script, the test can be restricted to these modules. The `-short` and `-long` options allow the user to select only the short or long test examples, respectively. Some examples of usage are given in the following table:

`TTEST dscf`	called in the `TURBOTEST` directory, performs only the tests for DSCF module.
`TTEST`	called in the `TURBOTEST/dscf` directory, does the same.
`TTEST -long`	executes long examples for all modules.
`TTEST ridft -short`	performs all short examples from the `ridft` directory.

Recursive testing creates some additional files in the central `TURBOTEST` directory. The global protocol file `TESTPROTOKOLL`.*sysname* contains short result messages for all test and a list of errors occurred. The list of failed tests is also written to the `PROBLEMS`.*sysname* file and can be rerun by calling the test script with the `-r` option,

```
TTEST -r PROBLEMS.i786-pc-linux-gnu.
```

The `-r` may also be useful to create any user-defined selection of test examples. The full list of available examples is obtained by the `TTEST -list` command.

Once you are done with testing, you may wish to clean up afterwards. To do it, use the `-clean` and `-realclean` options of the `TTEST` script. The difference between these two is that `TTEST -clean` deletes only the test directories and protocols that were created for the current computer architecture as returned by `Sysname`. In contrast, the `TTEST -realclean` wipes out all test directories and protocols that get in its way.

25.3 Taking the timings and benchmarking

Benchmarking differs from testing only in that program timings are computed and compared with reference timings. Calling the script as

```
TTEST --timings
```

performs the test, calculates the CPU and wall clock timings, and writes the raw results to the `TESTTIMINGS`.*sysname.nodename* file. Auxiliary scripts `Tbtim` and `Tblist` help to convert this data to a more readable form and produce summaries as LaTeX tables. The `Tbtim` script creates a summary of benchmark results for a given computer platform from the original timings file. `Tblist` produces benchmark comparisons of different platforms. The corresponding timings files must be provided as arguments to the `Tblist` script. For more details and options, see `TBTIM -help` and `TBLIST -help`.

25.4 Modes and options of the TTEST script

The TTEST script knows several operation modes: "run", "check", "list", "clean", "realclean", and "validate", controlled by its options. The "run" mode is default and means that the test calculations are performed and the results are written to the TESTPROTOKOLL file. The "check" mode differs only in that the programs are not executed, but the existing program output is checked against the reference. The results of the check are written to the CHECKPROTOKOLL file. Calling the test script in the "list" mode simply lists the test examples that are currently available. This allows the user to save the full list to file, edit, and re-use it with the -r option. The "clean" and "realclean" options are for cleaning up the test directories and protocols. Finally, the "validate" mode is mainly of use for writing the CRIT files. It helps to verify the match patterns provided in the test criteria and shows if it extracts the expected data for comparison with the reference. For every output file used for testing, the "validate" option produces a copy with an additional .val extension. The match strings evaluated for test criteria are highlighted in the output by «««< and »»»> marks.

There is a lot of options controlling the behavior of TTESTṪesting specific versions of TURBOMOLE modules is provided by loading path options, -l for binaries, -ls for scripts, and -x for a specific executable. For benchmarking, you need the -timings option to produce the timing summaries, and the -newref option to save the current program timings as the new reference. The module specifications and -short, -long, and -r options can be used for selecting the test examples. The more specialized options are summarized in the following table. Note that most of these options can also be set in the DEFCRIT file (see below).

Operation modes

-help	Prints out the help message and exits.
-h	
-?	
-list	Lists the available test examples.
-clean	Deletes the test directories and summary files for the current architecture (given by SYSNAME, see Chapter 1.5).
-realclean	Deletes all test directories and protocols.

`-check` *dir*	Checks the correctness of an existing program test in the directory *dir* (default: `TESTDIR`.*sysname*). Useful if new criteria or new references are established.
`-validate` *dir* `-val` *dir*	Examines the output files in the directory *dir* (default: `TESTDIR`.*sysname*) and highlights the positions of the retrieved matches.

Loading path and naming options

`-loaddir` *dir* `-l` *dir*	Loading path for the TURBOMOLE binaries (default: `$TURBODIR/bin/`*sysname*).
`-scriptdir` *dir* `-ls` *dir*	Loading path for the TURBOMOLE scripts (default: `$TURBODIR/scripts`).
`-testprog` *prog* `-x` *prog*	Tests the given executable *prog*.
`-dir` *dir*	Name for the local test directory (default: `TESTDIR`.*sysname*).
`-critfile` *file*	Name for the local criteria file (default: `CRIT`).
`-defcritfile` *file*	Name for the test suite settings file (default: `DEFCRIT`).
`-protfile` *file* `-output` *file*	Name for the local protocol file (default: `TESTPROTOKOLL`).
`-gprotfile` *file*	Name for the global protocol file (default: `TESTPROTOKOLL`.*sysname*).
`-checkfile` *file*	Name for the check protocol file (default: `CHECKPROTOKOLL`).
`-errfile` *file*	Name for the local error output file (default: `output.err`).
`-probfile` *file*	Name for the failed tests list (default: `PROBLEMS`.*sysname*).
`-timfile` *file*	Name for the timings file (default: `TIMINGS`.*sysname*).
`-valfile` *file*	Name for the validation file for 'run' criteria (default: `RUNCRIT.val`).

Execution options

`-short`	Only `short` / `long` subdirectories of the
`-long`	test suite will be tested (default: `-short -long`).
`-restart` *file*	The list of test examples for execution will
`-r` *file*	be read in from *file*
	(default: `PROBLEMS`.*sysname*).
`-newref` *string*	Produces new reference timings and writes them to the `CRIT` file. A short description of the reference platform is provided by *string*.
`-fileref`	Produces new reference files.
`-batchmode`	Running in batch mode, no screen output.
`-errorstop`	Stops / Does not stop after the first error.
`-noerrorstop`	(default: `-noerrorstop`).
`-timings`	Writes / Does not write the timings on file for
`-notimings`	further processing. (default: `-notimings`).
`-runopts`	Sets the conditions under which the test is run
`-o`	(default: `"sequential, parallel"`)

Bibliography

[1] R. Ahlrichs; M. Bär; M. Häser; H. Horn; C. Kölmel. Electronic structure calculations on workstation computers: The program system Turbomole. *Chem. Phys. Lett.*, **162**(3), 165–169, (1989).

[2] S. G. Balasubramani; G. P. Chen; S. Coriani; M. Diedenhofen; M. S. Frank; Y. J. Franzke; F. Furche; R. Grotjahn; M. E. Harding; C. Hättig; A. Hellweg; B. Helmich-Paris; C. Holzer; U. Huniar; M. Kaupp; A. Marefat Khah; S. Karbalaei Khani; T. Müller; F. Mack; B. D. Nguyen; S. M. Parker; E. Perlt; D. Rappoport; K. Reiter; S. Roy; M. Rückert; G. Schmitz; M. Sierka; E. Tapavicza; D. P. Tew; C. van Wüllen; V. K. Voora; F. Weigend; A. Wodyński; J. M. Yu. Turbomole: Modular program suite for *ab initio* quantum-chemical and condensed-matter simulations. *J. Chem. Phys.*, **152**, 184107, (2020).

[3] B. P. Pritchard; D. Altarawy; B. Didier; T. D. Gibson; T. L. Windus. New basis set exchange: An open, up-to-date resource for the molecular sciences community. *J. Chem. Inf. Model.*, **59**(11), 4814–4820, (2019).

[4] K. L. Schuchardt; B. T. Didier; T. Elsethagen; L. Sun; V. Gurumoorthi; J. Chase; J. Li; T. L. Windus. Basis set exchange: A community database for computational science. *J. Chem. Inf. Model.*, **47**(3), 1045–1052, (2007).

[5] A. Schäfer; H. Horn; R. Ahlrichs. Fully optimized contracted Gaussian basis sets for atoms Li to Kr. *J. Chem. Phys.*, **97**(4), 2571–2577, (1992).

[6] A. Schäfer; C. Huber; R. Ahlrichs. Fully optimized contracted Gaussian basis sets of triple zeta valence quality for atoms Li to Kr. *J. Chem. Phys.*, **100**(8), 5829–5835, (1994).

[7] K. Eichkorn; F. Weigend; O. Treutler; R. Ahlrichs. Auxiliary basis sets for main row atoms and transition metals and their use to approximate Coulomb potentials. *Theor. Chem. Acc.*, **97**(1–4), 119–124, (1997).

[8] F. Weigend; F. Furche; R. Ahlrichs. Gaussian basis sets of quadruple zeta valence quality for atoms H–Kr. *J. Chem. Phys.*, **119**(24), 12753–12762, (2003).

[9] F. Weigend; R. Ahlrichs. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.*, **7**(18), 3297–3305, (2005).

[10] A. K. Rappé; C. J. Casewit; K. S. Colwell; W. A. Goddard III; W. M. Skiff. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.*, **114**(25), 10024–10035, (1992).

[11] C. Hättig; F. Weigend. CC2 excitation energy calculations on large molecules using the resolution of the identity approximation. *J. Chem. Phys.*, **113**(13), 5154–5161, (2000).

[12] C. Hättig; K. Hald. Implementation of RI-CC2 for triplet excitation energies with an application to *trans*-azobenzene. *Phys. Chem. Chem. Phys.*, **4**(11), 2111–2118, (2002).

[13] C. Hättig; A. Köhn; K. Hald. First–order properties for triplet excited states in the approximated coupled cluster model CC2 using an explicitly spin coupled basis. *J. Chem. Phys.*, **116**(13), 5401–5410, (2002).

[14] C. Hättig. Geometry optimizations with the coupled-cluster model CC2 using the resolution-of-the-identity approximation. *J. Chem. Phys.*, **118**(17), 7751–7761, (2003).

[15] A. Köhn; C. Hättig. Analytic gradients for excited states in the coupled-cluster model CC2 employing the resolution-of-the-identity approximation. *J. Chem. Phys.*, **119**(10), 5021–5036, (2003).

[16] C. Hättig; A. Hellweg; A. Köhn. Distributed memory parallel implementation of energies and gradients for second-order Møller-Plesset perturbation theory with the resolution-of-the-identity approximation. *Phys. Chem. Chem. Phys.*, **8**(10), 1159–1169, (2006).

[17] A. Hellweg; S. Grün; C. Hättig. Benchmarking the performance of spin–component scaled CC2 in ground and electronically excited states. *Phys. Chem. Chem. Phys.*, **10**, 1159–1169, (2008).

[18] N. O. C. Winter; C. Hättig. Scaled opposite-spin CC2 for ground and excited states with fourth order scaling computational costs. *J. Chem. Phys.*, **134**, 184101, (2011).

[19] R. A. Bachorz; F. A. Bischoff; A. Glöß; C. Hättig; S. Höfener; W. Klopper; D. P. Tew. The mp2-f12 method in the turbomole programm package. *J. Comput. Chem.*, **32**, 2492–2513, (2011).

[20] D. P. Tew; W. Klopper; C. Neiss; C. Hättig. Quintuple-ζ quality coupled-cluster correlation energies with triple-ζ basis sets. *Phys. Chem. Chem. Phys.*, **9**, 1921–1930, (2007).

[21] C. Hättig; D. P. Tew; A. Köhn. Accurate and efficient approximations to explicitly correlated coupled-cluster singles and doubles, CCSD-F12. *J. Chem. Phys.*, **132**, 231102, (2010).

[22] D. P. Tew; W. Klopper. Open-shell explicitly correlated f12 methods. *Mol. Phys.*, **108**, 315–325, (2010).

[23] G. Schmitz; B. Helmich; C. Hättig. A $O(N^3)$-scaling PNO-MP2 method using a hybrid OSV-PNO approach with an iterative direct generation of OSVs. *Mol. Phys.*, **111**, 2463–2473, (2013).

[24] G. Schmitz; C. Hättig; D. Tew. Explicitly correlated PNO-MP2 and PNO-CCSD and its application to the S66 set and large molecular systems. *Phys. Chem. Chem. Phys.*, **16**, 22167–22178, (2014).

[25] R. Bauernschmitt; R. Ahlrichs. Treatment of electronic excitations within the adiabatic approximation of time dependent density functional theory. *Chem. Phys. Lett.*, **256**(4–5), 454–464, (1996).

[26] R. Bauernschmitt; R. Ahlrichs. Stability analysis for solutions of the closed shell Kohn-Sham equation. *J. Chem. Phys.*, **104**(22), 9047–9052, (1996).

[27] M. Kühn; F. Weigend. Implementation of Two-component Time-Dependent Density Functional Theory in TURBOMOLE. *J. Chem. Theory Comput.*, **9**, 5341–5348, (2013).

[28] M. Kühn; F. Weigend. Two-Component Hybrid Time-Dependent Density Functional Theory within the Tamm-Dancoff Approximation. *J. Chem. Phys.*, **142**, 034116, (2015).

[29] C. Holzer; W. Klopper. Ionized, electron-attached, and excited states of molecular systems with spin–orbit coupling: Two-component gw and bethe–salpeter implementations. *J. Chem. Phys.*, **150**(20), 204116, (2019).

[30] C. Holzer; W. Klopper. Communication: A hybrid bethe–salpeter/time-dependent density-functional-theory approach for excitation energies. *J. Chem. Phys.*, **149**(10), 101101, (2018).

[31] X. Gui; C. Holzer; W. Klopper. Accuracy assessment of gw starting points for calculating molecular excitation energies using the bethe–salpeter formalism. *J. Chem. Theory Comput.*, **14**(4), 2127–2136, (2018).

[32] C. Holzer; X. Gui; M. E. Harding; G. Kresse; T. Helgaker; W. Klopper. Bethe–salpeter correlation energies of atoms and molecules. *J. Chem. Phys.*, **149**(14), 144106, (2018).

[33] K. Krause; W. Klopper. Implementation of the bethe-salpeter equation in the turbomole program. *J. Comput. Chem.*, **38**, 383–388, (2017).

[34] M. J. van Setten; F. Weigend; F. Evers. The gw-method for quantum chemistry applications: Theory and implementation. *J. Chem. Theory Comput*, **9**, 232, (2013).

[35] M. Kühn; F. Weigend. One-Electron Energies from the Two-Component GW Method. *J. Chem. Theory Comput.*, **11**, 969–979, (2015).

[36] C. Holzer; A. M. Teale; F. Hampe; S. Stopkowicz; T. Helgaker; W. Klopper. Gw quasiparticle energies of atoms in strong magnetic fields. *J. Chem. Phys.*, **150**(21), 214112, (2019).

[37] M. Kehry; Y. J. Franzke; C. Holzer; W. Klopper. Quasirelativistic two-component core excitations and polarisabilities from a damped-response formulation of the bethe–salpeter equation. *Mol. Phys.*, **118**(21-22), e1755064, (2020).

[38] F. Furche; R. Ahlrichs. Adiabatic time-dependent density functional methods for excited state properties. *J. Chem. Phys.*, **117**(16), 7433–7447, (2002).

[39] M. Häser; R. Ahlrichs; H. P. Baron; P. Weis; H. Horn. Direct computation of second-order SCF properties of large molecules on workstation computers with an application to large carbon clusters. *Theor. Chim. Acta*, **83**(5–6), 455–470, (1992).

[40] M. Kollwitz; J. Gauss. A direct implementation of the GIAO-MBPT(2) method for calculating NMR chemical shifts. Application to the naphthalenium and and anthracenium ions. *Chem. Phys. Lett.*, **260**(5–6), 639–646, (1996).

[41] Y. J. Franzke; F. Weigend. NMR shielding tensors and chemical shifts in scalar-relativistic local exact two-component theory. *J. Chem. Theory Comput.*, **15**(2), 1028–1043, (2019).

[42] C. Holzer; Y. J. Franzke; A. Pausch. Current density functional framework for spin–orbit coupling. *J. Chem. Phys.*, **157**(20), 204102, (2022).

[43] Y. J. Franzke; C. Holzer. Exact two-component theory becoming an efficient tool for nmr shieldings and shifts with spin–orbit coupling. *ChemRxiv*, (2023).

[44] K. Reiter; F. Mack; F. Weigend. Calculation of magnetic shielding constants with meta-gga functionals employing the multipole-accelerated resolution of the identity: Implementation and assessment of accuracy and efficiency. *J. Chem. Theory Comput.*, **14**(1), 191–197, (2018).

[45] K. Reiter; M. Kühn; F. Weigend. Vibrational circular dichroism spectra for large molecules and molecules with heavy elements. *J. Chem. Phys.*, **146**(5), 054102, (2017).

[46] C. van Wüllen. Shared-memory parallelization of the TURBOMOLE programs AOFORCE, ESCF, and EGRAD: How to quickly parallelize legacy code. *J. Comput. Chem.*, **32**, 1195–1201, (2011).

[47] M. von Arnim; R. Ahlrichs. Geometry optimization in generalized natural internal coordinates. *J. Chem. Phys.*, **111**(20), 9183–9190, (1999).

[48] P. Pulay; G. Fogarasi; F. Pang; J. E. Boggs. Systematic ab initio gradient calculation of molecular geometries, force constants, and dipole moment derivatives. *J. Am. Chem. Soc.*, **101**(10), 2550–2560, (1979).

[49] M. Dolg; U. Wedig; H. Stoll; H. Preuß. Energy-adjusted ab initio pseudopotentials for the first row transition elements. *J. Chem. Phys.*, **86**(2), 866–872, (1986).

[50] C. C. J. Roothaan. Self-consistent field theory for open shells of electronic systems. *Rev. Mod. Phys.*, **32**(2), 179–185, (1960).

[51] R. Ahlrichs; F. Furche; S. Grimme. Comment on "Assessment of exchange correlation functionals". *Chem. Phys. Lett.*, **325**(1–3), 317–321, (2000).

[52] M. Sierka; A. Hogekamp; R. Ahlrichs. Fast evaluation of the Coulomb potential for electron densities using multipole accelerated resolution of identity approximation. *J. Chem. Phys.*, **118**(20), 9136–9148, (2003).

[53] F. Weigend. A fully direct RI-HF algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency. *Phys. Chem. Chem. Phys.*, **4**(18), 4285–4291, (2002).

[54] R. Fletcher. *Practical Methods of Optimization. Unconstrained Optimization.* Band 1. Wiley: New York, 1980.

[55] T. Helgaker. Transition-state optimizations by trust-region image minimization. *Chem. Phys. Lett.*, **182**(5), 503–510, (1991).

[56] F. Jensen. Locating transition structures by mode following: A comparison of six methods on the Ar_8 Lennard-Jones potential. *J. Chem. Phys.*, **102**(17), 6706–6718, (1995).

[57] P. Császár; P. Pulay. Geometry optimization by direct inversion in the iterative subspace. *J. Mol. Struct.*, **114**, 31–34, (1984).

[58] R. Fletcher. A new approach to variable metric algorithms. *Comput. J.*, **13**(3), 317–322, (1970).

[59] H. B. Schlegel. Optimization of equilibrium geometries and transition structures. *J. Comput. Chem.*, **3**(2), 214–218, (1982).

[60] H. B. Schlegel. Estimating the hessian for gradient-type geometry optimizations. *Theor. Chim. Acta*, **66**(5), 333–340, (1984).

[61] M. Ehrig. Diplomarbeit. Master's thesis, Universität Karlsruhe, 1990.

[62] T. Koga; H. Kobayashi. Exponent optimization by uniform scaling technique. *J. Chem. Phys.*, **82**(3), 1437–1439, (1985).

[63] A. K. Rappé; W. A. Goddard III. Charge equilibration for molecular dynamics simulations. *J. Phys. Chem.*, **95**(8), 3358–3363, (1991).

[64] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. General considerations. *J. Inst. Math. Appl.*, **6**(1), 76–90, (1970).

[65] D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comput.*, **24**(109), 23–26, (1970).

[66] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Math. Comput.*, **24**(111), 647–656, (1970).

[67] P. Pulay. Convergence acceleration of iterative sequences. the case of SCF iteration. *Chem. Phys. Lett.*, **73**(2), 393–398, (1980).

[68] S. Grimme; C. Bannwarth; P. Shushkov. A robust and accurate tight-binding quantum chemical method for structures, vibrational frequencies, and noncovalent interactions of large molecular systems parametrized for all spd-block elements (z = 1–86). *J. Chem. Theory Comput.*, **13**, 1989–2009, (2017).

[69] C. Bannwarth; S. Ehlert; S. Grimme. Gfn2-xtb—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *J. Chem. Theory Comput.*, **15**, 1652–1671, (2019).

[70] M. P. Allen; D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press: Oxford, 1987.

[71] M. Sierka. Synergy between theory and experiment in structure resolution of low-dimensional oxides. *Prog. Surf. Sci.*, **85**, 398–434, (2010).

[72] T. Halgren; W. Lipscomb. Synchronous-transit method for determining reaction pathways and locating molecular transition-states. *Chem. Phys. Lett.*, **49**(2), 225–232, (1977).

[73] R. Elber; M. Karplus. A method for determining reaction paths in large molecules - application to myoglobin. *Chem. Phys. Lett.*, **139**(5), 375–380, (1987).

[74] M. G; H. Jonsson. Quantum and thermal effects in h-2 dissociative adsorption - evaluation of free-energy barriers in multidimensional quantum-systems. *Phys. Rev. Lett.*, **72**(7), 1124–1127, (1994).

[75] G. Henkelman; H. Jonsson. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *J. Chem. Phys.*, **113**(22), 9978–9985, (2000).

[76] E. Weinan; W. Ren; E. Vanden-Eijnden. String method for the study of rare events. *Phys. Rev. B*, **66**(5), 052301, (2002).

[77] P. Plessow. Reaction path optimization without neb springs or interpolation algorithms. *J. Chem. Theory Comput.*, **9**(3), 1305–1310, (2013).

[78] K. Eichkorn; O. Treutler; H. Öhm; M. Häser; R. Ahlrichs. Auxiliary basis sets to approximate Coulomb potentials (erratum, 1995, **242**, 283). *Chem. Phys. Lett.*, **242**(6), 652–660, (1995).

[79] J. A. Pople; R. K. Nesbet. Self-consistent orbitals for radicals. *J. Chem. Phys.*, **22**(3), 571–572, (1954).

[80] J. Čižek; J. Paldus. Stability conditions for solutions of Hartree-Fock equations for atomic and molecular systems. application to pi-electron model of cyclic plyenes. *J. Chem. Phys.*, **47**(10), 3976–3985, (1967).

[81] F. Neese; F. Wennmohs; A. Hansen; U. Becker. Efficient, approximate and parallel Hartree–Fock and hybrid DFT calculations. A 'chain-of-spheres' algorithm for the Hartree–Fock exchange. *Chem. Phys.*, **356**, 98–109, (2009).

[82] J. S. Andrews; D. Jayatilaka; R. G. Bone; N. C. Handy; R. D. Amos. Spin contamination in single-determinant wavefunctions. *Chem. Phys. Lett.*, **183**(5), 423–431, (1991).

[83] F. Bruder; Y. J. Franzke; F. Weigend. Paramagnetic NMR shielding tensors based on scalar exact two-component and spin-orbit perturbation theory. *J. Phys. Chem. A*, **126**(30), 5050–5069, (2022).

[84] U. Ekström; L. Visscher; R. Bast; A. J. Thorvaldsen; K. Ruud. Arbitrary-order density functional response theory from automatic differentiation. *J. Chem. Theory Comput.*, **6**, 1971–1980, (2010).

[85] Y. Zhao; D. G. Truhlar. The M06 suite of density functionals for main group thermochemistry, thermochemical kinetics, noncovalent interactions, excited states, and transition elements: two new functionals and systematic testing of four M06-class functionals and 12 other functionals. *Theor. Chem. Acc.*, **120**, 215–241, (2008).

[86] S. Lehtola; C. Steigemann; M. J. T. Oliveira; M. A. L. Marques. Recent developments in LIBXC - A comprehensive library of functionals for density functional theory. *SoftwareX*, **7**, 1–5, (2018).

[87] J.-D. Chaia; M. Head-Gordon. Systematic optimization of long–range corrected hybrid density functionals. *J. Chem. Phys.*, **128**, 084106, (2008).

[88] P. A. M. Dirac. Quantum mechanics of many-electron systems. *Proc. Royal Soc. (London) A*, **123**(792), 714–733, (1929).

[89] J. C. Slater. A simplification of the Hartree-Fock method. *Phys. Rev.*, **81**(3), 385–390, (1951).

[90] S. Vosko; L. Wilk; M. Nusair. Accurate spin-dependent electron-liquid correlation energies for local spin density calculations: a critical analysis. *Can. J. Phys.*, **58**(8), 1200–1211, (1980).

[91] J. P. Perdew; Y. Wang. Accurate and simple analytic representation of the electron-gas correlation energy. *Phys. Rev. B*, **45**(23), 13244–13249, (1992).

[92] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behaviour. *Phys. Rev. A*, **38**(6), 3098–3100, (1988).

[93] C. Lee; W. Yang; R. G. Parr. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, **37**(2), 785–789, (1988).

[94] J. P. Perdew. Density-functional approximation for the correlation-energy of the inhomogenous electron gas. *Phys. Rev. B*, **33**(12), 8822–8824, (1986).

[95] J. P. Perdew; K. Burke; M. Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, **77**(18), 3865–3868, (1996).

[96] J. Tao; J. P. Perdew; V. N. Staroverov; G. E. Scuseria. Climbing the density functional ladder: Nonempirical meta–generalized gradient approximation designed for molecules and solids. *Phys. Rev. Lett.*, **91**(14), 146401, (2003).

[97] J. Sun; A. Ruzsinszky; J. P. Perdew. Strongly constrained and appropriately normed semilocal density functional. *Phys. Rev. Lett.*, **115**, 036402, (2015).

[98] J. W. Furness; A. D. Kaplan; J. Ning; J. P. Perdew; J. Sun. Accurate and numerically efficient r2scan meta-generalized gradient approximation. *J. Phys. Chem. Lett.*, **11**(19), 8208–8215, (2020).

[99] S. Grimme; A. Hansen; S. Ehlert; J.-M. Mewes. r2scan-3c: An efficient "swiss army knife" composite electronic-structure method. *J. Chem. Phys.*, **154**(6), 064103, (2021).

[100] J. G. Brandenburg; J. E. Bates; J. Sun; J. P. Perdew. Benchmark tests of a strongly constrained semilocal functional with a long-range dispersion correction. *Phys. Rev. B*, **94**, 115144, (2016).

[101] A. D. Becke. A new mixing of Hartree-Fock and local density-functional theories. *J. Chem. Phys.*, **98**(2), 1372–1377, (1993).

[102] A. D. Becke. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.*, **98**(7), 5648–5652, (1993).

[103] J. P. Perdew; M. Ernzerhof; K. Burke. Rationale for mixing exact exchange with density functional approximations. *J. Chem. Phys.*, **105**(22), 9982–9985, (1996).

[104] V. N. Staroverov; G. E. Scuseria; J. Tao; J. P. Perdew. Comparative assessment of a new nonempirical density functional: Molecules and hydrogen-bonded complexes. *J. Chem. Phys.*, **119**(23), 12129–12137, (2003).

[105] A. Görling; M. Levy. Correlation-energy functional and its high-density limit obtained from a coupling-constant perturbation expansion. *Phys. Rev. B*, **47**, 13105, (1993).

[106] A. Görling; M. Levy. Exact Kohn-Sham scheme based on perturbation theory. *Phys. Rev. A*, **50**, 196, (1994).

[107] X. Ren; A. Tkatchenko; P. Rinke; M. Scheffler. Beyond the random-phase approximation for the electron correlation energy: The importance of single excitations. *Phys. Rev. Lett.*, **106**(15), 153003, (2011).

[108] L. Goerigk; S. Grimme. Double-hybrid density functionals. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, **4**(6), 576–600, (2014).

[109] S. Grimme. Semiempirical hybrid density functional with perturbative second-order correlation. *J. Chem. Phys.*, **124**, 034108, (2006).

[110] A. Karton; A. Tarnopolsky; J.-F. Lamére; G. C. Schatz; J. M. Martin. Highly accurate first-principles benchmark data sets for the parametrization and validation of density functional and other approximate methods. derivation of a robust, generally applicable, double-hybrid functional for thermochemistry and thermochemical kinetics. *J. Phys. Chem A*, **112**(50), 12868–12886, (2008).

[111] Y. Zhang; X. Xu; W. A. Goddard III. Doubly hybrid density functional for accurate descriptions of nonbond interactions, thermochemistry, and thermochemical kinetics. *PNAS*, **106**, 4963–4968, (2009).

[112] M. Seidl; S. Giarrusso; S. Vuckovic; E. Fabiano; P. Gori-Giorgi. Communication: Strong-interaction limit of an adiabatic connection in hartree-fock theory. *J. Chem. Phys*, **149**(24), 241101, (2018).

[113] M. Seidl; J. P. Perdew; S. Kurth. Density functionals for the strong-interaction limit. *Phys. Rev. A*, **62**(1), 012502, (2000).

[114] M. Seidl; J. P. Perdew; S. Kurth. Simulation of all-order density-functional perturbation theory, using the second order and the strong-correlation limit. *Phys. Rev. Lett.*, **84**(22), 5070, (2000).

[115] E. Fabiano; P. Gori-Giorgi; M. Seidl; F. Della Sala. Interaction-strength interpolation method for main-group chemistry: Benchmarking, limitations, and perspectives. *J. Chem. Theory Comput.*, **12**(10), 4885–4896, (2016).

[116] T. J. Daas; E. Fabiano; F. Della Sala; P. Gori-Giorgi; S. Vuckovic. Noncovalent interactions from models for the møller–plesset adiabatic connection. *J. Phys. Chem. Lett.*, **12**(20), 4867–4875, (2021).

[117] S. Śmiga; F. Della Sala; P. Gori-Giorgi; E. Fabiano. Self-consistent implementation of kohn–sham adiabatic connection models with improved treatment of the strong-interaction limit. *J. Chem. Theory Comput.*, **18**(10), 5936–5947, (2022).

[118] S. Vuckovic; P. Gori-Giorgi; F. Della Sala; E. Fabiano. Restoring size consistency of approximate functionals constructed from the adiabatic connection. **9**(11), 3137–3142, (2018).

[119] J. Jaramillo; G. E. Scuseria; M. Ernzerhof. Local hybrid functionals. *J. Chem. Phys.*, **118**, 1068–1073, (2003).

[120] H. Bahmann; M. Kaupp. Efficient self-consistent implementation of local hybrid functionals. *J. Chem. Theory Comput.*, **11**, 1540–1548, (2015).

[121] S. Klawohn; H. Bahmann; M. Kaupp. Implementation of molecular gradients for local hybrid density functionals using seminumerical integration techniques. *J. Chem. Theory Comput.*, **12**, 4254–4262, (2016).

[122] T. M. Maier; H. Bahmann; M. Kaupp. Efficient semi-numerical implementation of global and local hybrid functionals for time-dependent density functional theory. *J. Chem. Theory Comput.*, **11**, 4226–4237, (2015).

[123] F. Mack; C. J. Schattenberg; M. Kaupp; F. Weigend. Nuclear spin–spin couplings: Efficient evaluation of exact exchange and extension to local hybrid functionals. *J. Phys. Chem. A*, **124**(41), 8529–8539, (2020).

[124] C. Holzer. An improved seminumerical coulomb and exchange algorithm for properties and excited states in modern density functional theory. *J. Chem. Phys.*, **153**(18), 184115, (2020).

[125] C. Holzer; Y. J. Franzke; M. Kehry. Assessing the accuracy of local hybrid density functional approximations for molecular response properties. *J. Chem. Theory Comput.*, **17**(5), 2928–2947, (2021).

[126] R. Grotjahn; F. Furche; M. Kaupp. Development and implementation of excited-state gradients for local hybrid functionals. *J. Chem. Theory Comput.*, **15**, 5508–5522, (2019).

[127] C. J. Schattenberg; K. Reiter; F. Weigend; M. Kaupp. An efficient coupled-perturbed kohn–sham implementation of NMR chemical shift computations with local hybrid functionals and gauge-including atomic orbitals. *J. Chem. Theory Comput.*, **16**(2), 931–943, (2020).

[128] P. Plessow; F. Weigend. Seminumerical calculation of the Hartree-Fock exchange matrix: Application to two-component procedures and efficient evaluation of local hybrid density functionals. *J. Comput. Chem.*, **33**, 810, (2012).

[129] H. Bahmann; A. Rodenberg; A. V. Arbuznikov; M. Kaupp. A thermochemically competitive local hybrid functional without gradient corrections. *J. Chem. Phys.*, **126**, 011103, (2007).

[130] A. V. Arbuznikov; M. Kaupp. Local hybrid exchange-correlation functionals based on the dimensionless density gradient. *Chem. Phys. Lett.*, **440**, 160–168, (2007).

[131] A. V. Arbuznikov; M. Kaupp. Importance of the correlation contribution for local hybrid functionals: Range separation and self-interaction corrections. *J. Chem. Phys.*, **136**, 014111, (2012).

[132] A. V. Arbuznikov; M. Kaupp. Towards improved local hybrid functionals by calibration of exchange-energy densities. *J. Chem. Phys.*, **141**, 204101, (2014).

[133] T. M. Maier; M. Haasler; A. V. Arbuznikov; M. Kaupp. New approaches for the calibration of exchange-energy densities in local hybrid functionals. *Phys. Chem. Chem. Phys.*, **18**, 21133–21144, (2016).

[134] M. Haasler; T. M. Maier; R. Grotjahn; S. Gückel; A. V. Arbuznikov; M. Kaupp. A local hybrid functional with wide applicability and good balance between (de)localization and left–right correlation. *jctc*, **16**, 5645–5657, (2020).

[135] A. D. Becke. Density-functional thermochemistry. IV. A new dynamical correlation functional and implications for exact-exchange mixing. *J. Chem. Phys.*, **104**, 1040–1046, (1996).

[136] J. P. Perdew; V. N. Staroverov; J. Tao; G. E. Scuseria. Density functional with full exact exchange, balanced nonlocality of correlation, and constraint satisfaction. *Phys. Rev. A*, **78**, 052513, (2008).

[137] E. R. Johnson. Local-hybrid functional based on the correlation length. *J. Chem. Phys.*, **141**(12), 124120, (2014).

[138] C. Holzer; Y. J. Franzke. A local hybrid exchange functional approximation from first principles. *J. Chem. Phys.*, **157**(3), 034108, (2022).

[139] K. Theilacker; A. V. Arbuznikov; M. Kaupp. Gauge effects in local hybrid functionals evaluated for weak interactions and the gmtkn30 test set. *Mol. Phys.*, **114**, 1118, (2016).

[140] M. K. Armbruster; F. Weigend; C. van Wüllen; W. Klopper. Self-consistent treatment of spin-orbit interactions with efficient hartree-fock and density functional methods. *Phys. Chem. Chem. Phys.*, **10**, 1748–1756, (2008).

[141] D. Peng; M. Reiher. Exact decoupling of the relativistic fock operator. *Theor. Chem. Acc.*, **131**, 1081, (2012).

[142] D. Peng; M. Reiher. Local relativistic exact decoupling. *J. Chem. Phys.*, **136**, 244108, (2012).

[143] D. Peng; N. Middendorf; F. Weigend; M. Reiher. An efficient implementation of two-component relativistic exact-decoupling methods for large molecules. *J. Chem. Phys.*, **138**, 184105, (2013).

[144] L. Visscher; K. G. Dyall. Dirac-fock atomic electronic structure calculations using different nuclear charge distributions. *Atom. Data Nucl. Data Tabl.*, **67**, 207, (1997).

[145] J. C. Boettger. Approximate two-electron spin-orbit coupling term for density-functional-theory dft calculations using the douglas-kroll-hess transformation. *Phys. Rev. B*, **62**, 7809–7815, (2000).

[146] M. Filatov; W. Zou; D. Cremer. Spin-orbit coupling calculations with the two-component normalized elimination of the small component method. *J. Chem. Phys.*, **139**, 014106, (2013).

[147] W. Zou; M. Filatov; D. Cremer. Analytical energy gradient for the two-component normalized elimination of the small component method. *J. Chem. Phys.*, **142**, 214106, (2015).

[148] Y. J. Franzke; N. Middendorf; F. Weigend. Efficient implementation of one- and two-component analytical energy gradients in exact two-component theory. *J. Chem. Phys.*, **148**, 104110, (2018).

[149] F. Weigend; A. Baldes. Segmented contracted basis sets for one- and two-component Dirac-Fock effective core potentials. *J. Chem. Phys.*, **133**, 174102, (2010).

[150] M. Dolg; H. Stoll; H. Preuss. Energy-adjusted ab initio pseudopotentials for the rare earth elements. *J. Chem. Phys.*, **90**, 1730, (1989).

[151] R. Gulde; P. Pollak; F. Weigend. Error-Balanced Segmented Contracted Basis Sets of Double-Zeta to Quadruple-Zeta Valence Quality for the Lanthanides. *J. Chem. Theory Comput.*, **8**, 4062, (2012).

[152] W. Küchle; M. Dolg; H. Stoll; H. Preuß. Energy-adjusted pseudopotentials for the actinides. parameter sets and test calculations for thorium and thorium monoxide. *J. Chem. Phys.*, **100**, 7535, (1994).

[153] X. Cao; M. Dolg; H. Stoll. Valence basis sets for relativistic energy-consistent small-core actinide pseudopotentials. *J. Chem. Phys.*, **118**, 487, (2003).

[154] A. Baldes; F. Weigend. Efficient two-component self-consistent field procedures and gradients: implementation in turbomole and application to Au20-. *Mol. Phys.*, **111**, 2617–2624, (2013).

[155] P. Pollak; F. Weigend. Segmented contracted error-consistent basis sets of double- and triple-ζ valence quality for one- and two-component relativistic all-electron calculations. *J. Chem. Theory Comput.*, **13**, 3696 – 3705, (2017).

[156] Y. J. Franzke; R. Treß; T. M. Pazdera; F. Weigend. Error-consistent segmented contracted all-electron relativistic basis sets of double- and triple-zeta quality for NMR shielding constants. *Phys. Chem. Chem. Phys.*, **21**, 16658–16664, (2019).

[157] Y. J. Franzke; L. Spiske; P. Pollak; F. Weigend. Segmented contracted error-consistent basis sets of quadruple-ζ valence quality for one- and two-component relativistic all-electron calculations. *J. Chem. Theory Comput.*, **16**, 5658–5674, (2020).

[158] M. Reiher; A. Wolf. Exact decoupling of the Dirac Hamiltonian. I. General theory. *J. Chem. Phys.*, **121**, 2037–2047, (2004).

[159] M. Reiher; A. Wolf. Exact decoupling of the Dirac Hamiltonian. II. The generalized Douglas-Kroll-Hess transformation up to arbitrary order. *J. Chem. Phys.*, **121**, 10945–10956, (2004).

[160] M. Reiher. Douglas-Kroll-Hess Theory: a relativistic electrons-only theory for chemistry. *Theor. Chem. Acc.*, **116**, 241–252, (2006).

[161] J. Hepburn; G. Scoles; R. Penco. A simple but reliable method for the prediction of intermolecular potentials. *Chem. Phys. Lett.*, **36**, 451–456, (1975).

[162] R. Ahlrichs; R. Penco; G. Scoles. Intermolecular forces in simple systems. *Chem. Phys.*, **19**, 119–130, (1977).

[163] S. Grimme. Accurate Description of van der Waals Complexes by Density Functional Theory Including Empirical Corrections. *J. Comput. Chem.*, **25**(12), 1463–1473, (2004).

[164] S. Grimme. Semiempirical GGA-type density functional constructed with a long-range dispersion contribution. *J. Comput. Chem.*, **27**(15), 1787–1799, (2006).

[165] S. Grimme; J. Antony; S. Ehrlich; H. Krieg. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *J. Chem. Phys.*, **132**, 154104, (2010).

[166] E. Caldeweyher; C. Bannwarth; S. Grimme. Extension of the d3 dispersion coefficient model. *J. Chem. Phys.*, **147**, 034112, (2017).

[167] E. Caldeweyher; S. Ehlert; A. Hansen; H. Neugebauer; S. Spicher; C. Bannwarth; S. Grimme. A generally applicable atomic-charge dependent london dispersion correction. *J. Chem. Phys.*, **150**, 154122, (2019).

[168] S. Grimme; S. Ehrlich; L. Goerigk. Effect of the damping function in dispersion corrected density functional theory. *J. Comput. Chem.*, **32**, 1456–1465, (2011).

[169] O. A. Vydrov; T. V. Voorhis. Nonlocal van der waals density functional: The simpler the better. *J. Chem. Phys.*, **133**, 244103, (2010).

[170] W. Hujo; S. Grimme. Comparison of the performance of dispersion-corrected density functional theory for weak hydrogen bonds. *Phys. Chem. Chem. Phys.*, **13**, 13942–13950, (2011).

[171] P. Su; H. Li. Energy decomposition analysis of covalent bonds and intermolecular interactions. *J. Chem. Phys.*, **131**, 014102, (2009).

[172] R. Łazarski; A. M. Burow; M. Sierka. Density functional theory for molecular and periodic systems using density fitting and continuous fast multipole methods. *J. Chem. Theory Comput.*, **11**, 3029–3041, (2015).

[173] R. Łazarski; A. M. Burow; L. Grajciar; M. Sierka. Density functional theory for molecular and periodic systems using density fitting and continuous fast multipole method: Analytical gradients. *J. Comput. Chem.*, **37**(28), 2518–2526, (2016).

[174] A. M. Burow; M. Sierka. Linear scaling hierarchical integration scheme for the exchange-correlation term in molecular and periodic systems. *J. Chem. Theory Comput.*, **7**, 3097–3104, (2011).

[175] L. Grajciar. Low-memory iterative density fitting. *J. Comput. Chem.*, **36**, 1521–1535, (2015).

[176] A. M. Burow; M. Sierka; F. Mohamed. Resolution of identity approximation for the coulomb term in molecular and periodic systems. *J. Chem. Phys.*, **131**, 214101, (2009).

[177] C. Müller; M. Sharma; M. Sierka. Real-time time-dependent density functional theory using density fitting and the continuous fast multipole method. *Journal of Computational Chemistry*, **41**(30), 2573–2582, (2020).

[178] G. Kresse; J. Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Comput. Mat. Sci.*, **6**, 15–50, (1996).

[179] M. F. Peintinger; D. V. Oliveira; T. Bredow. Consistent gaussian basis sets of triple-zeta valence with polarization quality for solid-state calculations. *J. Comput. Chem.*, **34**, 451–459, (2013).

[180] F. Furche; D. Rappoport. Density functional methods for excited states: equilibrium structure and electronic spectra. In M. Olivucci, Ed., *Computational Photochemistry*, Band 16 von *Computational and Theoretical Chemistry*, Kapitel III. Elsevier, Amsterdam, 2005.

[181] F. Furche. On the density matrix based approach to time-dependent density functional theory. *J. Chem. Phys.*, **114**(14), 5982–5992, (2001).

[182] F. Furche; K. Burke. Time-dependent density functional theory in quantum chemistry. *Annual Reports in Computational Chemistry*, **1**, 19–30, (2005).

[183] D. Rappoport; F. Furche. Excited states and photochemistry. In M. A. L. Marques; C. A. Ullrich; F. Nogueira; A. Rubio; K. Burke; E. K. U. Gross, Eds., *Time-Dependent Density Functional Theory*, Kapitel 22. Springer, 2005.

[184] S. M. Parker; D. Rappoport; F. Furche. Quadratic Response Properties from TDDFT: Trials and Tribulations. *J. Chem. Theory Comput.*, **14**(2), 807–819, (2018).

[185] J. E. Bates; F. Furche. Harnessing the meta-generalized gradient approximation for time-dependent density functional theory. *J. Chem. Phys.*, **137**, 164105, (2012).

[186] S. Grimme; F. Furche; R. Ahlrichs. An improved method for density functional calculations of the frecuency-dependent optical rotation. *Chem. Phys. Lett.*, **361**(3–4), 321–328, (2002).

[187] H. Weiss; R. Ahlrichs; M. Häser. A direct algorithm for self-consistent-field linear response theory and application to C_{60}: Excitation energies, oscillator strengths, and frequency-dependent polarizabilities. *J. Chem. Phys.*, **99**(2), 1262–1270, (1993).

[188] D. Rappoport; F. Furche. Lagrangian approach to molecular vibrational raman intensities using time-dependent hybrid density functional theory. *J. Chem. Phys.*, **126**(20), 201104, (2007).

[189] S. M. Parker; S. Roy; F. Furche. Multistate hybrid time-dependent density functional theory with surface hopping accurately captures ultrafast thymine photodeactivation. *Phys. Chem. Chem. Phys.*, **21**, 18999–19010, (2019).

[190] R. Bauernschmitt. *Statische und dynamische Aspekte des Kohn-Sham-Formalismus: Stabilität, statischer Response, Anregungsenergien.* PhD thesis, Universität Karlsruhe, 1997.

[191] *Calculation of NMR and EPR Parameters: Theory and Applications.* M. Kaupp; M. Bühl; V. G. Malkin, Eds. Wiley-VCH Verlag GmbH & Co. KGaA: Weinheim, 2004.

[192] Y. J. Franzke. Reducing exact two-component theory for NMR couplings to a one-component approach: Efficiency and accuracy. *J. Chem. Theory Comput.*, **19**(7), 2010–2028, (2023).

[193] Y. J. Franzke; F. Mack; F. Weigend. NMR indirect spin–spin coupling constants in a modern quasi-relativistic density functional framework. *J. Chem. Theory Comput.*, **17**(7), 3974–3994, (2021).

[194] Y. J. Franzke; C. Holzer; F. Mack. NMR coupling constants based on the bethe–salpeter equation in the *GW* approximation. *J. Chem. Theory Comput.*, **18**(2), 1030–1045, (2022).

[195] F. Furche. *Dichtefunktionalmethoden für elektronisch angeregte Moleküle. Theorie–Implementierung–Anwendung.* PhD thesis, Universität Karlsruhe, 2002.

[196] E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comp. Phys.*, **17**(1), 87–94, (1975).

[197] F. Wang; T. Ziegler. Time-dependent density functional theory based on a noncollinear formulation of the exchange-correlation potential. *J. Chem. Phys.*, **121**(24), 12191–12196, (2004).

[198] M. Kühn; F. Weigend. Phosphorescence energies of organic light-emitting diodes from spin-flip Tamm-Dancoff approximation time-dependent density functional theory. *Chem. Phys. Chem.*, **12**, 3331–3336, (2011).

[199] M. Kühn; F. Weigend. Phosphorescence lifetimes of organic light-emitting diodes from two-component time-dependent density functional theory. *J. Chem. Phys.*, **141**(22), 224302, (2014).

[200] N. F. Ramsey. Electron coupled interactions between nuclear spins in molecules. *Phys. Rev.*, **91**, 303–307, (1953).

[201] F. Jensen. The optimum contraction of basis sets for calculating spin–spin coupling constants. *Theor. Chem. Acc.*, **126**(5), 371–382, (2010).

[202] P. A. Aggelund; S. P. A. Sauer; F. Jensen. Development of polarization consistent basis sets for spin-spin coupling constant calculations for the atoms Li, Be, Na, and Mg. *J. Chem. Phys.*, **149**(4), 044117, (2018).

[203] K. G. Dyall. Relativistic and nonrelativistic finite nucleus optimized triple-zeta basis sets for the 4p, 5p and 6p elements. *Theor. Chem. Acc.*, **108**(6), 335–340, (Dec 2002).

[204] K. G. Dyall. Relativistic quadruple-zeta and revised triple-zeta and double-zeta basis sets for the 4p, 5p, and 6p elements. *Theor. Chem. Acc.*, **115**(5), 441–447, (May 2006).

[205] K. G. Dyall. Relativistic double-zeta, triple-zeta, and quadruple-zeta basis sets for the light elements h–ar. *Theor. Chem. Acc.*, **135**(5), 128, (2016).

[206] Y. J. Franzke. *Calculation of NMR Parameters in a Modern Relativistic Density Functional Framework: Theory, Implementation, and Application.* Dissertation, Karlsruhe Institute of Technology (KIT), Germany, 2021.

[207] F. Mack. *Effiziente Implementierung und Berechnung von NMR-Kopplungskonstanten im Rahmen von nichtrelativistischer und quasirelativistischer Theorie*. Dissertation, Karlsruher Institut für Technologie (KIT), Germany, 2021.

[208] G. Giannone; F. Della Sala. Minimal auxiliary basis set for time-dependent density functional theory and comparison with tight-binding approximations: Application to silver nanoparticles. *J. Chem. Phys.*, **153**(8), 084110, (2020).

[209] Z. Zhou; F. Della Sala; S. M. Parker. Minimal auxiliary basis set approach for the electronic excitation spectra of organic molecules. *J. Phys. Chem. Lett.*, **14**(7), 1968–1976, (2023).

[210] S. Grimme. A simplified tamm-dancoff density functional approach for the electronic excitation spectra of very large molecules. *J. Chem. Phys.*, **138**(24), 244104, (Juni 2013).

[211] C. Bannwarth; S. Grimme. A simplified time-dependent density functional theory approach for electronic ultraviolet and circular dichroism spectra of very large molecules. *Comput. Theoret. Chem.*, **1040-1041**, 45, (2014).

[212] R. Rüger; E. van Lenthe; T. Heine; L. Visscher. Tight-binding approximations to time-dependent density functional theory — a fast approach for the calculation of electronically excited states. *J. Chem. Phys.*, **144**(18), 184103, (2016).

[213] F. Haase; R. Ahlrichs. Semidirect MP2 gradient evaluation on workstation computers: The MPGRAD program. *J. Comp. Chem.*, **14**(8), 907–912, (1993).

[214] F. Weigend; M. Häser. RI-MP2: first derivatives and global consistency. *Theor. Chem. Acc.*, **97**(1–4), 331–340, (1997).

[215] F. Weigend; M. Häser; H. Patzelt; R. Ahlrichs. RI-MP2: Optimized auxiliary basis sets and demonstration of efficiency. *Chem. Phys. Letters*, **294**(1–3), 143–152, (1998).

[216] F. Weigend; A. Köhn; C. Hättig. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. *J. Chem. Phys.*, **116**(8), 3175–3183, (2001).

[217] C. L. Janssen; I. M. B. Nielsen. New diagnostics for coupled-cluster and Møller-Plesset perturbation theory. *Chem. Phys. Lett.*, **290**(4–6), 423–430, (1998).

[218] I. M. B. Nielsen; C. L. Janssen. Double-substitution-based diagnostics for coupled-cluster and Møller-Plesset perturbation theory. *Chem. Phys. Lett.*, **310**(5–6), 568–576, (1999).

[219] F. A. Bischoff; S. Höfener; A. Glöß; W. Klopper. Explicitly correlated second-order perturbation theory calculations on molecules containing heavy main-group elements. *Theor. Chem. Acc.*, **121**(1), 11–19, (2008).

[220] D. P. Tew. Explicitly correlated coupled-cluster theory with brueckner orbitals. *J. Chem. Phys.*, **145**(7), 074103, (2016).

[221] F. R. Manby. Density fitting in second-order linear-r_{12} Møller-Plesset perturbation theory. *J. Chem. Phys.*, **119**(9), 4607–4613, (2003).

[222] E. F. Valeev. Improving on the resolution of the identity in linear R12 ab initio theories. *Chem. Phys. Lett.*, **395**(4-6), 190–195, (2004).

[223] K. E. Yousaf; K. A. Peterson. Optimized auxiliary basis sets for explicitly correlated methods. *J. Chem. Phys.*, **129**(18), 184108, (2008).

[224] K. A. Peterson; T. B. Adler; H.-J. Werner. Systematically convergent basis sets for explicitly correlated wavefunctions: The atoms H, He, B–Ne, and Al–Ar. *J. Chem. Phys.*, **128**(8), 084102, (2008).

[225] W. Klopper; C. C. M. Samson. Explicitly correlated second-order Møller-Plesset methods with auxiliary basis sets. *J. Chem. Phys.*, **116**(15), 6397–6410, (2002).

[226] W. Klopper; W. Kutzelnigg. Møller-Plesset calculations taking care of the correlation cusp. *Chem. Phys. Lett.*, **134**(1), 17–22, (1987).

[227] S. Ten-no. Explicitly correlated second order perturbation theory: Introduction of a rational generator and numerical quadratures. *J. Chem. Phys.*, **121**(1), 117–129, (2004).

[228] S. F. Boys. Localized orbitals and localized adjustment functions. In P.-O. Löwdin, Ed., *Quantum Theory of Atoms, Molecules and the Solid State*, Page 253. Academic Press, New York, 1966.

[229] J. Pipek; P. G. Mezey. A fast intrinsic localization procedure applicable for ab initio and semiempirical linear combination of atomic orbital wave functions. *J. Chem. Phys.*, **90**(9), 4916–4926, (1989).

[230] D. P. Tew; W. Klopper. New correlation factors for explicitly correlated electronic wave functions. *J. Chem. Phys.*, **123**(7), 074101, (2005).

[231] W. Klopper; B. Ruscic; D. P. Tew; F. A. Bischoff; S. Wolfsegger. Atomization energies from coupled-cluster calculations augmented with explicitly-correlated perturbation theory. *Chem. Phys.*, **356**(1–3), 14–24, (2009).

[232] S. Höfener; F. A. Bischoff; A. Glöß; W. Klopper. Slater-type geminals in explicitly-correlated perturbation theory: application to n-alkanols and analysis of errors and basis-set requirements. *Phys. Chem. Chem. Phys.*, **10**(23), 3390–3399, (2008).

[233] O. Christiansen; H. Koch; P. Jørgensen. The second-order approximate coupled cluster singles and doubles model CC2. *Chem. Phys. Lett.*, **243**(5–6), 409–418, (1995).

[234] W. Klopper; F. R. Manby; S. Ten-no; E. F. Valeev. R12 methods in explicitly correlated molecular electronic structure theory. *Int. Rev. Phys. Chem.*, **25**(3), 427–468, (2006).

[235] C. Hättig; A. Köhn. Transition moments and excited state first-order properties in the second-order coupled cluster model CC2 using the resolution of the identity approximation. *J. Chem. Phys.*, **117**(15), 6939–6951, (2002).

[236] T. Helgaker; P. Jørgensen; J. Olsen. *Molecular Electronic-Structure Theory.* Wiley: New York, 2000.

[237] O. Christiansen; P. Jørgensen; C. Hättig. Response functions from Fourier component variational perturbation theory applied to a time-averaged quasienergy. *Int. J. Quantum Chem.*, **68**(1), 1–52, (1998).

[238] L. S. Cederbaum; W. Domcke; J. Schirmer. Many-body theory of core holes. *Phys. Rev. A*, **22**, 206–222, (1980).

[239] C. Hättig; P. Jørgensen. Derivation of coupled cluster excited states response functions and multiphoton transition moments between two excited states as derivatives of variational functionals. *J. Chem. Phys.*, **109**(21), 9219–9236, (1998).

[240] C. Hättig; O. Christiansen; P. Jørgensen. Multiphoton transition moments and absorption cross section in coupled cluster response theory employing variational transition moment functionals. *J. Chem. Phys.*, **108**(20), 8331–8354, (1998).

[241] H. Koch; P. Jørgensen. Coupled cluster response functions. *J. Chem. Phys.*, **93**, 3333, (1990).

[242] C. Hättig; O. Christiansen; P. Jørgensen. Coupled cluster response calculations of twophoton transition probability rate constants for helium, neon and argon. *J. Chem. Phys.*, **108**(20), 8355–8359, (1998).

[243] D. H. Friese; C. Hättig; K. Ruud. Calculation of two-photon absorption strengths with the approximate coupled cluster singles and doubles model cc2 using the resolution-of-identity approximation. *Phys. Chem. Chem. Phys.*, **14**, 1175–1184, (2012).

[244] N. K. Graf; D. H. Friese; N. O. C. Winter; C. Hättig. Excited state polarizabilities for CC2 using the resolution-of-the-identity approximation. *J. Chem. Phys.*, **143**(24), 244108, (2015).

[245] B. Helmich-Paris; C. Hättig; C. van Wüllen. Spin-free cc2 implementation of induced transitions between singlet ground and triplet excited states. *J. Chem. Theory Comput.*, **12**(4), 1892–1904, (2016).

[246] C. Hättig. Structure optimizations for excited states with correlated second-order methods: CC2, CIS(D∞), and ADC(2). *Adv. Quant. Chem.*, **50**, 37–60, (2005).

[247] S. Grimme; E. Ugorodina. Calculation of 0-0 excitation energies of organic molecules by CIS(D) quantum chemical methods. *Chem. Phys.*, **305**, 223–230, (2004).

[248] Y. M. Rhee; M. Head-Gordon. Scaled second–order perturbation corrections to configuration interaction singles: Efficient and reliable excitation energy methods. *J. Phys. Chem. A*, **111**, 5314–5326, (2007).

[249] H. Fliegl; C. Hättig; W. Klopper. Coupled–cluster theory with simplified linear-r_{12} corrections: The CCSD(R12) model. *J. Chem. Phys.*, **122**, 084107, (2005).

[250] T. Shiozaki; M. Kamiya; S. Hirata; E. F. Valeev. Explicitly correlated coupled-cluster singles and doubles method based on complete diagrammatic equations. *J. Chem. Phys.*, **129**, 071101, (2008).

[251] A. Köhn; G. W. Richings; D. P. Tew. Implementation of the full explicitly correlated coupled-cluster singles and doubles model CCSD-F12 with optimally reduced auxiliary basis dependence. *J. Chem. Phys.*, **129**, 201103, (2008).

[252] T. B. Adler; G. Knizia; H.-J. Werner. A simple and efficient ccsd(t)-f12 approximation. *J. Chem. Phys.*, **127**, 221106, (2007).

[253] G. Knizia; T. B. Adler; H.-J. Werner. Simplified ccsd(t)-f12 methods: Theory and benchmarks. *J. Chem. Phys.*, **130**, 054104, (2009).

[254] M. Torheyden; E. F. Valeev. Variational formulation of perturbative explicitly-correlated coupled-cluster methods. *Phys. Chem. Chem. Phys.*, **10**, 3410–3420, (2008).

[255] E. F. Valeev; D. Crawford. Simple coupled-cluster singles and doubles method with perturbative inclusion of triples and explicitly correlated geminals: The ccsd(t)$_{r\bar{1}2}$ model. *J. Chem. Phys.*, **128**, 244113, (2008).

[256] K. D. Vogiatzis; E. C. Barnes; W. Klopper. Interference-corrected explicitly-correlated second-order perturbation theory. *Chem. Phys. Lett.*, **503**(1-3), 157–161, (2011).

[257] D. P. Tew. Principal domains in local correlation theory. *J. Chem. Theory Comput.*, **15**(12), 6597–6606, (2019).

[258] H. Eshuis; J. Yarkony; F. Furche. Fast computation of molecular random phase approximation correlation energies using resolution of the identity and imaginary frequency integration. *J. Chem. Phys.*, **132**, 234114, (2010).

[259] H. Eshuis; J. E. Bates; F. Furche. Electron correlation methods based on the random phase approximation. *Theor. Chem. Acc.*, **131**, 1084, (2012).

[260] A. M. Burow; J. E. Bates; F. Furche; H. Eshuis. Analytical first-order molecular properties and forces within the adiabatic connection random phase approximation. *J. Chem. Theory Comput.*, **10**(1), 180–194, (2014).

[261] G. P. Chen; V. K. Voora; M. M. Agee; S. G. Balasubramani; F. Furche. Random-phase approximation methods. *Annu. Rev. Phys. Chem.*, **68**, 421–445, (2017).

[262] M. Kühn. Correlation Energies from the Two-component Random Phase Approximation. *J. Chem. Theory Comput.*, **10**, 623–633, (2014).

[263] J. E. Bates; F. Furche. Random phase approximation renormalized many-body perturbation theory. *J. Chem. Phys.*, **139**(17), 171103, (2013).

[264] G. P. Chen; M. M. Agee; F. Furche. Performance and scope of perturbative corrections to random-phase approximation energies. *J. Chem. Theory Comput.*, **14**(11), 5701–5714, (2018).

[265] V. K. Voora; S. G. Balasubramani; F. Furche. Variational generalized kohn-sham approach combining the random-phase-approximation and green's-function methods. *Phys. Rev. A*, **99**, 012518, (2019).

[266] E. Trushin; A. Thierbach; A. Görling. Toward chemical accuracy at low computational cost: Density-functional theory with σ-functionals for the correlation energy. *J. Chem. Phys.*, **154**(1), 014104, (2021).

[267] S. Fauser; E. Trushin; C. Neiss; A. Görling. Chemical accuracy with σ-functionals for the kohn–sham correlation energy optimized for different input orbitals and eigenvalues. *J. Chem. Phys.*, **155**(13), 134111, (2021).

[268] J. Erhard; S. Fauser; E. Trushin; A. Görling. Scaled σ-functionals for the kohn–sham correlation energy with scaling functions from the homogeneous electron gas. *J. Chem. Phys.*, **157**(11), 114105, (2022).

[269] C. Neiss; S. Fauser; A. Görling. Geometries and vibrational frequencies with kohn–sham methods using σ-functionals for the correlation energy. *J. Chem. Phys.*, **158**(4), 044107, (2023).

[270] F. Furche. Molecular tests of the random phase approximation to the exchange-correlation energy functional. *Phys. Rev. B*, **64**, 195120, (2001).

[271] F. Furche. Developing the random phase approximation into a practical post-Kohn–Sham correlation model. *J. Chem. Phys.*, **129**, 114105, (2008).

[272] V. K. Voora; R. Galhenage; J. C. Hemminger; F. Furche. Effective one-particle energies from generalized kohn–sham random phase approximation: A direct approach for computing and analyzing core ionization energies. *The Journal of Chemical Physics*, **151**(13), 134106, (2019).

[273] I. Duchemin; X. Blase. Robust Analytic-Continuation Approach to Many-Body GW Calculations. *J. Chem. Theory Comput.*, **16**(3), 1742–1756, (2020).

[274] P. Deglmann; F. Furche; R. Ahlrichs. An efficient implementation of second analytical derivatives for density functional methods. *Chem. Phys. Lett.*, **362**(5–6), 511–518, (2002).

[275] P. Deglmann; F. Furche. Efficient characterization of stationary points on potential energy surfaces. *J. Chem. Phys.*, **117**(21), 9535–9538, (2002).

[276] M. Bürkle; J. Viljas; T. Hellmuth; E. Scheer; F. Weigend; G. Schön; F. Pauly. Influence of vibrations on electron transport through nanoscale contacts. *Phys. Status Solidi B*, **250**, 2468, (2013).

[277] S. Gillhuber; Y. J. Franzke; F. Weigend. Paramagnetic NMR shielding tensors and ring currents: Efficient implementation and application to heavy element compounds. *J. Phys. Chem. A*, **125**(44), 9707–9723, (2021).

[278] T. Ziegler; G. Schreckenbach. Calculation of NMR shielding tensors using gauge-including atomic orbitals and modern density functional theory. *J. Phys. Chem.*, **99**(2), 606–611, (1995).

[279] S. N. Maximoff; G. E. Scuseria. Nuclear magnetic resonance shielding tensors calculated with kinetic energy density-dependent exchange-correlation functionals. *Chem. Phys. Lett.*, **390**(4), 408–412, (2004).

[280] C. J. Schattenberg; M. Kaupp. Effect of the current dependence of tau-dependent exchange-correlation functionals on nuclear shielding calculations. *J. Chem. Theory Comput.*, **17**(3), 1469–1479, (2021).

[281] Y. J. Franzke; C. Holzer. Communication: Impact of the current density on paramagnetic NMR properties. *J. Chem. Phys.*, **157**(3), 031102, (2022).

[282] S. K. Wolff; T. Ziegler; E. van Lenthe; E. J. Baerends. Density functional calculations of nuclear magnetic shieldings using the zeroth-order regular approximation (ZORA) for relativistic effects: ZORA nuclear magnetic resonance. *J. Chem. Phys.*, **110**(16), 7689–7698, (1999).

[283] M. Kollwitz; M. Häser; J. Gauss. Non-abelian point group symmetry in direct second-order many-body perturbation theory calculations of NMR chemical shifts. *J. Chem. Phys.*, **108**(20), 8295–8301, (1998).

[284] C. van Wüllen. On the use of effective core potentials in the calculation of magnetic properties, such as magnetizabilites and magnetic shieldings. *J. Chem. Phys.*, **136**(11), 114110, (2012).

[285] Z. Rinkevicius; J. Vaara; L. Telyatnyk; O. Vahtras. Calculations of nuclear magnetic shielding in paramagnetic molecules. *J. Chem. Phys.*, **118**, 2550–2561, (2003).

[286] Y. J. Franzke; J. M. Yu. Hyperfine coupling constants in local exact two-component theory. *J. Chem. Theory Comput.*, **18**(1), 323–343, (2022).

[287] Y. J. Franzke; J. M. Yu. Quasi-relativistic calculation of EPR g tensors with derivatives of the decoupling transformation, gauge-including atomic orbitals, and magnetic balance. *J. Chem. Theory Comput.*, **18**(4), 2246–2266, (2022).

[288] B. Martin; J. Autschbach. Temperature dependence of contact and dipolar nmr chemical shifts in paramagnetic molecules. *J. Chem. Phys.*, **142**(5), 054108, (2015).

[289] J. Jusélius; D. Sundholm; J. Gauss. Calculation of current densities using gauge-including atomic orbitals. *J. Chem. Phys.*, **121**(9), 3952–3963, (2004).

[290] H. Fliegl; S. Taubert; O. Lehtonen; D. Sundholm. The gauge including magnetically induced current method. *Phys. Chem. Chem. Phys.*, **13**, 20500–20518, (2011).

[291] D. Sundholm; H. Fliegl; R. J. F. Berger. Calculations of magnetically induced current densities: theory and applications. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, **6**(6), 639–678, (2016).

[292] S. Lehtola; M. Dimitrova; H. Fliegl; D. Sundholm. Benchmarking magnetizabilities with recent density functionals. *J. Chem. Theory Comput.*, **17**(3), 1457–1468, (2021).

[293] F. Bruder; Y. J. Franzke; C. Holzer; F. Weigend. Zero-field splitting parameters within exact two-component theory and modern density functional theory using seminumerical integration. *ChemRxiv*, (2023).

[294] S. G. Chiodo; M. Leopoldini. Molsoc: A spin–orbit coupling code. *Comput. Phys. Commun.*, **185**(2), 676–683, (2014).

[295] S. Koseki; M. W. Schmidt; M. S. Gordon. MCSCF/6-31G(d,p) calculations of one-electron spin-orbit coupling constants in diatomic molecules. *J. Phys. Chem.*, **96**(26), 10768–10772, (1992).

[296] F. Neese. Efficient and accurate approximations to the molecular spin-orbit coupling operator and their use in molecular g-tensor calculations. *J. Chem. Phys.*, **122**, 034107, (2005).

[297] S. Schmitt; P. Jost; C. van Wüllen. Zero-field splittings from density functional calculations: Analysis and improvement of known methods. *J. Chem. Phys.*, **134**(19), 194113, (2011).

[298] S. Sinnecker; F. Neese. Spin–spin contributions to the zero-field splitting tensor in organic triplets, carbenes and biradicals – A density functional and ab initio study. *J. Phys. Chem. A*, **110**(44), 12267–12275, (2006).

[299] A. Klamt; G. Schüürmann. COSMO: A new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient. *J. Chem. Soc. Perkin Trans.2*, (5), 799–805, (1993).

[300] A. Klamt; C. Moya; J. Palomar. A comprehensive comparison of the iefpcm and ss(v)pe continuum solvation methods with the cosmo approach. *J. Chem. Theory Comput.*, **11**(9), 4220–4225, (2015).

[301] A. Klamt; M. Diedenhofen. A refined cavity construction algorithm for the conductor-like screening model. *J. Comput. Chem.*, **39**(21), 1648–1655, (2018).

[302] A. Klamt; V. Jonas. Treatment of the outlying charge in continuum solvation models. *J. Chem. Phys.*, **105**(22), 9972–9981, (1996).

[303] A. Klamt. Calculation of UV/Vis spectra in solution. *J. Phys. Chem.*, **100**(9), 3349–3353, (1996).

[304] F. J. Olivares del Valle; J. Tomasi. Electron correlation and solvation effects. I. Basic formulation and preliminary attempt to include the electron correlation in the quantum mechanical polarizable continuum model so as to study solvation phenomena. *Chem. Phys.*, **150**(2), 139–150, (1991).

[305] J. G. Ángyán. Rayleigh-Schrödinger perturbation theory for nonlinear Schrödinger equations with linear perturbation. *Int. J. Quantum Chem.*, **47**(6), 469–483, (1993).

[306] J. G. Ángyán. Choosing between alternative MP2 algorithms in the self-consistent reaction field theory of solvent effects. *Chem. Phys. Lett.*, **241**(1–2), 51–56, (1995).

[307] R. Cammi; B. Mennucci; J. Tomasi. Second-order Møller-Plesset analytical derivatives for the polarizable continuum model using the relaxed density approach. *J. Phys. Chem. A*, **103**(45), 9100–9108, (1999).

[308] T. Schwabe; K. Sneskov; J. M. Olsen; J. Kongsted; O. Christiansen; C. Hättig. PERI-CC2: A polarizable embedded RI-CC2 method. *J. Chem. Theory Comput.*, **8**, 3274–3283, (2012).

[309] G. Scalmani; M. J. Frisch; B. Mennucci; J. Tomasi; R. Cammi; V. Barone. Geometries and properties of excited states in the gas phase and in solution: Theory and application of a time-dependent density functional theory polarizable continuum model. *J. Chem. Phys.*, **124**(9), 094107, (2006).

[310] S. Sinnecker; A. Rajendran; A. Klamt; M. Diedenhofen; F. Neese. Calculation of solvent shifts on electronic g-tensors with the Conductor-like Screening Model (COSMO) and its self-consistent generalization to real solvents (Direct COSMO-RS). *J. Phys. Chem. A*, **110**, 2235–2245, (2006).

[311] F. Eckert; A. Klamt. Fast solvent screening via quantum chemistry: COSMO-RS approach. *AICHE Journal*, **48**, 369–385, (2002).

[312] A. Klamt; V. Jonas; T. Bürger; J. C. W. Lohrenz. Refinement and parametrization of COSMO-RS. *J. Phys. Chem. A*, **102**, 5074–5085, (1998).

[313] S. K. Khani; A. M. Khah; C. Hättig. Cosmo-ri-adc(2) excitation energies and excited state gradients. *Phys. Chem. Chem. Phys.*, **20**(24), 16354–16363, (2018).

[314] S. K. Khani; R. Faber; F. Santoro; S. Coriani; C. Hättig. Uv absorption and magnetic circular dichroism spectra of purine, adenine, and guanine: A Coupled Cluster study in vacuo and in aqueous solution. *J. Chem. Theory Comput.*, **15**(2), 1242–1254, (2019).

[315] P. Cortona. Self-consistently determined properties of solids without band-structure calculations. *Phys. Rev. B*, **44**, 8454, (1991).

[316] T. A. Wesolowski; A. Warshel. Frozen density functional approach for ab initio calculations of solvated molecules. *J. Phys. Chem.*, **97**, 8050, (1993).

[317] T. A. Wesolowski. In J. Leszczynski, Ed., *Chemistry: Reviews of Current Trends*, Band 10, Page 1. World Scientific: Singapore, 2006, Singapore, 2006.

[318] T. A. Wesolowski; A. Warshel. Kohn–Sham equations with constrained electron density: an iterative evaluation of the ground-state electron density of interacting molecules. *Chem. Phys. Lett.*, **248**, 71, (1996).

[319] S. Laricchia; E. Fabiano; F. Della Sala. Frozen density embedding with hybrid functionals. *J. Chem. Phys.*, **133**, 164111, (2010).

[320] S. Laricchia; E. Fabiano; F. Della Sala. Frozen density embedding calculations with the orbital-dependent localized Hartree–Fock Kohn–Sham potential. *Chem. Phys. Lett.*, **518**, 114, (2011).

[321] R. S. Treß; C. Hättig; S. Höfener. Employing pseudopotentials to tackle excited-state electron spill-out in frozen density embedding calculations. *Journal of Chemical Theory and Computation*, **18**(3), 1737–1747, (2022).

[322] L. A. Constantin; E. Fabiano; S. Laricchia; F. Della Sala. Semiclassical neutral atom as a reference system in density functional theory. *Phys. Rev. Lett.*, **106**, 186406, (2011).

[323] S. Laricchia; E. Fabiano; L. A. Constantin; F. Della Sala. Generalized gradient approximations of the noninteracting kinetic energy from the semiclassical atom theory: Rationalization of the accuracy of the frozen density embedding theory for nonbonded interactions. *J. Chem. Theory Comput.*, **7**, 2439, (2011).

[324] A. Lembarki; H. Chermette. Obtaining a gradient-corrected kinetic-energy functional from the Perdew-Wang exchange functional. *Phys. Rev. A*, **50**, 5328, (1994).

[325] M. Sierka; A. Burow; J. Döbler; J. Sauer. Point defects in CeO_2 and CaF_2 investigated using periodic electrostatic embedded cluster method. *J. Chem. Phys.*, **130**(17), 174710, (2009).

[326] K. N. Kudin; G. E. Scuseria. A fast multipole method for periodic systems with arbitrary unit cell geometries. *Chem. Phys. Lett.*, **283**, 61–68, (1998).

[327] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, **64**, 253–287, (1921).

[328] J. M. Olsen; K. Aidas; J. Kongsted. Excited states in solution through polarizable embedding. *J. Chem. Theory Comput*, **6**, 3721–3734, (2010).

[329] K. Sneskov; T. Schwabe; J. Kongsted; O. Christiansen. The polarizable embedding coupled cluster method. *J. Chem. Phys.*, **134**, 104108, (2011).

[330] K. Sneskov; T. Schwabe; J. Kongsted; O. Christiansen. Scrutinizing the effects of polarization in QM/MM excited state calculations. *Phys. Chem. Chem. Phys.*, **13**, 18551–18560, (2011).

[331] K. Krause; W. Klopper. Communication: A simplified coupled-cluster lagrangian for polarizable embedding. *J. Chem. Phys.*, **144**, 041101, (2016).

[332] L. Gagliardi; R. Lindh; G. Karlström. Local properties of quantum chemical systems: The LoProp approach. *J. Chem. Phys.*, **121**, 4494–5000, (2004).

[333] B. Lunkenheimer; A. Köhn. Solvent effects on electronically excited states using the conductor- like screening model and the second-order correlated method adc(2). *J. Chem. Theory Comput.*, **9**, 977–994, (2015).

[334] A. M. Khah; S. K. Khani; C. Hättig. Analytic excited state gradients for the qm/mm polarizable embedded second-order algebraic diagrammatic construction for the polarization propagator pe-adc(2). *J. Chem. Theory Comput.*, **14**, 4640–4650, (2018).

[335] A. E. Reed; R. B. Weinstock; F. Weinhold. Natural population analysis. *J. Chem. Phys.*, **83**(2), 735–746, (1985).

[336] C. Ehrhardt; R. Ahlrichs. Population Analysis Based on Occupation Numbers. II. Relationship between Shared Electron Numbers and Bond Energies and Characterization of Hypervalent Contributions. *Theor. Chim. Acta*, **68**(3), 231–245, (1985).

[337] K. Wiberg. Application of the pople-santry-segal cndo method to the cyclopropylcarbinyl and cyclobutyl cation and to bicyclobutane. *Tetrahedron*, **24**(3), 1083 – 1096, (1968).

[338] I.-M. Hoyvik; B. Jansik; P. Jorgensen. Orbital localization using fourth central moment minimization. *J. Chem. Phys.*, **137**(22), (2012).

[339] G. Knizia. Intrinsic atomic orbitals: An unbiased bridge between quantum theory and chemical concepts. *J. Chem. Theory Comput.*, **9**, 4834–4843, (2013).

[340] A. V. Luzanov; A. A. Sukhorukov; V. E. Úmanskii. Application of transition density matrix for analysis of excited states. *Theor. Exp. Chem.*, **10**, 354, (1976).

[341] R. L. Martin. Natural transition orbitals. *J. Chem. Phys.*, **118**, 4775, (2003).

[342] F. Weigend; C. Schrodt. Atom-type assignment in molecule and clusters by pertubation theory— A complement to X-ray structure analysis. *Chem. Eur. J.*, **11**(12), 3559–3564, (2005).

[343] A. D. Becke; K. E. Edgecombe. A simple measure of electron localization in atomic and molecular systems. *J. Chem. Phys.*, **92**(9), 5397–5403, (1990).

[344] F. D. Sala; A. Görling. Efficient localized Hartree-Fock methods as effective exact-exchange Kohn-Sham methods for molecules. *J. Chem. Phys.*, **115**(13), 5718–5732, (2001).

[345] A. Görling. Orbital- and state-dependent functionals in density-functional theory. *J. Chem. Phys.*, **123**(6), 062203, (2005).

[346] S. Kümmel; L. Kronik. Orbital-dependent density functionals: Theory and applications. *Rev. Mod. Phys.*, **80**(1), 3, (2008).

[347] F. Della Sala. Orbital-dependent exact-exchange mmethods in density functional theory. In M. Springborg, Ed., *Chemical Modelling: Applications and Theory*, Band 7, Pages 115–161. Royal Society of Chemistry, 2010.

[348] A. Heßelmann; A. W. Götz; F. Della Sala; A. Görling. Numerically stable optimized effective potential method with balanced gaussian basis sets. *J. Chem. Phys.*, **127**(5), 054102, (2007).

[349] J. B. Krieger; Y. Li; G. J. Iafrate. Construction and application of an accurate local spin-polarized kohn-sham potential with integer discontinuity: Exchange-only theory. *Phys. Rev. A*, **45**, 101, (1992).

[350] O. V. Gritsenko; E. J. Baerends. Orbital structure of the kohn-sham exchange potential and exchange kernel and the field-counteracting potential for molecules in an electric field. *Phys. Rev. A*, **64**, 042506, (2001).

[351] A. F. Izmaylov; V. N. Staroverov; G. E. Scuseria; E. R. Davidson; G. Stoltz; E. Cancès. The effective local potential method: Implementation for molecules and relation to approximate optimized effective potential techniques. *J. Chem. Phys.*, **126**(8), 084107, (2007).

[352] F. Della Sala; A. Görling. The asymptotic region of the Kohn-Sham exchange potential in molecules. *J. Chem. Phys.*, **116**(13), 5374–5388, (2002).

[353] F. Della Sala; A. Görling. Asymptotic behavior of the Kohn-Sham exchange potential. *Phys. Rev. Lett.*, **89**, 033003, (2002).

[354] W. Hieringer; F. Della Sala; A. Görling. Density-functional calculations of NMR shielding constants using the localized Hartree–Fock method. *Chem. Phys. Lett.*, **383**(1-2), 115–121, (2004).

[355] E. Fabiano; M. Piacenza; S. D'Agostino; F. Della Sala. Towards an accurate description of the electronic properties of the biphenylthiol/gold interface: The role of exact exchange. *J. Chem. Phys.*, **131**(23), 234101, (2009).

[356] F. Della Sala; E. Fabiano. Accurate singlet and triplet excitation energies using the Localized Hartree-Fock Kohn-Sham potential. *Chem. Phys.*, **391**(1), 19 – 26, (2011).

[357] E. Tapavicza; F. Furche; D. Sundholm. Importance of vibronic effects in the uv-vis spectrum of the 7, 7, 8, 8-tetracyanoquinodimethane anion. *J. Chem. Theory Comput.*, **12**(10), 5058–5066, (2016).

[358] F. Duschinsky. The importance of the electron spectrum in multi atomic molecules. concerning the Franck-Condon principle. *Acta Physicochim. URSS*, **7**, 551, (1937).

[359] F. G. Mehler. Ueber die Entwicklung einer Function von beliebig vielen Variabeln nach Laplaceschen Functionen höherer Ordnung. *J. Reine Angew. Math.*, **66**, 161–176, (1866).

[360] M. Etinski; J. Tatchen; C. M. Marian. Time-dependent approaches for the calculation of intersystem crossing rates. *J. Chem. Phys.*, **134**(15), 154105, (2011).

[361] E. Tapavicza. Generating function approach to single vibronic level fluorescence spectra. *J. Phys. Chem. Lett.*, **10**, 6003–6009, (2019).

[362] J. J. Dongarra; J. Du Croz; S. Hammarling; I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, **16**, 1–17, (1990).

[363] W. H. Press; S. A. Teukolsky; W. T. Vetterling; B. P. Flannery. *Numerical recipes in C*. Band 2. Cambridge university press: Cambridge, 1996.

[364] R. C. Hilborn. Einstein coefficients, cross sections, f values, dipole moments, and all that. *Am. J. Phys.*, **50**(11), 982–986, (1982).

[365] O. Treutler; R. Ahlrichs. Efficient molecular numerical integration schemes. *J. Chem. Phys.*, **102**(1), 346–354, (1995).

[366] A. D. Becke. A multicenter numerical integration scheme for polyatomic molecules. *J. Chem. Phys.*, **88**(4), 2547–2553, (1988).

[367] A. T. B. Gilbert; N. A. Besley; P. M. W. Gill. Self-Consistent Field Calculations of Excited States Using the Maximum Overlap Method (MOM). *J. Phys. Chem. A*, **112**, 13164–13171, (2008).

[368] A. Wolf; M. Reiher; B. Hess. The generalized Douglas-Kroll transformation. *J. Chem. Phys.*, **117**, 9215–9226, (2002).

[369] L. Visscher; K. G. Dyall. Dirac-fock atomic electronic structure calculations using different nuclear charge distributions. *At. Data Nucl. Data Tables*, **67**, 207–224, (1997).

[370] R. Send; F. Furche. First-order nonadiabatic couplings from time-dependent hybrid density functional response theory: Consistent formalism, implementation, and performance. *J. Phys. Chem.*, **132**, 044107, (2010).

[371] C. Hättig; D. P. Tew; B. Helmich. Local explicitly correlated second- and third-order møller–plesset perturbation theory with pair natural orbitals. *J. Chem. Phys.*, **146**, 204105, (2012).

[372] J. C. Tully. Molecular dynamics with electronic transitions. *J. Chem. Phys.*, **93**, 1061, (1990).

[373] E. Tapavicza; I. Tavernelli; U. Rothlisberger. Trajectory surface hopping within linear response time-dependent density-functional theory. *Phys. Rev. Lett.*, **98**, 023001, (2007).

[374] E. Tapavicza; A. M. Meyer; F. Furche. Unravelling the details of vitamin D photosynthesis by non-adiabatic molecular dynamics simulations. *Phys. Chem. Chem. Phys.*, **13**, 20986, (2011).

[375] B. G. Levine; C. Ko; J. Quenneville; T. J. Martinez. Conical intersections and double excitations in density functional theory. *Mol. Phys.*, **104**, 1039, (2006).

[376] E. Tapavicza; I. Tavernelli; U. Rothlisberger; C. Filippi; M. E. Casida. Mixed time-dependent density-functional theory/classical trajectory surface hopping study of oxirane photochemistry. *J. Chem. Phys.*, **129**(12), 124108, (2008).

Index