# Manual for the MRCC Program System

Version: 25.1.0

MŰEGYETEM 1782

Department of Physical Chemistry and Materials Science
Budapest University of Technology and Economics

https://www.mrcc.hu/

# Contents

1 Introduction 4

2 How to read this manual 4

3 Authors 4

4 Citation 5

5 Interfaces 6
- 5.1 Cfour . . . 6
- 5.2 Columbus . . . 7
- 5.3 Dirac . . . 7
- 5.4 Molpro . . . 8
- 5.5 Amber . . . 8
- 5.6 xTB . . . 9
- 5.7 Mopac . . . 9

6 Features 9
- 6.1 Single-point energy calculations . . . 10
- 6.2 Geometry optimizations and first-order properties . . . 15
- 6.3 Harmonic frequencies and second-order properties . . . 17
- 6.4 Higher-order properties . . . 18
- 6.5 Diagonal Born–Oppenheimer corrections . . . 19
- 6.6 Electronically excited states . . . 19
- 6.7 Ionized and electron-attached states . . . 22
- 6.8 Relativistic calculations . . . 22
- 6.9 Reduced-scaling and local correlation calculations . . . 23
- 6.10 Optimization of basis sets . . . 26

7 Installation 29
- 7.1 Installation of pre-compiled binaries . . . 29
- 7.2 Installation from source code . . . 29
- 7.3 Installation under Windows . . . 33

8 Testing Mrcc 34

9 Running Mrcc 34
- 9.1 Running Mrcc in serial mode . . . 35
- 9.2 Running Mrcc in parallel using OpenMP . . . 35
- 9.3 Running Mrcc in parallel using MPI . . . 36
- 9.4 Troubleshooting MPI . . . 37

10 The programs of the suite 38

11 Input files 39

**12 Keywords** **40**

**13 Symmetry** **167**

**14 Interface to molecular visualization software** **168**
- 14.1 Molden . . . 168
- 14.2 xyz-file . . . 168

**15 Acknowledgments** **169**

**References** **170**

**List of keywords** **194**

# 1 Introduction

Mrcc is a suite of *ab initio* and density functional quantum chemistry programs for high-accuracy electronic structure calculations developed and maintained by the quantum chemistry research group at the Department of Physical Chemistry and Materials Science, TU Budapest [1, 2]. Its special feature, the use of automated programming tools enabled the development of tensor manipulation routines which are independent of the number of indices of the corresponding tensors, thus significantly simplifying the general implementation of quantum chemical methods. Applying the automated tools of the program several quantum chemistry models and techniques of high complexity have been implemented so far including arbitrary single-reference coupled-cluster (CC) and configuration interaction (CI) methods, multi-reference CC approaches, CC and CI energy derivatives and response functions, arbitrary perturbative CC approaches. Many features of the package are also available with relativistic Hamiltonians allowing for accurate calculations on heavy element systems. The developed cost-reduction techniques and local correlation approaches also enable high-precision calculations for medium-sized and large molecules.

# 2 How to read this manual

In the following, words set in `typewriter` font denote file names, shell commands, environment variables, and input records of the input file. These must be typed as shown. Variables, that is, numbers, options, etc. which must be specified by the user will be given as <*variable*>. These must be replaced by the corresponding values of the variables. Optional items are denoted by brackets, e.g., as [<*variable*>].

# 3 Authors

The main authors of the Mrcc code and their major contributions are as follows.

Mihály Kállay: general design; driver program (`dmrcc`); input analyzer (`minp`); automated, string-based many-body code (`goldstone`, `xmrcc`, `mrcc`); domain construction for local correlation calculations (`mulli`); particular integral evaluation algorithms (`integ`); direct and density-fitting (DF) Hartree–Fock (HF) algorithms, DFT algorithms (`scf`); density-fitting MP2 and RPA algorithms (`drpa`); explicitly correlated models (`mp2f12`)

Péter R. Nagy: local correlation methods and multi-level approaches (`drpa`); closed-shell coupled-cluster singles and doubles with perturbative triples [CCSD(T)] code (`ccsd`); parallelization of CCSD(T)

Dávid Mester: excited-state approaches (`cis`), range-separated double hybrid functionals (`dft`), density-based basis-set corrections (`dft`), bond functions (`integ`)

László Gyevi-Nagy: closed-shell CCSD(T) code (`ccsd`); parallelization of CCSD(T)

József Csóka: quadratic and multi-configurational self-consistent field algorithms, reduced-cost SCF algorithms (`scf`); analytic gradients for local density fitting SCF and Huzinaga-equation-based embedding (`prop`)

P. Bernát Szabó: open-shell CCSD(T) code (`uccsd`), open-shell local correlation approaches (`drpa`)

Zoltán Rolik: integral transformation and orbital optimization code (`ovirt`); domain construction for local correlation calculations (`mulli`)

Gyula Samu: density-fitting integrals (`integ`)

Bence Hégely: QM/MM and embedding approaches (`qmmod`)

Bence Ladóczki: four-center atomic orbital integral code (`integ`), parallelization of explicitly correlated MP2 and CCSD(T) theories (`mp2f12`)

Klára Petrov: orbital relaxation contribution to gradients for density-fitting methods, density-fitting MP2 gradient, natural auxiliary function (NAF) MP2 gradient (`prop`)

József Csontos: installation script (`build.mrcc`), geometry optimization, basis set optimization, the Mrcc homepage

Ádám Ganyecz: solvation models

István Ladjánszki: Hartree–Fock self-consistent field code (`scf`)

Lóránt Szegedy: closed-shell CCSD(T) code (`ccsd`)

Máté Farkas: orbital relaxation contribution to gradients for coupled-cluster methods (`prop`)

Pál D. Mezei: double hybrid and van der Waals density functionals (`dft`)

Réka A. Horváth: explicitly correlated models (`mp2f12`)

Balázs D. Lőrincz: non-atom-centered basis sets (`integ`)

In addition, Igor Gerasimov, Bence Kornis, Levente Dojcsák, Huliyar S. Nataraj, and Sanghamitra Das have also contributed to the development of the Mrcc code.

# 4 Citation

If results obtained with the Mrcc code are published, the publication must acknowledge the following two references.

"D. Mester, P. R. Nagy, J. Csóka, L. Gyevi-Nagy, P. B. Szabó, R. A. Horváth, K. Petrov, B. Hégely, B. Ladóczki, G. Samu, B. D. Lőrincz, and M. Kállay: Overview of Developments in the MRCC Program System, *J. Phys. Chem. A* **129**, 2086 (2025)."

"Mrcc, a quantum chemical program suite written by M. Kállay, P. R. Nagy, D. Mester, L. Gyevi-Nagy, J. Csóka, P. B. Szabó, Z. Rolik, G. Samu, B. Hégely, B. Ladóczki, K. Petrov, J. Csontos, Á. Ganyecz, I. Ladjánszki, L. Szegedy, M. Farkas, P. D. Mezei, R. A. Horváth, and B. D. Lőrincz. See `www.mrcc.hu`."

In addition, credit must be given to the corresponding papers which describe the implementation and the underlying methodological developments. The corresponding references are given in Sect. 6 of the manual.

If MRCC is used combined with other program systems or libraries, the users are also requested to include appropriate citations to those packages as required by their authors.

# 5 Interfaces

MRCC can be used as a standalone code, but interfaces have been developed to the CFOUR, COLUMBUS, DIRAC, MOLPRO, ORCA, and PSI quantum chemistry packages, the xTB and MOPAC semi-empirical programs as well as the AMBER molecular dynamics (MD) code. MRCC, in standalone mode, can currently be used for single-point energy calculations, evaluation of first-order properties, geometry optimizations, and harmonic vibrational frequency calculations with the standard nonrelativistic Hamiltonian and effective core potentials. The interfaces enable the calculation of further molecular properties as well as several other features, such as the use of relativistic Hamiltonians and MCSCF orbitals or quantum mechanics/molecular mechanics (QM/MM) simulations. If MRCC is used together with the aforementioned quantum chemistry packages, the integral, property integral, HF, MCSCF, and CPHF calculations, the integral and density-matrix transformations, etc. are performed by these program systems. Transformed MO (property) integrals are passed over to MRCC, which carries out the correlation calculation and returns unrelaxed MO density matrices if necessary.

In the following we describe the use of the CFOUR, COLUMBUS, DIRAC, MOLPRO, xTB, MOPAC, and AMBER interfaces and the features that they enable. For a complete list of available features, see Sect. 6. See also the description of keyword `iface`. For the ORCA and PSI interfaces see the manual of these packages.

Additionally, interfaces are available to driver programs running MRCC as one of their quantum chemistry engines [74–77]. These drivers, including CUBY [74, 78], SCINE [75, 79], APOST-3D [76, 80], and ASH [77], implement various composite, QM/MM, structure, optimization, dynamics, reaction network exploration, inverse molecule design, *a posteriori* wave function analysis (oxidation states, local spin, energy decomposition) etc. capabilities, detailed in their respective publications and user manuals.

## 5.1 CFOUR

Most of the implemented features are available via the CFOUR interface using RHF, ROHF, and UHF orbitals: single-point energy calculations, geometry optimizations, first-, second-, and third-order property calculations, electronic excitation energies, excited-state and transition properties, diagonal Born–Oppenheimer correction (DBOC) calculations. Most of the properties implemented in CFOUR are also available with MRCC. The interface also enables the use of several relativistic Hamiltonians.

The CFOUR interface is very user-friendly. You only have to prepare the input file `ZMAT` for CFOUR with the keyword `CC_PROG=MRCC`, and run CFOUR. The MRCC input file is then written automatically and MRCC is called directly by CFOUR, and you do not need to write any input file for MRCC. Most of the features of MRCC can be controlled by the corresponding CFOUR keyword, see CFOUR's homepage at `www.cfour.de`. If you use the CFOUR interface, you can safely ignore the rest of this manual.

You also have the option to turn off the automatic construction of the MRCC input file by giving `INPUT_MRCC=OFF` in the CFOUR input file `ZMAT`. However, it is only recommended for expert users.

## 5.2 COLUMBUS

Single-point energies, equilibrium geometries, ground- and excited-state first-order properties, and transition moments can be computed with RHF, ROHF, and MCSCF reference states using the COLUMBUS interface. Evaluation of harmonic vibrational frequencies is also possible via numerically differentiated analytical gradients.

Running MRCC with COLUMBUS requires three additional programs, `colto55`, `coldens`, and `runc_mrcc`, which are available for COLUMBUS licensees from the authors of MRCC upon request.

To use this interface for single-point energy calculations first prepare input files for COLUMBUS using the `colinp` script. It is important to set a calculation in the input file which requires a complete integral transformation (e.g., CISD and not just MCSCF). Execute COLUMBUS. If you do not need the results of the COLUMBUS calculations, you can stop them after completing the integral transformation. Run the `colto55` program in the `WORK` directory created by COLUMBUS. This will convert the COLUMBUS integral files to the MRCC format. Prepare input file `MINP` for MRCC as described in Sect. 11. Run `dmrcc` as described in Sect. 9. It is recommended to execute first some inexpensive calculation (e.g., CISD) with MRCC and compare the HF and CISD energies in order to test your input files.

For property calculations create the COLUMBUS and the MRCC input files. In the COLUMBUS input set the corresponding MRCI property calculation. Copy the MRCC input file `MINP` to the `WORK` directory of COLUMBUS. If the directory does not exist, create it. Then execute the `runc_mrcc` script.

## 5.3 DIRAC

The interface to the DIRAC code enables four-component relativistic calculations with the full Dirac–Coulomb Hamiltonian and several approximate variants thereof. Single-point energy calculations are possible with all CC and CI methods implemented in MRCC using Kramers-paired Dirac–Fock orbitals. First-order property (unrelaxed) calculations are available with iterative CC and CI methods. See Refs. 18 and 20 for more details.

If you use DIRAC, you should first prepare input files for DIRAC (see http://diracprogram.org/). It is important to run a full integral transformation with DIRAC (see the description of the `MOLTRA` keyword in DIRAC's manual), and to use Abelian symmetry (that is, the $D_{2h}$ group and its subgroups). Execute the `pam` script saving the `MRCONEE` and `MDCINT` files, e.g., running it as

```
pam --get="MRCONEE MDCINT" --inp=Y.inp --mol=X.mol
```

where `X.mol` and `Y.inp` should be replaced by your input files as appropriate. Then run the `dirac_mointegral_export` interface program, which generates the files needed by MRCC. It also creates a sample input file `MINP` for MRCC, which contains the input for a closed-shell CCSD calculation. If you intend to run another type of calculation, please edit the file as described in Sect. 11. Please also note that you may need to modify the occupation vector under the `refdet` keyword (see the description of the keyword on page 143), and you should set `hamilton=x2c` if exact 2-component Hamiltonians are used. Then run `dmrcc` as described in Sect. 9.

For relativistic property calculations define the corresponding operator in the DIRAC input file (see the description of the `PROPERTIES` and `MOLTRA` keywords in DIRAC's manual). Then execute the `pam` script as

```
pam --get="MRCONEE MDCINT MDPROP" --inp=Y.inp --mol=X.mol
```

and edit the `MINP` file, in particular, set the `dens` keyword (page 65). The CC property code currently does not work with double-group symmetry, and you need to turn off symmetry for CC property calculations, that is, set `symm=off` in the `MINP` file. Finally run `dmrcc`.

## 5.4 MOLPRO

With MOLPRO single-point energy calculations are available using RHF, UHF, ROHF, and MCSCF orbitals. The interface also enables the use of Douglas–Kroll–Hess Hamiltonians as well as effective core potentials.

The MOLPRO interface is very user-friendly. You only have to prepare the input file for MOLPRO with a line starting with the `mrcc` label followed by the corresponding keywords, and run MOLPRO. The MRCC input file is then written automatically and MRCC is called directly by MOLPRO, and you do not need to write any input file for MRCC. Most of the features of MRCC can be controlled by the corresponding MOLPRO keywords. If you use MOLPRO, you also have the option to install MRCC with the makefile of MOLPRO.

For a detailed description of the interface point your browser to the MOLPRO User's Manual at `www.molpro.net` and then click "34 The MRCC program of M. Kallay (MRCC)".

If you use the MOLPRO interface, you can safely ignore the rest of this manual.

## 5.5 AMBER

The AMBER/MRCC interface enables QM/MM and other multiscale calculations. The interface of the AMBER MD code and MRCC is based on the work of Götz *et al.* [81], which facilitates the integration of QM codes into AMBER as external modules. The detailed description of the AMBER/MRCC interface is documented in Ref. 29. Currently for the separation of the QM and MM subsystems only the link atom approach is supported.

With the AMBER/MRCC program the projection-based embedding techniques, namely the projector-augmented operator of Manby and Miller [82] and our Huzinaga-operator [33] approaches, can also be employed for the QM region. The multilevel approach based on our local correlation methods is also supported [33]. The latter approaches enables the embedding of wave function theory (WFT) or density functional theory (DFT) methods into lower-level WFT or DFT methods and also the combinations thereof (DFT-in-DFT, WFT-in-DFT, WFT-in-WFT, and WFT-in-WFT-in-DFT). Consequently, with the AMBER interface you can also define three (QM/QM/MM) or four (QM/QM/QM/MM) layers for the multilevel calculations. At the present stage of development, the multilevel methods with three or four layers are only available for single point energy calculations. The traditional multilevel approach, "Our own n-layered integrated molecular orbital and molecular mechanics" (ONIOM) of Morokuma *et al.* [83], is also supported with two QM layers by our in-house interface for single point, geometry optimization, and (Born–Oppenheimer) molecular dynamics calculations. The unofficial Mrcc module, which supports ONIOM calculations, is available on user-request.

Using the RISM modules of Amber, the embedded cluster reference interaction site model (EC-RISM) method of Kast *et al.* [84, 85] was implemented as a solvation model [56]. These calculations are controlled by Mrcc, and Amber is used for the 3D-RISM part of the calculations. Based on the integral equation theory, 3D-RISM can provide solvation thermodynamics and distribution functions like molecular dynamics (MD), but in a few minutes, while its accuracy is also satisfying. EC-RISM is a QM/MM-like combination of 3D-RISM and QM to describe systems in solutions. See keyword `rism` for further details.

To use the Amber/Mrcc interface you need a properly installed version of Amber (version 2017 or later), see the Amber homepage at ambermd.org. The detailed description of the usage of the interface is well documented in the Amber manual, see section “11.2.6.6. AMBER/MRCC” in the manual of Amber21.

## 5.6 xTB

The ONIOM method [83] can be also applied with the xTB program [86], which implements the density functional tight binding (DFTB) methods of Grimme and co-workers [87, 88]. These theoretical models are parameterized for geometries, vibrational frequencies, and non-covalent interactions (GFN), and the necessary parameters are available for most of the periodic system. In the ONIOM implementation, single-point and gradient calculations are supported with the mechanical embedding approach, single-point calculations with various point charge models can be utilized for electronic embedding, including the Mulliken charges of the GFN-xTB models, and single-point and gradient calculations can be carried out with electronic embedding if the point charges are user-defined and independent of the geometry. Note that the use of the force-field version of the GFN family (GFN-FF) [89] and the implicit solvent models of xTB [90] is also supported in the ONIOM framework. Finally, the GFN1-xTB and GFN2-xTB can also be used for initial guess calculations as it is described in Ref. 54.

## 5.7 Mopac

The Mrcc program is also interfaced to the Mopac program of Stewart [91], which allows various semi-empirical molecular orbital methods, for instance, the PM6 and PM7 versions of the neglect of diatomic differential overlap (NDDO) approach, to be used with the mechanical embedding version of the ONIOM implementation. Single-point and gradient calculations are supported along with the COSMO implicit solvent model of the Mopac program.

# 6 Features

In this section the available features of the Mrcc code are summarized. We also specify what type of reference states (orbitals) can be used, and if a particular feature requires one of the interfaces or is available with Mrcc in standalone mode. We also give the corresponding references which describe the underlying methodological developments.

## 6.1 Single-point energy calculations

Single-point energy calculations are possible with the following methods. See also Tables 1 and 2 for the available ground- and excited-state models, respectively, as well as Table 3 for the reduced-cost approaches. For a detailed list of the available density functional approximations, see Table 4.

### Available methods

1. conventional and density-fitting (resolution-of-identity) Hartree–Fock SCF (Ref. 24): restricted HF (RHF), unrestricted HF (UHF), and restricted open-shell HF (ROHF)

2. conventional and density-fitting (resolution-of-identity) multi-configurational SCF (MCSCF)

3. conventional and density-fitting (resolution-of-identity) Kohn–Sham (KS) density functional theory (DFT) (Ref. 26): restricted KS (RKS), unrestricted KS (UKS), restricted open-shell KS (ROKS); local density approximation (LDA), generalized gradient approximation (GGA), meta-GGA (depending on kinetic-energy density and/or the Laplacian of the density), hybrid, range-separated hybrid, double hybrid (DH), third-order double hybrid, and range-separated double hybrid (RSDH) functionals (for the available functionals see the description of keyword `dft`); dispersion corrections

4. time-dependent HF (TD-HF), time-dependent DFT (TD-DFT), TD-DFT in the Tamm–Dancoff approximation (TDA); currently only for closed-shell molecules with RHF/RKS reference using density-fitting, for LDA and GGA functionals and their hybrids and double hybrids even with range separation (Refs. 40, 41, 49, 52, and 57)

5. density-fitting (resolution-of-identity) MP2, spin-component scaled MP2 (SCS-MP2), and scaled opposite-spin MP2 (SOS-MP2); currently only for RHF and UHF references (Ref. 26)

6. density-fitting (resolution-of-identity) random-phase approximation (RPA, also known as ring-CCD, rCCD), direct RPA (dRPA, also known as direct ring-CCD, drCCD), second-order screened exchange (SOSEX), renormalized second-order perturbation theory (rPT2), and approximate RPA with exchange (RPAX2); currently the dRPA, SOSEX, rPT2, and RPAX2 methods are available for RHF/RKS and UHF/UKS references, while the RPA method is only implemented for RHF/RKS (Refs. 26 and 30)

7. scaled-equation and down-scaled dRPA and SOSEX methods (sedRPA, seSOSEX, ds-dRPA, dsSOSEX (Ref. 44); range-separated dRPA (Ref. 45).

8. density-fitting (resolution-of-identity) second-order coupled-cluster singles and doubles (CC2), configuration interaction singles with perturbative correction for double excitations [CIS(D)], iterative doubles correction to configuration interaction singles [CIS($D_\infty$)], and second-order algebraic diagrammatic construction [ADC(2)] approaches; spin-component scaled CC2, CIS(D), CIS($D_\infty$), and ADC(2) [SCS-CC2, SCS-CIS(D), SCS-CIS($D_\infty$), and SCS-ADC(2)]; scaled opposite-spin CC2, CIS(D), CIS($D_\infty$), and ADC(2) [SOS-CC2, SOS-CIS(D), SOS-CIS($D_\infty$), and SOS-ADC(2)] (Refs. 35, 37, 41, and 43)

9. arbitrary single-reference coupled-cluster methods (Ref. 3): CCSD, CCSDT, CCSDTQ, CCSDTQP, . . . , CC($n$)

10. arbitrary single-reference configuration-interaction methods (Ref. 3): CIS, CISD, CISDT, CISDTQ, CISDTQP, . . . , CI($n$), . . . , full CI

11. arbitrary perturbative coupled-cluster models (Refs. 9, 8, and 15):
    - CCSD[T], CCSDT[Q], CCSDTQ[P], . . . , CC($n-1$)[$n$]
    - CCSDT[Q]/A, CCSDTQ[P]/A, . . . , CC($n-1$)[$n$]/A
    - CCSDT[Q]/B, CCSDTQ[P]/B, . . . , CC($n-1$)[$n$]/B
    - CCSD(T), CCSDT(Q), CCSDTQ(P), . . . , CC($n-1$)($n$)
    - CCSDT(Q)/A, CCSDTQ(P)/A, . . . , CC($n-1$)($n$)/A
    - CCSDT(Q)/B, CCSDTQ(P)/B, . . . , CC($n-1$)($n$)/B
    - CCSD(T)$_\Lambda$, CCSDT(Q)$_\Lambda$, CCSDTQ(P)$_\Lambda$, . . . , CC($n-1$)($n$)$_\Lambda$
    - CCSDT-1a, CCSDTQ-1a, CCSDTQP-1a, . . . , CC($n$)-1a
    - CCSDT-1b, CCSDTQ-1b, CCSDTQP-1b, . . . , CC($n$)-1b
    - CC2, CC3, CC4, CC5, . . . , CC$n$
    - CCSDT-3, CCSDTQ-3, CCSDTQP-3, . . . , CC($n$)-3

12. explicitly correlated MP2 (MP2-F12) using the 2B ansatz, the F + K commutator approximation, fixed amplitudes, spin-flipped geminals in open-shell calculations (Ref. 53)

13. explicitly correlated CCSD and CCSD(T) utilizing the CCSD(F12*)(T+) model and the above MP2-F12 approximation (Ref. 53)

14. density-based basis-set correction (DBBSC) for wave function methods (Refs. 67, 71, 72, 92)

15. multi-reference CI approaches, Davidson correction for MRCI (Ref. 4)

16. multi-reference CC approaches using a state-selective ansatz (Ref. 4)

17. arbitrary single-reference linear-response (equation-of-motion, EOM) CC methods (Ref. 7): LR-CCSD (EOM-CCSD), LR-CCSDT (EOM-CCSDT), LR-CCSDTQ (EOM-CCSDTQ), LR-CCSDTQP (EOM-CCSDTQP), . . . , LR-CC($n$) [EOM-CC($n$)]

18. linear-response (equation-of-motion) MRCC schemes (Ref. 7)

19. DFT/WFT embedding (Ref. 33): DFT-in-DFT, WFT-in-DFT, WFT-in-WFT, and WFT-in-WFT-in-DFT embedding (where WFT stands for wave function theory); currently LDA, GGA, meta-GGA, or hybrid functionals can be used as the DFT method; any WFT method implemented in Mrcc can by used in WFT-in-DFT type embedding calculations; WFT-in-WFT multi-level methods (via keyword `corembed`) is only available for local correlation methods.

20. ONIOM approach with arbitrary number of layers [83]. Currently, only the integrated MO+MO (IMOMO) [93] part of ONIOM is available in the standalone version of Mrcc with mechanical embedding, electrostatic embedding, and automated link atom handling. Note that all types of the aforementioned DFT/WFT embedding techniques are also supported as top-level ONIOM calculations.

The CI and CC approaches listed above are available with RHF, UHF, standard/semi-canonical ROHF, MCSCF orbitals. Density-fitting with CI or high-order CC is currently enabled only for RHF references. For the perturbative CC approaches with ROHF reference determinant, for theoretical reasons, semi-canonical orbitals are used (see Ref. 15).

### Features available via interfaces

1. The CI and CC approaches listed above are also available with the following interfaces and references.

   RHF: Cfour, Columbus, and Molpro

   ROHF, standard orbitals: Cfour, Columbus, and Molpro

   ROHF, semi-canonical orbitals: Cfour

   UHF: Cfour, and Molpro

   MCSCF: Columbus and Molpro

2. QM/MM calculations can be performed by the Amber interface using any method implemented in Mrcc for the QM region.

3. Semi-empirical quantum mechanical calculations are available through the Mopac and xTB interfaces in the ONIOM framework with mechanical embedding using automated link atom handling. The electrostatic embedding version of the ONIOM scheme is only available with the xTB program.

### Notes

1. Single-point calculations are also possible with several types of relativistic Hamiltonians and reference functions, see Sect. 6.8 for more details.

2. Reduced-scaling approaches for the above CC and CI methods are available (Ref. 21). See Sect. 6.9 for details.

3. Local CC approaches for arbitrary single-reference and perturbative coupled-cluster models, local MP2 approaches, and local dRPA are available (Refs. 22, 24, 30, 31, 36, and 38). See Sect. 6.9 for details.

4. CC$n$ calculations with ROHF orbitals are not possible for theoretical reasons, see Ref. 15 for explanation.

5. Single-point CI and CC calculations are, in principle, possible with RKS, UKS, ROKS orbitals.

Table 1: Methods implemented in MRCC for ground states.

| Method | | References | | Grad.[c] |
|---|---|---|---|---|
| Abbreviation | Full name/description | Meth.[a] | Imp.[b] | |
| RHF | restricted Hartree–Fock | 98 | 24 | + |
| UHF | unrestricted Hartree–Fock | 98 | 24 | + |
| ROHF | restricted open-shell Hartree–Fock | 99 | 24 | – |
| MCSCF | multiconfigurational self-consistent field | 100 | 101 | – |
| RKS | restricted Kohn–Sham | 102 | 30 | + |
| UKS | unrestricted Kohn–Sham | 102 | 30 | + |
| ROKS | restricted open-shell Kohn–Sham | 102, 103 | 30 | – |
| RS-DFT | range-separated DFT | 104 | 49 | – |
| DH-DFT | double hybrid DFT | 105 | 28, 32 | + |
| RSDH-DFT | range-separated double hybrid DFT | 106, 107 | 49 | – |
| MP2 | second-order Møller–Plesset perturbation theory | 108 | 26 | + |
| SCS-MP2 | spin-component scaled MP2 | 109 | 26 | – |
| SOS-MP2 | scaled opposite-spin MP2 | 110 | 26 | – |
| MP2-F12 | explicitly correlated MP2 | 111, 112 | 53 | – |
| dRPA | direct random-phase approximation | 113 | 26, 30 | – |
| SOSEX | second-order screened exchange | 114 | 30 | – |
| rPT2 | renormalized second-order perturbation theory | 115 | 44 | – |
| RPA | random-phase approximation | 116 | 30 | – |
| RPAX2 | approximate RPA with exchange | 117 | 28 | – |
| CC2 | second-order CC singles and doubles | 118 | 35 | – |
| SCS-CC2 | spin-component scaled CC2 | 119 | 35, 41 | – |
| SOS-CC2 | scaled opposite-spin CC2 | 120 | 35, 41 | – |
| MP3 | third-order MP perturbation theory | 121 | 46 | – |
| CCSD | CC singles and doubles | 122, 123 | 24, 46 | + |
| CCSDT | CC singles, doubles, and triples | 124 | 3 | + |
| CCSDTQ | CC singles, doubles, triples, and quadruples | 125 | 3 | + |
| CCSDTQP | CC singles, doubles, triples, quadruples, and pentuples | 3 | 3 | + |
| CC($n$) | CC method including up to $n$-tuple excitations | 3 | 3 | + |
| CCSD(T) | CC singles and doubles with perturbative triples | 126 | 24, 46 | – |
| CCSDT(Q) | CCSDT with perturbative quadruples | 8, 15 | 9, 15 | – |
| CCSDTQ(P) | CCSDTQ with perturbative pentuples | 9, 15 | 9, 15 | – |
| CC($n-1$)($n$) | CC($n-1$) method with perturbative $n$-tuple excitations | 9, 15 | 9 | – |
| CCSD[T] | CCSD with perturbative triples, “bracket T” | 127 | 9, 15 | – |
| CCSDT[Q] | CCSDT with perturbative quadruples, “bracket Q” | 128 | 9, 15 | – |
| CC($n-1$)[$n$] | generalization of CCSD[T] to $n$-tuple excitations | 9, 15 | 9, 15 | – |

| | | | | |
|---|---|---|---|---|
| CCSD(T)$_\Lambda$ | CCSD with perturbative asymmetric (T) correction | 129, 130 | 9 | – |
| CCSDT(Q)$_\Lambda$ | CCSDT with perturbative asymmetric (Q) correction | 9 | 9 | – |
| CC$(n-1)(n)_\Lambda$ | generalization of CCSD(T)$_\Lambda$ to $n$-tuple excitations | 9 | 9 | – |
| CCSDT-1a | CCSD with iterative approximate triples, ansatz 1a | 127, 131 | 9 | – |
| CCSDTQ-1a | CCSDT with iterative approximate quadruples, ansatz 1a | 128 | 9 | – |
| CC($n$)-1a | generalization of CCSDT-1a to $n$-tuple excitations | 9 | 9 | – |
| CCSDT-1b | CCSD with iterative approximate triples, ansatz 1b | 127 | 9 | – |
| CCSDTQ-1b | CCSDT with iterative approximate quadruples, ansatz 1b | 9 | 9 | – |
| CC($n$)-1b | generalization of CCSDT-1b to $n$-tuple excitations | 9 | 9 | – |
| CC3 | CCSD with iterative approximate triples, CC3 ansatz | 132 | 9 | – |
| CC4 | CCSDT with iterative approximate quadruples, CC3 ansatz | 9 | 9 | – |
| CC$n$ | generalization of CC3 to $n$-tuple excitations | 9 | 9 | – |
| CCSDT-3 | CCSD with iterative approximate triples, ansatz 3 | 127 | 9 | – |
| CCSDTQ-3 | CCSDT with iterative approximate quadruples, ansatz 3 | 9 | 9 | – |
| CC($n$)-3 | generalization of CCSDT-3 to $n$-tuple excitations | 9 | 9 | – |
| CCSD(F12*) | explicitly correlated CCSD using the CCSD(F12*) ansatz | 133 | 53 | – |
| CCSD(F12*)(T) | explicitly correlated CCSD using the CCSD(F12*) ansatz and the (T) correction | 133 | 53 | – |
| CCSD(F12*)(T*) | explicitly correlated CCSD using the CCSD(F12*) ansatz and the (T*) correction | 133, 134 | 53 | – |
| CCSD(F12*)(T+) | explicitly correlated CCSD using the CCSD(F12*) ansatz and the (T+) correction | 53, 133 | 53 | – |
| DBBSC | density-based basis-set correction | 135, 136 | 67, 72 | – |
| CISD | CI singles and doubles | 98 | 3 | + |
| CISDT | CI singles, doubles, and triples | 98 | 3 | + |
| CISDTQ | CI singles, doubles, triples, and quadruples | 98 | 3 | + |
| CI($n$) | CI method including up to $n$-tuple excitations | 98 | 3 | + |
| FCI | full CI | 98 | 3 | + |

| | | | | |
|---|---|---|---|---|
| MRCCSD | multireference CC singles and doubles | 137, 138 | 4 | + |
| MRCC($n$) | multireference CC including up to $n$-tuple excitations | 4, 138 | 4 | + |
| MRCISD | multireference CI singles and doubles | 139 | 4 | + |
| MRCISD+Q | MRCISD with Davidson correction | 139, 140 | 4 | − |
| MRCI($n$) | multireference CI including up to $n$-tuple excitations | 4, 139 | 4 | + |

[a]References describing the methodological developments [b]References describing the implementation in Mrcc [c]Availability of analytic gradients

6. Solvation effects can be modeled by the polarizable continuum model (PCM) [94] via an interface to the PCMSolver library [95–97]. The PCM treatment is self-consistent for HF and KS SCF calculations, while, in the post-SCF steps, the potential of the solvent optimized at the SCF level is frozen. See the descriptions of keywords `pcm*`. An improved treatment of solvent effects is possible using the embedded cluster reference interaction site model (EC-RISM) [56], which is available via the Amber interface, see Sect. 5.5 and keyword `rism` for further details. Note that the implicit solvation models of the Mopac and xTB programs are also available in the ONIOM framework.

## 6.2 Geometry optimizations and first-order properties

### Available methods

Geometry optimizations and first-order property calculations can be performed using analytic gradient techniques with the following methods. See also Tables 1 and 2 for the availability of analytic gradients for ground- and excited-state models, respectively.

1. conventional and DF (RI) HF-SCF (Ref. 24): RHF and UHF

2. conventional and DF (RI) DFT (Ref. 26): RKS and UKS with LDA, GGA, meta-GGA (depending only on kinetic-energy density), and hybrid functionals as well as dispersion corrections

3. local DF RHF and hybrid RKS methods (Ref. 64)

4. double hybrid density functional methods (Ref. 26), such as B2PLYP, B2PLYP-D3, B2GPPLYP, etc. (current limitations: only MP2 correlation, closed shell RKS, no spin-component scaling, no meta-GGA functionals, only DH functionals for which the DFT contribution to the energy is stationary with respect to the variation of the MO coefficients)

5. DF-MP2 (RI-MP2), currently only for RHF references (Ref. 26)

6. NAF-MP2, currently only for RHF references (Ref. 70)

7. arbitrary single-reference coupled-cluster methods (Refs. 3 and 5): CCSD, CCSDT, CCSDTQ, CCSDTQP, . . . , CC($n$)

8. arbitrary single-reference configuration-interaction methods (Refs. 3 and 5): CIS, CISD, CISDT, CISDTQ, CISDTQP, . . . , CI($n$), . . . , full CI
9. multi-reference CI approaches (Refs. 4 and 5)
10. multi-reference CC approaches using a state-selective ansatz (Refs. 4 and 5)
11. CIS and TD-HF methods using RHF references
12. arbitrary single-reference linear-response (equation-of-motion, EOM) CC methods (Refs. 5 and 7): LR-CCSD (EOM-CCSD), LR-CCSDT (EOM-CCSDT), LR-CCSDTQ (EOM-CCSDTQ), LR-CCSDTQP (EOM-CCSDTQP), . . . , LR-CC($n$) [EOM-CC($n$)]
13. linear-response (equation-of-motion) MRCC schemes (Refs. 5 and 7)
14. The ONIOM approach with arbitrary number of layers [83]. Currently, only the IMOMO [93] part of ONIOM is available in the standalone version of Mrcc with mechanical embedding and automated link atom handling. ONIOM gradients with electronic embedding are also available with perturbation independent point charges.
15. Huzinaga-equation-based embedding, currently only for closed-shell systems (Ref. 69)

Currently only unrestricted geometry optimizations are possible, and electric dipole, quadrupole, and octapole moments as well as the electric field at the atomic centers can be evaluated. In addition, Mulliken, Löwdin, intrinsic atomic orbital (IAO), CHELPG, and Merz–Singh–Kollman atomic charges, and Mayer bond orders can be computed using the SCF wave functions. Analytic gradients for the CI and CC methods listed above are available with RHF, UHF, and standard ROHF orbitals without density fitting.

The following keywords are available to control the optimization process

`optalg` – to select an algorithm for the optimization

`optmaxit` – maximum number of iterations allowed

`optetol` – convergence criterion for energy change

`optgtol` – convergence criterion for the gradient change

`optstol` – convergence criterion for the step-size

The optimization will be terminated and regarded as successful when the maximum gradient component becomes less than `optgtol` and either an energy change from the previous step is less than `optetol` or the maximum displacement from the previous step is less than `optstol`. For their detailed description see Sect. 12.

### Features available via interfaces

1. The implemented analytic gradients for the CI and CC approaches listed above can also be utilized via the Cfour and Columbus interfaces with the following references.

   RHF: Cfour and Columbus

   ROHF, standard orbitals: Cfour and Columbus

   UHF: Cfour

   MCSCF: Columbus

   In addition to geometries, most of the first-order properties (dipole moments, quadrupole moments, electric field gradients, relativistic contributions, etc.) implemented in Cfour and Columbus can be calculated with Mrcc.

2. QM/MM geometry optimizations and MD calculations can be performed by the Amber interface using any method implemented in Mrcc for which analytic gradients are available.

3. Analytic gradients are also supported for the semi-empirical quantum mechanics methods of the Mopac and xTB programs with the mechanical embedding of the ONIOM framework. The electronic embedding with perturbation independent point charges is also supported for the DFTB methods of the xTB program.

### Notes

1. Geometry optimizations and first-order property calculations can also be performed via numerical differentiation for all methods available in Mrcc using the Cfour interface.

2. Analytic gradients are are also available with several types of relativistic Hamiltonians and reference functions, see Sect. 6.8 for more details.

3. Analytic gradients are available with the PCM via an interface to the PCMSolver library [96, 97] at the HF and KS SCF levels. See the descriptions of keywords `pcm*`.

4. The implicit solvation models of the xTB and Mopac programs and their analytic gradient are available in the ONIOM implementation. See the descriptions of keywords `pcm*` and `oniom_pcm`.

## 6.3 Harmonic frequencies and second-order properties

### Available methods

Harmonic vibrational frequencies, infrared (IR) intensities, and ideal gas thermodynamic properties can be evaluated using numerically differentiated analytic gradients for all the methods listed in Sect. 6.2.

### Features available via interfaces

CC and CI harmonic frequency and second-order property calculations for RHF and UHF references can also be performed using analytic second derivatives (linear response functions) with the aid of the Cfour interface. Analytic Hessians (LR functions) are available for the following approaches.

1. arbitrary single-reference coupled-cluster methods (Refs. 3, 5, 6, 11, 12, and 16): CCSD, CCSDT, CCSDTQ, CCSDTQP, . . . , CC($n$)
2. arbitrary single-reference configuration-interaction methods (Refs. 3, 5, and 6): CIS, CISD, CISDT, CISDTQ, CISDTQP, . . . , CI($n$), . . . , full CI
3. multi-reference CI approaches (Refs. 4, 5, and 6)
4. multi-reference CC approaches using a state-selective ansatz (Refs. 4, 5, 6, and 11)

In addition to harmonic vibrational frequencies [6], the analytic Hessian code has been tested for NMR chemical shifts [6], static and frequency-dependent electric dipole polarizabilities [11], magnetizabilities and rotational $g$-tensors [12], electronic $g$-tensors [16], spin-spin coupling constants, and spin rotation constants. These properties are available via the Cfour interface.

### Notes

1. Using the Cfour interface, harmonic frequency calculations are also possible via numerical differentiation of energies for all implemented methods with RHF, ROHF, and UHF orbitals.
2. Using the Cfour or the Columbus interface, harmonic frequency calculations are also possible via numerical differentiation of analytic gradients for all implemented methods for which analytic gradients are available (see Sect. 6.2 for a list of these methods). With Cfour, the calculation of static polarizabilities is also possible using numerical differentiation.
3. NMR chemical shifts can be computed for closed-shell molecules using gauge-including atomic orbitals and RHF reference function.

## 6.4 Higher-order properties

### Features available via interfaces

Third-order property calculations can be performed using analytic third derivative techniques (quadratic response functions) invoking the Cfour interface for the following methods with RHF and UHF orbitals.

1. arbitrary single-reference coupled-cluster methods (Refs. 3, 5, 6, 13, and 14): CCSD, CCSDT, CCSDTQ, CCSDTQP, . . . , CC($n$)
2. multi-reference CC approaches using a state-selective ansatz (Refs. 4, 5, 6, 13, and 14)

### Notes

1. The analytic third derivative code has been tested for static and frequency-dependent electric-dipole first (general, second-harmonic-generation, optical-rectification) hyperpolarizabilities [13] and Raman intensities [14]. Please note that the orbital relaxation effects are not considered for the electric-field. These properties are available via the CFOUR interface.

2. Using the CFOUR interface, anharmonic force fields and the corresponding spectroscopic properties can be computed using numerical differentiation techniques together with analytic first and/or analytic second derivatives at all computational levels for which these derivatives are available (see Sect. 6.2 and 6.3 for a list of these methods).

## 6.5 Diagonal Born–Oppenheimer corrections

### Features available via interfaces

Diagonal Born–Oppenheimer correction (DBOC) calculations can be performed using analytic second derivative techniques via the CFOUR interface for the following methods with RHF and UHF references.

1. arbitrary single-reference coupled-cluster methods (Refs. 3, 5, 6, and 10): CCSD, CCSDT, CCSDTQ, CCSDTQP, . . . , CC($n$)

2. arbitrary single-reference configuration-interaction methods (Refs. 3, 5, 6, and 10): CIS, CISD, CISDT, CISDTQ, CISDTQP, . . . , CI($n$), . . . , full CI

3. multi-reference CI approaches (Refs. 4, 5, 6, and 10)

4. multi-reference CC approaches using a state-selective ansatz (Refs. 4, 5, 6, and 10)

## 6.6 Electronically excited states

### Available methods

Excitation energies, first-order excited-state properties, and ground to excited-state transition moments can be computed as well as excited-state geometry optimizations can be performed using linear response theory and analytic gradients with the following methods.

1. arbitrary single-reference linear-response CC methods (Refs. 3, 5, and 7): LR-CCSD, LR-CCSDT, LR-CCSDTQ, LR-CCSDTQP, . . . , LR-CC($n$)

2. linear-response MRCC schemes (Refs. 4, 5, and 7)

3. arbitrary single-reference configuration-interaction methods (Refs. 3, 5, and 7): CIS, CISD, CISDT, CISDTQ, CISDTQP, . . . , CI($n$), . . . , full CI

4. multi-reference CI approaches (Refs. 4, 5, and 7)

Excitation energy and property calculations for the aforementioned methods are available with RHF, UHF, and standard ROHF orbitals. Density-fitting is only possible for RHF-based single point calculations. So far electric and magnetic dipole transition moments, both in the length and the velocity gauge, as well as the corresponding oscillator and rotator strengths have been implemented. For the list of implemented first-order properties see Sect. 6.2.

Excitation energies can also be computed for closed-shell systems using the density-fitting approximation and RHF (RKS) orbitals with the following methods (Refs. 35, 37, 40, 41, 49, 52, and 57).

1. CIS, time-dependent HF (TD-HF), Tamm–Dancoff approximation (TDA), time-dependent DFT (TD-DFT)
2. configuration interaction singles with perturbative correction for double excitations [CIS(D)]
3. second-order coupled-cluster singles and doubles (CC2) method
4. iterative doubles correction to configuration interaction singles [CIS($D_\infty$)] method
5. second-order algebraic diagrammatic construction [ADC(2)] approach
6. spin-scaled versions of the latter approaches: SCS-CIS(D), SCS-CC2, SCS-CIS($D_\infty$), SCS-ADC(2), SOS-CIS(D), SOS-CC2, SOS-CIS($D_\infty$), and SOS-ADC(2)
7. CIS(D)- and ADC(2)-based double hybrid TD-DFT methods even with range separation

Analytic gradients are implemented for CIS and TD-HF. Ground to excited-state transition moments are available for TD-HF, TDA, TD-DFT, hybrid and double-hybrid TD-DFT even with range separation, CIS(D), SCS-CIS(D), SOS-CIS(D), ADC(2), SCS-ADC(2), and SOS-ADC(2). For the CIS, TD-HF, hybrid TDA and TD-DFT, CC2, CIS($D_\infty$), ADC(2), SCS-CC2, SCS-CIS($D_\infty$), SCS-ADC(2), SOS-CC2, SOS-CIS($D_\infty$), and SOS-ADC(2) methods efficient reduced-cost approaches are also implemented, see Sect. 6.9 for details.

Core excitation energies and transition moments can also be calculated employing the core-valence separation (CVS) approximation with all the above methods (Refs. 62 and 63).

### Features available via interfaces

Excitation energies, first-order excited-state properties, and ground to excited-state transition moments can also be calculated as well as excited-state geometry optimizations can also be carried out using the following interfaces and reference states.

RHF: Cfour, Columbus, and Molpro (only excitation energy)

ROHF, standard orbitals: Cfour, Columbus, and Molpro (only excitation energy)

UHF: Cfour and Molpro (only excitation energy)

MCSCF: Columbus and Molpro (only excitation energy)

Table 2: Methods implemented in Mrcc for excited electronic states.

| Method | | References | | Grad.[c] |
|---|---|---|---|---|
| Abbreviation | Full name/description | Meth.[a] | Imp.[b] | |
| CIS | configuration interaction singles | 141 | 40 | + |
| MCSCF | multiconfigurational self-consistent field | 100 | 101 | − |
| TDHF | time-dependent Hartree–Fock | 142 | 40 | + |
| TDDFT | time-dependent DFT | 143 | 40 | − |
| TDA-TDDFT | Tamm–Dancoff approximation TDDFT | 144 | 40 | − |
| RS-TDDFT | range-separated TDDFT | 145, 146 | 49, 57 | − |
| DH-TDDFT | double hybrid TDDFT | 147 | 41 | − |
| RSDH-TDDFT | range-separated double hybrid TDDFT | 49, 148 | 52, 57 | − |
| CIS(D) | CIS with perturbative doubles | 149 | 35 | − |
| SCS-CIS(D) | spin-component scaled CIS(D) | 150, 151 | 35, 41 | − |
| SOS-CIS(D) | scaled opposite-spin CIS(D) | 151 | 35, 41 | − |
| CIS($D_\infty$) | CIS with iterative approximate doubles | 152 | 35 | − |
| SCS-CIS($D_\infty$) | spin-component scaled CIS($D_\infty$) | 119 | 35, 41 | − |
| SOS-CIS($D_\infty$) | scaled opposite-spin CIS($D_\infty$) | 120 | 35, 41 | − |
| ADC(2) | second-order algebraic-diagrammatic construction | 153 | 37 | − |
| SCS-ADC(2) | spin-component scaled ADC(2) | 119 | 37, 41 | − |
| SOS-ADC(2) | scaled opposite-spin ADC(2) | 120 | 37, 41 | − |
| LR-CC2 | linear response CC2 | 118 | 35 | − |
| LR-SCS-CC2 | linear response SCS-CC2 | 119 | 35, 41 | − |
| LR-SOS-CC2 | linear response SOS-CC2 | 120 | 35, 41 | − |
| LR-CCSD | linear response CCSD | 123 | 3, 7 | + |
| LR-CCSDT | linear response CCSDT | 154 | 3, 7 | + |
| LR-CCSDTQ | linear response CCSDTQ | 125 | 3, 7 | + |
| LR-CCSDTQP | linear response CCSDTQP | 3 | 3, 7 | + |
| LR-CC($n$) | linear response CC($n$) | 3 | 3, 7 | + |
| CISD | CI singles and doubles | 98 | 3 | + |
| CISDT | CI singles, doubles, and triples | 98 | 3 | + |
| CISDTQ | CI singles, doubles, triples, and quadruples | 98 | 3 | + |
| CI($n$) | CI method including up to $n$-tuple excitations | 98 | 3 | + |
| FCI | full CI | 98 | 3 | + |
| LR-MRCCSD | linear response MRCCSD | 137, 138 | 4, 7 | + |
| LR-MRCC($n$) | linear response MRCC($n$) | 4, 138 | 4, 7 | + |
| MRCISD | multireference CI singles and doubles | 139 | 4 | + |
| MRCI($n$) | multireference CI including up to $n$-tuple excitations | 4, 139 | 4 | + |

[a]References describing the methodological developments [b]References describing the implementation in Mrcc [c]Availability of analytic gradients

**Notes**

1. Please note that for excitation energies and geometries LR-CC methods are equivalent to the corresponding EOM-CC models. It is not true for first-order properties and transition moments.
2. With CI methods, excited to excited-state transition moments can also be evaluated.
3. Excited-state harmonic frequencies can be evaluated for the above methods with the help of numerical differentiation of analytical gradients, see Sect. 6.3.
4. Excited-state harmonic frequencies can also be calculated for the above methods via numerical differentiation using the Cfour or Columbus interface.
5. Excited-state harmonic frequencies and second-order properties can be evaluated for CI methods using analytic second derivatives and the Cfour interface.
6. Natural transition orbitals (NTOs) can be computed for the CIS, TDHF, TDA, TD-DFT, ADC(2), CC2, and CIS($D_\infty$) methods, while state-averaged natural orbitals (NOs) can be computed for the second-order methods. See the description of keyword `nto` for details.

## 6.7 Ionized and electron-attached states

Ionization potentials (IPs) and electron affinities (EAs) can be calculated for closed-shell systems using the density-fitting approximation and RHF (RKS) orbitals with the following methods (Ref. 65):

1. configuration interaction singles with perturbative correction for double excitations [CIS(D)]
2. second-order algebraic diagrammatic construction [ADC(2)] approach
3. spin-scaled versions of the latter approaches: SCS-CIS(D), SCS-ADC(2), SOS-CIS(D), and SOS-ADC(2)
4. CIS(D)- and ADC(2)-based double hybrid TD-DFT methods even with range separation

Dyson orbitals can be computed for ADC(2)-based IP/EA states. See the description of keyword `dyson` for details.

## 6.8 Relativistic calculations

Treatment of special relativity in single-point energy calculations is possible for all the CC and CI methods listed in Sect. 6.1 using various relativistic Hamiltonians with the following interfaces (Refs. 18 and 20).

1. With MOLPRO, relativistic calculations can be performed with Douglas–Kroll–Hess Hamiltonians using RHF, UHF, ROHF, and MCSCF orbitals. The interface also enables the use of effective core potentials (see MOLPRO's manual for the specification of the Hamiltonian and effective core potentials).
2. With CFOUR, exact two-component (X2C) and spin-free Dirac–Coulomb (SF-DC) calculations can be performed. The evaluation of mass-velocity and Darwin corrections is also possible using analytic gradients for all the methods and reference functions listed in Sect. 6.2. (See the description of the `RELATIVISTIC` keyword in the CFOUR manual for the specification of the Hamiltonian.)
3. With DIRAC, relativistic calculations can be carried out with the full Dirac–Coulomb Hamiltonian and several approximate variants thereof using Kramers-paired Dirac–Fock orbitals. See Refs. 18 and 20 as well as Sect. 5.3 for more details.

Treatment of special relativity in analytic gradient calculations is possible for all the CC and CI methods listed in Sect. 6.2 using various relativistic Hamiltonians with the following interfaces.

1. With CFOUR, analytic gradient calculations can be performed with the exact two-component (X2C) treatment.
2. With DIRAC, unrelaxed first-order properties can be computed using the Dirac–Coulomb Hamiltonian. See Ref. 18 and Sect. 5.3 for more details.

## 6.9 Reduced-scaling and local correlation calculations

The methods for which reduced-cost algorithms are available are collected in Table 3.

### Speeding up SCF calculations

The different types of SCF calculations can be sped up utilizing various tricks. In the dual basis set methods [39, 155], the SCF equations are solved with a smaller basis set, the resulting density matrix is projected into a large basis set, and then, the Fock matrix is constructed and diagonalized in the large basis (see keyword `dual`). In a similar manner, dual auxiliary bases, dual grids, or dual fitting metrics can also be used [50] (see keywords `dual_df`, `grid_sm`, and `fitting`).

For the reduction of the computational expenses of the exchange contribution to the Fock matrix, several techniques have been implemented. The evaluation of three-center Coulomb integrals and their contribution can be accelerated with the multipole approximation [47] (see keywords `fmm`, `fmmord`). The exchange contribution can also be computed with semi-numerical techniques exploiting pseudo-spectral approximations [50, 156, 157], for which also a dual grid approach is available [50] (see keywords `pssp`, `agrid_pssp`, `agrid_pssp_sm`). Probably the most efficient method to speed up SCF calculations is the local density fitting approach, where products of orbitals are expanded in local domains of fitting functions [24, 31, 158] (see keywords `scfalg`, `excrad`, `excrad_fin`). The efficiency of the latter can further be improved with the OCC-RI-K algorithm [50, 159] algorithm, which can also be used together with non-local direct DF-SCF calculations (see keyword `occri`). Analytic gradients are available for local DF SCF calculations, currently only for closed-shell systems [64].

Table 3: Reduced-cost and reduced-scaling algorithms implemented in MRCC.

| Cost-reduction technique | Methods[a] | References |
|---|---|---|
| Dual basis set | RHF, UHF, ROHF, RKS, UKS, ROKS | 39 |
| Dual auxiliary basis, dual fitting metric, OCC-RI-K, pseudo-spectral exchange | RHF, UHF, ROHF, RKS, UKS, ROKS | 50 |
| Local density fitting | RHF, UHF, ROHF, RKS, UKS, ROKS | 24, 31, 64 |
| | CIS, TDHF, TDDFT, TDA-TDDFT | 40 |
| Multipole methods | RHF, UHF, ROHF, RKS, UKS, ROKS | 47 |
| Natural auxiliary functions | MP2, dRPA, SOSEX, CCSD(T) | 26, 48 |
| | CCSD-F12, CCSD(T)-F12 | 48, 60 |
| | LR-CC2 | 35 |
| | CIS, CIS(D), ADC(2), CIS($D_\infty$) | 37 |
| | CIS(D)- and ADC(2)-based DH-TDDFT | 41 |
| Natural orbitals | CCSD, CCSD(T), CCSDT, CCSDT(Q), . . . | 21 |
| | LR-CC2 | 35 |
| | ADC(2), CIS($D_\infty$) | 37 |
| Combined natural orbitals–natural auxiliary functions | LR-CC2 | 35 |
| | ADC(2), CIS($D_\infty$) | 37 |
| | CCSD, CCSD(T) | 48, 55 |
| | CCSD-F12, CCSD(T)-F12 | 48, 60 |
| Local correlation | MP2, DH-DFT | 31, 51 |
| | dRPA, SOSEX | 30 |
| | CCSD, CCSD(T), CCSDT, CCSDT(Q), . . . | 22, 24, 36, 38, 42, 66 |
| | ADC(2), CIS($D_\infty$), LR-CC2 | 43 |
| | DBBSC | 71 |

[a] The cost-reduction techniques are also available for the spin-scaled versions of MP2, CIS(D), ADC(2), CIS($D_\infty$), and CC2, as well as the corresponding spin-scaled DH-(TD)DFT methods.

### Orbital transformation techniques

The computational expenses of the CC and CI methods listed in Sect. 6.1 can be reduced via orbital transformation techniques (Ref. 21). In this framework, to reduce the computation time, the dimension of the properly transformed occupied and/or virtual one-particle space is truncated. Currently optimized virtual orbitals (OVOs) or MP2 natural orbitals (NOs) can be chosen. This technique is recommended for small to medium-size molecules. This scaling reduction approach is available using RHF or UHF orbitals for virtual orbitals. Occupied NOs are only implemented for RHF orbitals and with density fitting. Spatial symmetry for the frozen NO (FNO) methods can also be utilized with the general-order CC and CI methods of the `mrcc` program. For the correction of the dropped orbitals, besides the conventional ΔMP2 approach, novel PPL+ and (T+) corrections [55] can also be used. See the description of keywords `ovirt`, `eps`, `lnoepsv`, and `ovosnorb` for more details. Reduced-cost ground-state correlation methods can also be activated by adding the `FNO-` prefix to the acronym of the method [e.g., `calc=FNO-CCSD(T)`], see notes for keyword `calc`.

The FNO approach is also an efficient tool to speed up explicitly correlated CCSD(T) calculations [60]. Virtual FNOs can be used with RHF, UHF, and ROHF references. It can also be activated by the `FNO-` prefix, that is, `calc=FNO-CCSD(F12*)(T+)`.

### Natural auxiliary functions

The cost of density-fitting methods can be reduced using natural auxiliary functions (NAFs) introduced in Ref. 26. The approach is very efficient for dRPA, but considerable speedups can also be achieved for MP2, CC2, ADC(2), and CCSD(T) [35, 48] as well as for explicitly correlated MP2 and CCSD(T) [60, 68]. Analytical gradients are also available for the NAF-MP2 method [70]. See the description of keywords `naf_cor` and `naf_scf` for more details.

### Reduced-cost techniques for excited states

The computational expenses of CIS, TD-HF, hybrid TDA and TD-DFT, CC2, CIS($D_\infty$), ADC(2), SCS-CC2, SCS-CIS($D_\infty$), SCS-ADC(2), SOS-CC2, SOS-CIS($D_\infty$), and SOS-ADC(2) excited-state calculations can be efficiently reduced using local fitting domains as well as state-averaged NOs and NAFs (Refs. 35, 37, and 40). A reduced-cost algorithm for the calculation of core excitations is also available (Ref. 63). The scaling of CC2 and ADC(2) calculations can also be decreased by local correlation approaches (Ref. 41). See the description of keywords `redcost_exc` and `redcost_tddft` for more details.

### Local correlation methods

The cost of MP2, dRPA, SOSEX, corresponding double- and dual-hybrid DFT, as well as single-reference iterative and perturbative coupled-cluster calculations can be reduced for large molecules by the local natural orbital CC (LNO-CC) approach (Refs. 22, 24, 30, 31, 36, 38, 51, 66, and 42). This method combines ideas from the cluster-in-molecule approach of Li and coworkers [160], the incremental approach of Stoll *et al.* [161], domain- and pair approximations introduced first by Pulay *et al.* (see, e.g., Ref. 162) with frozen natural orbital, natural auxiliary function, and Laplace transform techniques. All of the above methods are available for closed-shell molecules using RHF (RKS) orbitals, as well as for open-shell molecules using using ROHF (ROKS) orbitals or QROs formed from unrestricted orbitals. [51, 66] The LMP2 and LNO-CCSD(T) implementations are highly competitive, applicable up to a thousand atoms or about 50,000 atomic orbitals [38, 42] with accurate settings and basis sets. Highly-converged LNO settings and quadruple- or quintuple-$\zeta$ basis sets are feasible up to a few hundred atoms [163–165] depending on the system type. Accuracy and efficiency benchmarks as well as a wide range of applications are reviewed for local correlation methods in general and for our LMP2 and LNO-CCSD(T) in particular in Ref. 166. Introductory level summary of the key local correlation approximations and methodological details is given in Section "Reduced-cost and reduced-scaling techniques" of Ref. 2 and Section 2 of Ref. 166.

Further unique features of our local correlation methods include the exceptionally small memory and disk use, frequent checkpointing and restarting capability (see `lccrest`), support for general-order LNO-CC approaches, and treatments of point group symmetry (see `localcorrsymm`), and near-linear dependent AO basis sets [38, 42, 166]. See the description of keywords `localcc`, `lnoepso`, `lnoepsv`, `domrad`, `lmp2dens`, `dendec`, `nchol`, `osveps`, `spairtol`, `wpairtol`, `laptol`, `lccrest`, and `lcorthr` for further details. Extensive benchmarks regarding the accuracy and efficiency of our LMP2 and LNO-CCSD(T) local correlation methods are also provided in Refs. 30, 31, 38, 51, 66, 166, and 42 and Section II.G of Ref. 1. The definition, performance analysis, and the illustration of the systematic convergence of the `lcorthr` settings, as well as of the corresponding LNO extrapolation scheme is documented in Refs. 42 and 166. Extensive independent LNO-CCSD(T) benchmarks and comparisons to alternative local CCSD(T) approaches are also available (see Table 4 and Figure 8 of Ref. 166 as well as references therein).

### Multi-level local correlation methods

Utilizing the above local correlation techniques a multi-level scheme is defined in which the LMOs are classified as active or environment (Refs. 33 and 39). The contributions of these LMOs to the total correlation energy are evaluated using different models for the two subsystems, for instance, one can choose a LNO-CC model for the active subsystem and LMP2 for the environment. See the description of keyword `corembed` for further details.

## 6.10 Optimization of basis sets

The optimization of basis set's exponents and contraction coefficients can be performed with any method for which single-point energy calculations are available (see Sect. 6.1). The implementation is presented in Ref. 27. The related keywords are

`basopt` – to turn on/off basis set optimization

`optalg` – to select an algorithm for the optimization

`optmaxit` – maximum number of iterations allowed

`optetol` – convergence criterion for energy change

`optstol` – convergence criterion for parameter (exponent, contraction coefficient) change

For their detailed description see Sect. 12.

For the optimization of basis sets it is important to know the format for the storage of the basis set parameters. In MRCC the format used by the CFOUR package is adapted. The format is communicated by the following example.

| `actual lines` | description |
|---|---|
| `C:6-31G` | ↩ Carbon atom:basis name |
| `Pople's Gaussian basis set` | ↩ comment line |
| | ↩ blank line |
| `2` | ↩ number of angular momentum types |
| `0           1` | ↩ 0→s , 1→p |
| `3           2` | ↩ number of contracted functions |
| `10          4` | ↩ number of primitives |
| | ↩ blank line |
| `3047.5249      457.36952     ...` | ↩ exponents for s functions |
| | ↩ blank line |
| `0.0018347      0.0000000     0.0000000` | ↩ contraction coefficients for s functions |
| `0.0140373      0.0000000     0.0000000` | |
| `0.0688426      0.0000000     0.0000000` | |
| `0.2321844      0.0000000     0.0000000` | |
| `0.4679413      0.0000000     0.0000000` | |
| `0.3623120      0.0000000     0.0000000` | |
| `0.0000000      0.1193324     0.0000000` | |
| `0.0000000      0.1608542     0.0000000` | |
| `0.0000000      1.1434564     0.0000000` | |
| `0.0000000      0.0000000     1.0000000` | |
| | ↩ blank line |
| `7.8682724      1.8812885     ...` | ↩ exponents for p functions |
| | ↩ blank line |
| `0.0689991      0.0000000` | ↩ contraction coefficients for p functions |
| `0.3164240      0.0000000` | |
| `0.7443083      0.0000000` | |
| `0.0000000      1.0000000` | |

In a basis set optimization process you need two files in the working directory: the appropriate `MINP` file with the `basopt` keyword set and a user supplied `GENBAS` file that contains the basis set information in the above format. You do not need to write the `GENBAS` file from scratch, you can use the files in the `BASIS` directory of Mrcc to generate one or you can use the Basis Set Exchange [167–170] to download a basis in the appropriate form (`CFOUR` format). Note that you can optimize several basis sets at a time: all the basis sets which are added to the `GENBAS` file will be optimized.

You can perform unconstrained optimization when all the exponents and contraction coefficients are optimized except the ones which are exactly 0.0 or 1.0. Alternatively, you can run constrained optimizations when particular exponents/coefficients or all exponents and coefficients for a given angular momentum quantum number are kept fixed during the optimization. The parameters to be optimized can be specified in the `GENBAS` file as follows.

1. Unconstrained optimization: no modifications are needed—by default all exponents and contraction coefficients will be optimized except the ones which are exactly 0.0 or 1.0.

2. Constrained optimization: by default all the exponents and coefficients will be optimized just as for the unconstrained optimization. To optimize/freeze particular exponents or

coefficients special marks should be used:

- use the "`--`" mark (without quotes) if you want to keep an exponent or coefficient fixed during the optimization. You should put this mark right after the fixed parameter (no blank space is allowed). If this mark is attached to an angular momentum quantum number, none of the exponents/coefficients of the functions in the given shell will be optimized except the ones which are marked by "`++`".
- use the "`++`" mark (without quotes) if you want a parameter to be optimized. Then you should put this mark right after it (no blanks are allowed). You might wonder why this is needed if the default behavior is optimization. Well, this makes life easier. If you want to optimize just a few parameters, it is easier to constrain all parameters first then mark those, which are needed to be optimized (see the example below).

Examples:

1. To reoptimize all parameters in the above basis set but the exponents and coefficients of s-type functions you should copy the basis set to the `GENBAS` file and put mark "`--`" after the angular momentum quantum number of 0. The first lines of the `GENBAS` file:

```
C:6-31G
Pople's Gaussian basis set

2
0--         1
3           2
10          4

3047.5249     457.36952    ...
```

2. Both s- and p-type functions are fixed but the first s-exponent:

```
C:6-31G
Pople's Gaussian basis set

2
0--           1--
3             2
10            4

3047.5249++       457.36952    ...
```

During the optimization the `GENBAS` file is continuously updated, and if the optimization terminated successfully, it will contain the optimized values (in this case it is equivalent to the `GENBAS.opt` file, see below, the only difference is that the file `GENBAS.opt` may contain the special marks, i.e., "`++`", "`--`"). Further files generated in the optimization are:

- `GENBAS.init` – the initial `GENBAS` file saved

- `GENBAS.tmp` – temporary file, updated after each iteration, can be used to restart conveniently a failed optimization process
- `GENBAS.opt` – this file contains the optimized parameters after a successful optimization.

# 7 Installation

## 7.1 Installation of pre-compiled binaries

After registration at the Mrcc homepage, pre-built binaries are available in the download area. These binaries were compiled with the Intel compiler (and Intel MKL) version 2021.10.0 and should utilize optimal instruction sets (e.g., AVX-512) of modern CPUs. If your CPU is still not correctly identified, the `MKL_ENABLE_INSTRUCTIONS` environment variable enables you to use an architecture-specific code path of your choice in Intel MKL routines. To install these executables, Linux operating system and the 2.34 or later version of the GNU C Library (glibc) is required. To use the MPI-parallel executables, Intel MPI 2021 also has to be installed. The precompiled binaries are linked against the Intel MPI 2021.10 library, and it is highly recommended to use this particular Intel MPI version with the precompiled binaries. It is also strongly suggested to install the newest stable version of Libfabric (1.9.0 or later) from the https://github.com/ofiwg/libfabric repository as some of the previous versions provided via the Intel MPI package could cause irregular runtime behavior. The binaries are provided in a gzipped tar file, `mrcc.YY.R.P.binary.tar.gz`, where `YY` is the year of the release, `R` stands for the major release number in that year, and `P` is the patch level. Note that you will find several program versions on the homepage. Unless there are overriding reasons not to do so, please always download the last version. To unpack the file type

```
tar xvzf mrcc*.binary.tar.gz
```

Please do not forget to add the name of the directory where the executables are placed to your `PATH` environment variable.

## 7.2 Installation from source code

To install Mrcc from source code some version of the Unix operating system, Fortran 90 and C compilers as well as BLAS (basic linear algebra subprograms) and LAPACK (linear algebra package) libraries are required. The tested BLAS and LAPACK implementations are Intel MKL (oneMKL), AOCL-BLIS, and AOCL-libFLAME. Optionally, Mrcc can also be linked with the Libxc library of density functionals [171–173] and the PCMSolver library for continuum solvation [96, 97]. For an MPI-parallel build, a working MPI installation is also required. Please be sure that the directories where the compilers are located are included in your `PATH` environment variable. Please also check your `LD_LIBRARY_PATH` environment variable, which must include the directories containing the BLAS and LAPACK and, if linked against, the Libxc, PCMSolver, and MPI libraries.

After registration at the Mrcc homepage the program can be downloaded as a gzipped tar file, `mrcc.YY.R.P.tar.gz`, where `YY` is the year of the release, `R` stands for the major release number in that year, and `P` is the patch level. Note that you will find several program versions on the homepage. Unless there are overriding reasons not to do so, please always download the last version. To unpack the file type

```
tar xvzf mrcc*.tar.gz
```

To install Mrcc add the current working directory to `PATH` (e.g., `export PATH=.:$PATH` on bash) and run the `build.mrcc` script as

`build.mrcc` [<*compiler*>] [`-i`<*option1*>] [`-p`<*option2*>] [`-g`] [`-d`] [`-s`] [`-f`<*folder*>] [`-l`<*library*>] [`-L`<*lspath*>] [`-nomkl`]

<*compiler*> specifies the compiler to be used. The options are as follows. Note that currently only the Intel compiler system is supported. The freely available Intel oneAPI compilers (with the base and hpc toolkits) are recommended if an existing Intel compiler installation is not available. Previously, the other listed compiler systems were also used to generate working binaries.

| | |
|---|---|
| `Intel` | Intel compiler |
| `GNU` | GNU compiler (g77 or gfortran) |
| `PGF` | Portland Group Fortran compiler |
| `G95` | G95 Fortran 95 compiler |
| `PATH` | Pathscale compiler |
| `HP` | HP Fortran Compiler |
| `DEC` | Compaq Fortran Compiler (DEC machines) |
| `XLF` | XL Fortran Compiler (IBM machines) |
| `Solaris10` | Sun Solaris10 and Studio10 Fortran Compiler (AMD64) |

If the `build.mrcc` script is invoked without specifying the <*compiler*> variable, a help message is displayed.

Optional arguments:

`-i` specifies if 32- or 64-bit integer variables are used. Accordingly, <*option1*> can take the value of `32` or `64`. The 32-bit integer version is not supported any more.
Default: `64` for 64-bit machines, `32` otherwise.

`-p` generates parallel code using message passing interface (MPI) or open multi-processing (OpenMP) technologies. Accordingly, <*option2*> can take the `MPI`[=<*MPI implementation*>] or `OMP` values. OpenMP and MPI parallelizations have been tested with the Intel compiler and the Intel MPI and Open MPI implementations. If `-pMPI` is specified, the library given by <*MPI implementation*> will be linked to Mrcc. The default value of <*MPI implementation*> is `IntelMPI` with the Intel compiler and `OpenMPI` for other compilers. Please note that currently the two parallelization schemes can only be combined for the `scf`, `mrcc`, and `ccsd` programs, other executables will be compiled with only OpenMP parallelization even if `-pOMP` and `-pMPI` are both set.
Default: no parallelization.

`-g` source codes are compiled with debugging option (use this for development purposes)

Default: no debugging option.

`-d` source codes are compiled for development, no optimization is performed (use this for development purposes)
Default: codes are compiled with highest level optimization.

`-f` specifies the installation folder. Executables, basis set libraries, and test jobs will be copied to directory <*folder*>. If this flag is not used, you will find the executables, etc. in the directory where you perform the installation.

`-l` MRCC is linked with an external library. The `libxc` and `pcm` options for <*library*> require the installation of the LIBXC and the PCMSOLVER libraries, respectively. See notes below. Other values for `-l`<*library*> are passed to the linker unchanged.

`-L` Adds <*lspath*> to the library search path while linking.

`-nomkl` Turns off linking with Intel MKL library when using the Intel compiler. If this option is used, the BLAS and LAPACK libraries must be specified manually using `-l`, see example below. It is recommended to use Intel MKL library for best performance.

`-s` MRCC is linked statically. Non-MPI-parallel executables are linked entirely statically, while MPI-parallel executables link Intel-provided libraries statically.

Notes:

1. After the installation please do not forget to add the directory where the MRCC executables are located to your `PATH` environment variable. This is the <*folder*> directory if you used the `-f` flag at the installation, otherwise the directory where you executed the `build.mrcc` script.

2. The `build.mrcc` script has been tested on several platforms with several versions of the supported compilers and libraries. Nevertheless you may need to customize the compiler flags, names of libraries, etc. These data can be found in the `build.mrcc.config` file, please edit this file if necessary. Please do not change `build.mrcc`.

3. To ensure the best performance of the software, the use of Intel compiler is recommended. The current release of MRCC has been tested with the 2021.10 version of the Intel compiler (freely available from Intel).

4. If you use MRCC together with MOLPRO, you can also use the MOLPRO installer to install MRCC, please follow the instructions in the MOLPRO manual (`www.molpro.net`).

5. If MRCC is linked with the LIBXC library, LIBXC must be installed before starting the installation of MRCC. Note that you must compile LIBXC with the same Fortran compiler as used for the installation of MRCC. At the installation of LIBXC, it is recommended to set the installation path of LIBXC (`--prefix=<install dir>` option of `configure`) to the directory where the installation of MRCC is carried out, otherwise please set the `LIBS_LIBXC` environment variable to the installation path of LIBXC (i.e., `export LIBS_LIBXC=<install dir>` in bash, where `<install dir>` contains `lib/libxcf03.a` and `lib/libxc.a`) before running `build.mrcc`. See the manual for the LIBXC project

for details [173], as well as the examples below. The current release of Mrcc has been tested with the 7.0.0 version of Libxc.

6. If Mrcc is linked with the PCMSolver library, PCMSolver must be installed before starting the installation of Mrcc. At the installation of PCMSolver, it is recommended to set the installation path of PCMSolver (`--prefix=<install dir>` option of `setup.py`) to the directory where the installation of Mrcc is carried out, otherwise please set the `LIBS_PCM` environment variable to the installation path of PCMSolver (i.e., `export LIBS_PCM=<install dir>` in bash, where `<install dir>` contains `lib/libpcm.so`, or `lib/libpcm.a` in the case of static linking) before running `build.mrcc`. See the manual for the PCMSolver project for details [97], as well as the examples below. The current release of Mrcc has been tested with the 1.3.0 version of PCMSolver.

7. If the program is compiled for multi-node parallel execution, MPI-parallel executables (`*_mpi`) are generated. The compilation is performed by mpiifort when <*MPI implementation*> = `IntelMPI` is set, otherwise the mpifort compiler wrapper is used.

8. For the compilation of MPI-parallel executables, a working MPI installation is necessary. Currently Open MPI version 4 and Intel MPI (2017 or later) implementations are supported. Open MPI has to be patched with commits 07830d0 and 51acbf7 from https://github.com/open-mpi/ompi. Please, consult Sect. 9.3 for additional MPI settings required at runtime.

Examples:

1. Compile Mrcc for OpenMP parallel execution with the Intel compiler (recommended):
   ```
   build.mrcc Intel -pOMP
   ```

2. Compile Mrcc for OpenMP parallel execution with the Intel compiler and install it to the `/prog/mrcc` directory (recommended):
   ```
   build.mrcc Intel -pOMP -f/prog/mrcc
   ```

3. Compile Mrcc for OpenMP and combined OpenMP-MPI parallel execution with the Intel compiler and Intel MPI, and link with the Libxc and the PCMSolver libraries supposing that Mrcc is compiled in the `/prog/mrcc` directory. This will enable all features of Mrcc and is highly recommended.

   (a) Installation of the Libxc library:
   Download the Libxc library (`libxc*.tar.bz2`) from the homepage of the Libxc project [173].
   ```
   tar xvjf libxc*.tar.bz2
   cd libxc*
   autoreconf -i
   ./configure --prefix=/prog/mrcc/ FC=ifort --enable-kxc
   make
   make check
   make install
   cd /prog/mrcc
   ```

    (b) Installation of the PCMSolver library:

    ```
    curl -L https://github.com/PCMSolver/pcmsolver/archive/v1.3.0.tar.gz |
    tar -xz
    cd pcmsolver-1.3.0
    ```

    Replace `-openmp` by `-qopenmp` in file `cmake/downloaded/autocmake_omp.cmake`

    ```
    cp cmake/custom/compilers/Intel/C.make
                               cmake/custom/compilers/Intel/C.cmake
    ./setup.py --cxx=icpc --cc=icc --fc=ifort --int64 --omp --prefix=/prog/mrcc/
    cd build
    make
    make install
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/prog/mrcc/lib
    cd /prog/mrcc
    cp pcmsolver-1.3.0/api/pcmsolver.f90 .
    ```

    (c) Compiling and linking MRCC:

    ```
    build.mrcc Intel -pOMP -pMPI=IntelMPI -llibxc -lpcm
    ```

4. Compile MRCC for serial execution with the Intel compiler:

   ```
   build.mrcc Intel
   ```

5. Compile MRCC for parallel execution using MPI environment with the Intel compiler for 32-bit machines:

   ```
   build.mrcc Intel -i32 -pMPI
   ```

6. Compile MRCC with the Intel compiler for parallel execution using OpenMP and MPI parallelization through the Open MPI library:

   ```
   build.mrcc Intel -pOMP -pMPI=OpenMPI
   ```

7. Compile MRCC with the Intel compiler using BLAS and LAPACK libraries other than Intel MKL installed in a standard directory:

   ```
   build.mrcc Intel -nomkl -lblas -llapack
   ```

8. Compile MRCC with the Intel compiler using BLAS and LAPACK libraries other than Intel MKL, the shared object files libblas.so and liblapack.so can be found in <*path-to-blas*> and <*path-to-lapack*>, respectively:

   ```
   build.mrcc Intel -nomkl -L<path-to-lapack> -llapack -L<path-to-blas> -lblas
   ```

## 7.3 Installation under Windows

Under the Windows operating system the pre-built binaries cannot be directly executed, and the direct compilation of the source code has not been attempted so far. For Windows users we recommend the use of virtualization software packages, such as VIRTUALBOX, which allow Linux as a guest operating system. In that environment MRCC can be installed in the normal way as described in the previous subsections.

# 8 Testing Mrcc

Once you have successfully installed Mrcc, you may wish to test the correctness of the installation. For that purpose, numerous test jobs are at your disposal. The corresponding input files can be found in the `MTEST` directory created at the installation, where a test script, `mtest`, is also available. Your only task is to change to the `MTEST` directory and execute the `mtest` script. (Please do not forget to add the directory where the Mrcc executables are located to your `PATH` environment variable.) The test jobs will be automatically executed and you will receive feedback about the results of the tests. The corresponding output files will be left in the `MTEST` directory, and you can also check them. If all the tests complete successfully, your installation is correct with high probability.

The execution of the test jobs will take for a couple of hours. If you want to run the test on another machine, e.g., on a node of a cluster, you should copy the entire `MTEST` directory to that machine and start the `mtest` script there.

Please note that there are some test jobs that allocate a small amount of memory to test the out-of-core algorithms of the program (`MINP_*smallmem`). These test jobs run with four threads by default when testing OpenMP-parallel executables (i.e., the `build.mrcc` script was run with the `-pOMP` switch) with the `mtest` script. If you run these test jobs with more than four threads, some of them will fail since the memory requirement for OpenMP-parallel runs grows with the number of threads. In this case, the failure of these tests does not indicate a problem with your installation.

Testing of the MPI-parallel executables is also supported. The `mtest` script automatically detects MPI-parallel executables in `PATH` if the `build.mrcc` script was run with the `-pMPI` switch and runs the appropriate MPI-parallel test jobs (`MINP_*_MPI`).

To test other external libraries, two switches can be used. With the `-l` switch, Libxc test jobs (containing `_Libxc` in the name) are also run, while with the `-p` switch, the `PCMSolver` library is also tested (`_PCM` in the file name). The two options can also be used at the same time, i.e., if `mtest` is invoked as `mtest -l -p`, all the test jobs will be executed.

Testing of the features available with external semi-empirical and MM programs, such as ONIOM models including semi-empirical layers or RISM solvation calculations, is also automated. The `mtest` script automatically searches for those programs, and the corresponding test jobs are executed if they are available.

Please note that you can also create your own test jobs, e.g., if you modify the code, compile the program with new compiler versions, or use unusual combination of keywords. To that end you should calculate a reliable energy for your test job (e.g., using a stable compiler version), include the `test` keyword and the calculated energy to the `MINP` file (see the description of the keyword for more details), and copy the `MINP` file to the `MTEST` directory renaming it as `MINP_`<*job_name*>. Then the new job will be automatically executed when the `mtest` script is invoked next time.

# 9 Running Mrcc

Please be sure that the directory where the Mrcc executables are located are included in your `PATH` environment variable. Note that the package includes several executables, and all of them must be copied to the aforementioned directory, not only the driver program `dmrcc`.

Please also check your `LD_LIBRARY_PATH` environment variable, which must include the directories containing the libraries linked with the program. This variable is usually set before the installation, but you should not change by removing the names of the corresponding directories. Please do not forget to copy the input file `MINP` (see Sect. 11) to the directory where the program is invoked.

## 9.1 Running MRCC in serial mode

To run MRCC in serial the user must invoke the driver of the package by simply typing

```
dmrcc
```

on a Unix console. To redirect the input one should execute `dmrcc` as

```
dmrcc > out
```

where `out` is the output file.

## 9.2 Running MRCC in parallel using OpenMP

Several executables of the package can be run in OpenMP parallel mode, hence it is recommended to use this option on multiprocessor machines.

The pre-built binaries available at the MRCC homepage support OpenMP-parallel execution. If you prefer source-code installation, to compile the program for OpenMP parallel execution you need to invoke the `build.mrcc` script with the `-pOMP` option at compilation (see Sect. 7). The OpenMP parallelization has been tested with the Intel compiler. Please be careful with other compilers, run, e.g., our test suite (see Sect. 8) with the OpenMP-complied executables before production calculations.

To run the code with OpenMP you need to set the environment variable `OMP_NUM_THREADS` to the number of cores you want to use. E.g., under Bourne shell (bash):

```
export OMP_NUM_THREADS=4
```

Then the program should be executed as described above.

The provided binaries are linked with threaded Intel MKL routines, thus, when those are executed, the environment variable `MKL_NUM_THREADS` should also be set, e.g.:

```
export MKL_NUM_THREADS=4
```

If source-code installation is preferred, it is recommended to link the MRCC objects with threaded BLAS and LAPACK libraries and employ the required runtime settings of the employed libraries (e.g., define `MKL_NUM_THREADS` for Intel MKL).

The binding of threads to hardware elements might affect the performance on certain systems. The thread affinity can be specified with the OpenMP environment variables `OMP_PLACES` and `OMP_PROC_BIND` or the Intel MKL specific variable `KMP_AFFINITY` when the precompiled binaries are executed or Intel MKL is used. For the computations involving the `ccsd` program, such as DF-, FNO-, and LNO-CCSD computations, nested parallelism is utilized. It is suggested to try setting the `OMP_PLACES=cores` and `OMP_PROC_BIND=spread,close` for improved performance. However, in some cases, such thread binding may negatively effect the performance (e.g., when the number of processes exceeds the number of physical CPUs).

## 9.3 Running Mrcc in parallel using MPI

Currently executables `scf`, `mrcc`, and `ccsd` can be run in parallel using the MPI technology. To compile the program for MPI-parallel execution, you need to invoke the `build.mrcc` script with the `-pMPI` option at compilations (see Sect. 7). It has been tested with the Intel compiler and the Open MPI (version 4) and Intel MPI (2017 or later) environments. If the precompiled binaries are used or Intel MPI 2021 or newer is linked to Mrcc, it is strongly recommended to install and use the newest stable version of Libfabric (1.9.0 or later) as some of the previous versions provided via the Intel MPI package could cause irregular runtime behavior. The Libfabric library can be downloaded from https://github.com/ofiwg/libfabric. If not the Intel provided Libfabric library is used, the environment for Intel MPI should be set using the `-ofi_internal=0` option of `mpivars.sh` (e.g., if Intel MPI 2019 is installed, `source` <*Intel MPI install dir*>`/parallel_studio_xe_2019/`
`compilers_and_libraries_2019/linux/mpi/intel64/bin/mpivars.sh`
`release_mt -ofi_internal=0`.

For the MPI-parallel execution, the `mpitasks` keyword has to be set. Then, it is sufficient to execute the `dmrcc` binary as usual. The program will spawn the number of `scf`, `mrcc`, or `ccsd` processes specified with the `mpitasks` keyword and copy the necessary input files to the compute nodes, therefore the input files need to be present only in the directory where `dmrcc` is executed. Note that the working directory can be the same for all MPI processes, e.g., a directory in the network file system of a computer cluster. Alternatively, process-specific working directories are also supported to exploit local hard drives within compute nodes. In both cases the spawned MPI process will start the execution in the directory with the same path, which might be on a separate file system.

If you wish to run Mrcc with other `mpirun` options, the MPI-parallel `dmrcc_mpi` executable should be launched as `mpirun -np 1 <options> dmrcc_mpi`. You should not run `dmrcc` using `mpirun` since it will result in launching `mpirun` recursively, and your job might fail to start. Please note that the total number of processes will be higher than `mpitasks`, so you might need to oversubscribe nodes using the appropriate `mpirun` or scheduler option (e.g., `sbatch --overcommit ...` with SLURM or `mpirun --oversubscribe ...` with Open MPI). For optimal performance, please set `mpitasks` at the total number of available CPUs, non-uniform memory access (NUMA) nodes, nodes, cores, etc., as the additional number of processes on top of `mpitasks` are driver processes running mostly in the background and do not require dedicated resources.

On systems consisting of more than one NUMA node (e.g., containing more than one CPU), the performance may be increased by running one process on each NUMA node of the compute nodes. This strategy is beneficial, for instance, when the number of OpenMP processes would otherwise surpass a few tens. Instead, the number of MPI tasks can be increased for better parallel efficiency. Note, however, that in this case the total memory requirement is increased because each process allocates the amount of memory specified in the input file as all MPI algorithms currently available in Mrcc rely on replicated memory strategies.

Pinning processes to CPU cores in MPI parallel runs might affect the performance. When Open MPI is linked to Mrcc, binding can be set by the `-bind-to` option of `mpirun`, via modular component architecture (MCA) parameters (e.g., `--mca hwloc_base_binding_policy core`), or setting the environment variable `OMPI_MCA_hwloc_base_binding_policy`. It is also suggested to set the Open MPI MCA parameter `rmaps_base_inherit` to 1. In the case Mrcc is linked with Intel MPI or the precompiled binaries are used, pinning can be controlled by the `I_MPI_PIN` and `I_MPI_PIN_PROCESSOR_LIST` environment variables. If the internode connection is established via an InfiniBand network, other MCA parameters might need to be set as well (e.g., `btl_openib_allow_ib` to `true`).

## 9.4 Troubleshooting MPI

To run Mrcc in parallel using MPI, the environment needs to be set up correctly. You need a working MPI installation with the Mrcc executables available on all nodes used for the job. MPI can be set up by setting the appropriate environment variables (e.g., `PATH`, `LD_LIBRARY_PATH`), sourcing the setup script that comes with MPI (e.g., if Intel MPI is used) or maybe in other ways (e.g., modules).

MPI might need to be compiled with your chosen resource manager's integration if you plan to use its own tools to launch MPI processes. If you do not use a resource manager, you should be able to ssh/rsh between the compute nodes. In this case, you might need to set the proper startup method with the appropriate option (such as `ssh, slurm, pbs, ...`) of your MPI implementation (e.g., the `I_MPI_HYDRA_BOOTSTRAP` environment variable or the -bootstrap option of mpirun in the case of Intel MPI). In the case, you use Intel MPI with the SLURM resource manager, you should set the `I_MPI_PMI_LIBRARY` environment variable pointing to SLURM's own PMI library. Alternatively, it is suggested that you do not apply a third party PMI library (`unset I_MPI_PMI_LIBRARY`) and use the mpirun command with ssh bootstrap.

Depending on the hardware and software setup of the network on your cluster, you may need to adjust the fabric interface provider. For instance, for Intel MPI, use the FI_PROVIDER variable, e.g., with `verbs, tcp,` etc.

For further details, please refer to the documentation of your chosen MPI implementation. You may also find useful tips to solve MPI-related issues on the Mrcc user forum (thread no. 1031, 1032, 1092, 1094). In the case of runtime issues (e.g., the program hangs in an MPI parallel job step or runs with error), it may be helpful to increase the verbosity level, e.g., with Intel MPI via setting the environment variables as `I_MPI_DEBUG=5`, `I_MPI_HYDRA_DEBUG=1`, and `I_MPI_OFI_PROVIDER_DUMP=1`.

In order to test if the problem is related to the MPI settings, not optimal but usually accessible options can be tried. For instance, for Intel MPI you may try to set one or more of the following:

```
unset I_MPI_PMI_LIBRARY
export I_MPI_HYDRA_BOOTSTRAP=ssh
export I_MPI_OFI_PROVIDER=tcp
```

and depending on the Intel MPI version:

```
source <Intel MPI install dir>/parallel_studio_xe_2019/
compilers_and_libraries_2019/linux/mpi/intel64/bin/mpivars.sh
release_mt -ofi_internal=1
```

or

```
source <Intel MPI install dir>/oneapi/mpi/latest/env/vars.sh
-i_mpi_library_kind=release_mt -i_mpi_ofi_internal=1.
```

# 10 The programs of the suite

In this section, we discuss the major characteristics of the programs of the Mrcc package, and also provide some information about their use and the corresponding outputs.

`dmrcc` Driver for the program system. It calls the programs of the suite (except `build.mrcc`). It is recommended to run always `dmrcc`, but advanced users may run the programs one-by-one (e.g., for the purpose of debugging). See also Sect. 9 for further details.

`minp` Input reader and analyzer. This program reads the input file `MINP`, checks keywords, options, and dependencies; sets default values for keywords.

`integ` An open-ended atomic orbitals integral code. This code reads and analyzes the molecular geometry, reads the basis sets, and calculates one- and two-electron integrals as well as property integrals over Gaussian-type atomic orbitals. Both the Obara–Saika and the Rys quadrature schemes are implemented for the evaluation of two-electron integrals. In principle integrals over basis functions of arbitrary high angular momentum can be evaluated using the Obara–Saika algorithm.

`scf` Hartree–Fock and Kohn–Sham SCF code. It solves the RHF, UHF, ROHF, RKS, UKS, or ROKS equations using either conventional or direct SCF techniques. It also performs the semi-canonicalization of orbitals (if requested) for ROHF wave functions.

`orbloc` Orbital localization program. It performs the localization of MOs using the Cholesky, Boys, or generalized Boys procedures. It also constructs the domains for local correlation calculations.

`drpa` An efficient three-index integral transformation, density-fitting MP2, RPA, dRPA, SOSEX, and RPAX2 code. The dRPA method is implemented using the modified algorithm of Ref. 117, which scales as the fourth power of the system size, see Ref. 26. `drpa` is also the driver for local correlation calculations.

`mulli` Domain construction for local correlation calculations. It assigns the localized MOs (LMOs) to atoms using the Boughton–Pulay method, and for each occupied LMO it constructs a domain of occupied and virtual LMOs on the basis of their spatial distance. Projected atomic orbitals (PAOs) are also constructed if requested.

`ovirt` Integral transformation and orbital optimization code. This program performs the four-index integral transformations of AO integral for correlation calculations. It also carries out the construction of optimized virtual orbitals (OVOs) or MP2 natural orbitals in the case of reduced-cost CC calculations.

`ccsd` A very fast, hand-coded, MO-integral-based (DF) CCSD and CCSD(T) code. The code has been designed for local CC calculations but is also very efficient for conventional CC calculations. Currently it only functions for closed-shell systems, and the spatial symmetry is not utilized.

`uccsd` A fast, hand-coded, MO-integral-based open-shell (DF) CCSD and CCSD(T) code. The code has been designed for local CC calculations but is also efficient for conventional CC calculations. Currently, the spatial symmetry is not utilized.

`mp2f12` An efficient explicitly correlated MP2 (MP2-F12) program. It also computes the intermediates for explicitly correlated CCSD and CCSD(T) calculations. The MPI implementation only works for closed shell systems and small memory runs are only supported without FNO/NAF/NAB approaches.

`cis` A very fast, hand-coded, integral direct DF CIS, TDA, TD-DFT, ADC(2), CIS(D), CIS($D_\infty$), and CC2 code. Currently it only works for closed-shell systems, and the spatial symmetry is not utilized.

`prop` This program solves the CPHF/KS equations, constructs relaxed density matrices, calculates first-order molecular properties and Cartesian gradients.

`qmmod` Interface program for QM/MM and embedding calculations.

`goldstone` This program generates the formulas for `mrcc`. The program also estimates the memory requirement of the calculation. This is a very crude (the symmetry and spin is not treated exactly) but quick estimate. The real memory requirement, which is usually much smaller, is calculated by `xmrcc` after the termination of `goldstone`.

`xmrcc` It calculates the exact memory requirement for `mrcc`. Note that it may take a couple of minutes for complicated wave functions (e.g., MRCC derivatives). It prints out five numbers at the end (in MBytes):

> `Real*8` Minimal and optimal memory for double-precision (real*8) arrays.
>
> `Integer` Memory allocated by `mrcc` for integer arrays.
>
> `Total` (= `Real*8` + `Integer`) The minimal and the optimal amount of total required memory. It is not worth starting the calculation if the real physical memory of the machine is smaller than the `Minimal` value. The performance of the program is optimal if it can use at least as much memory as the `Optimal` value. If the memory is between the `Minimal` and `Optimal` values, out-of-core algorithms will be executed for particular tasks, and it may result in slow down of the code. Please note that the memory available to the program can be specified by keyword `mem` (see page 115).

`mrcc` Automated, string-based many-body code. It performs the single-point energy as well as derivative calculations for general CC, LR-CC, and CI methods. Abelian spatial symmetry is utilized and a partial spin adaptation is also available for closed-shell systems.

`build.mrcc` Installation script of the suite. See Sect. 7 for a detailed description.

# 11 Input files

The input file of the Mrcc package is the `MINP` file. This file must be placed in the directory where the program is invoked. In addition, if you use your own basis sets (see keyword `basis`), angular integration grids for DFT calculations (see keyword `agrid`), or Laplace-quadrature for Laplace transform calculations (see keyword `dendec`), you may also need the `GENBAS` file, and then it must be also copied to the above directory.

In general, the execution of MRCC is controlled by keywords. The list of the keywords is presented in Sect. 12. The keywords and the corresponding options must be given in the `MINP` file as

. . .
*<keyword>*[=*<option>*]
. . .

You can add only one keyword per line, but there are keywords which require multiple-line input, and the corresponding variables must be specified in the subsequent lines as

. . .
*<keyword>*[=*<option>*]
*<input record 1>*
*<input record 2>*
. . .
*<input record n>*
. . .

The input is not case-sensitive. Any number of lines can be left blank between two items, however, if a keyword requires multiple-line input, the lines including the keyword and its input records cannot be separated. Under similar conditions any line can be used for comments, but the beginning of a comment line must not be identical to a keyword because that line may be identified as a keyword by the input reader and misinterpreted. Thus it is recommended to start comment lines with some special character, e.g., hash mark.

Please note that you can find input files for numerous test jobs in the `MTEST` directory created at the installation of MRCC (see Sect. 8). The input files have self-explanatory names and also include a short description at the beginning. You should look at these files for examples for the structure of the input file and the use of various keywords. You can use these files as templates, but please note that these files have been created to thoroughly check the correctness of the code and the installation, and thus some of them contain very tight convergence thresholds as well as unusual combination of (auxiliary) basis sets. In production calculations, you should use the default convergence thresholds (i.e., delete the lines including keywords `itol`, `scftol`, `cctol`, etc.), select the basis set carefully (i.e., set the appropriate option for keyword `basis`), and use the default auxiliary basis sets (i.e., delete the lines including keywords `dfbasis_scf` or `dfbasis_cor`). Please also do not forget to remove keyword `test` and to specify the amount of memory available to the program by setting the `mem` keyword.

# 12 Keywords

In this section, the keywords of the MRCC input file are listed in alphabetical order. To help the reader, the list of the keywords can also be found at the end of the manual.

`active` The active orbitals for multi-reference (active-space) CI/CC calculations can be specified using this keyword. Note that this keyword overwrites the effect of keywords `nacto` and `nactv`. Note also that this keyword only sets the active orbitals for the post-SCF calculation, the MCSCF active orbitals can be specified by keyword `mact`.

Options:

`none` All orbitals are inactive (i.e., single-reference calculation).

`serialno` Using this option one can select the active orbitals specifying their serial numbers. The latter should be given in the subsequent line as $< n_1 >$,$< n_2 >$,...,$< n_k >$-$< n_l >$,..., where $n_i$'s are the serial numbers of the correlated orbitals. Serial numbers separated by dash mean that $< n_k >$ through $< n_l >$ are active. Note that the numbering of the orbitals is relative to the first correlated orbital, that is, frozen orbitals are excluded.

`vector` Using this option one can set the active/inactive feature for each correlated orbital. In the subsequent line an integer vector should be supplied with as many elements as the number of correlated orbitals. The integers must be separated by spaces. Type 1 for active orbitals and 0 for inactive ones.

Default: `active=none`

Examples:

1. We have 20 correlated orbitals. Orbitals 1, 4, 5, 6, 9, 10, 11, 12, and 14 are active. Using the `serialno` option the input should include the following two lines:
```
active=serialno
1,4-6,9-12,14
```
2. The same using the `vector` option:
```
active=vector
1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0
```

`agrid` Specifies the angular integration grid for DFT calculations. Angular grids are taken from the `Grid` file which is located in the `BASIS` directory created at the installation. By default, the 6-, 14-, 26-, 38-, 50-, 74-, 86-, 110-, 146-, 170-, 194-, 230-, 266-, 302-, 350-, 434-, 590-, 770-, 974-, 1202-, 1454-, and 1730-point Lebedev quadratures [174] are included in the file, which are labeled, respectively, by `LD0006`, `LD0014`, etc. In addition to the above grids, any angular integration grid can be used by adding it to the `BASIS/Grid` file or alternatively to the `GENBAS` file to be placed in the directory where MRCC is executed. The format is as follows. On the first line give the label of the grid as `XXNNNN`, where `XX` is any character and `NNNN` is the number of the grid points (see the above examples). The subsequent `NNNN` lines must contain the Cartesian coordinates and the weights for the grid points. See also the description of keywords `grid`, `rgrid`, and `grtol`.

Options:

<*name of the grid*> the name of the quadrature as it is specified in the `BASIS/Grid` (or `GENBAS`) file. This angular quadrature will be used in each radial point.

`LDMMMM-LDNNNN` An adaptive integration grid will be used. For each radial point, depending on its distance from the nucleus, a different Lebedev grid will be selected. The minimal and maximal number of points are `MMMM` and `NNNN`, respectively. The angular grids are selected for each radial point so that the error in the angular integrals will not be larger than $10^{1-\texttt{grtol}}$. If `grid=TA*`, the above procedure is not carried out, but the angular grids are taken from Ref. 175.

Default: `agrid=LD0006-LD0590`

Examples:

1. for a 974-point Lebedev grid set `agrid=LD0974`

2. to use an adaptive grid with at least 110 and at most 974 angular points set
   `agrid=LD0110-LD0974`
3. for a very fine grid use
   `agrid=LD0110-LD0974`
   `grtol=12`

`agrid_pssp` Specifies the angular integration grid for COSX calculations. For more information about the usage and available options, see keyword `agrid`. To learn more about COSX calculations, see keywords `pssp` and `dual_df`.

Options: Same as for `agrid`

Default: `agrid_pssp=LD0006-LD0194`

Example: for a fine grid use `agrid=LD0006-LD0590`

`agrid_pssp_sm` Specifies the small grid for dual grid COSX calculations. For more information about COSX calculations, see keywords `pssp` and `dual_df`.

Options: Same as for `agrid` and `agrid_pssp`

Default: `agrid_pssp_sm=LD0006-LD0074`

Example: for a fine grid use `agrid=LD0006-LD0194`

`basis` Specifies the basis set used in all calculations. By default the basis sets are taken from the files named by the chemical symbol of the elements, which can be found in the `BASIS` directory created at the installation. The basis sets are stored in the format used by the CFOUR package (see Sect. 6.10). In addition to the basis sets provided by default, any basis set can be used by adding it to the corresponding files in the `BASIS` directory. Alternatively, you can also specify your own basis sets in the file `GENBAS` which must be copied to the directory where MRCC is executed.

Options:

<*basis set label*> If the same basis set is used for all atoms, the label of the basis set must be given.

`atomtype` If different basis set are used, but the basis sets are identical for atoms of the same type, `basis=atomtype` should be given, and the user must specify the basis sets for each atomtype in the subsequent lines as <*atomic symbol*>:<*basis set*> .

`mixed` Mixed basis sets will be used, that is, different basis sets will be used for different groups of atoms specified by their serial number. The number of groups, the basis sets, and corresponding atoms must be specified in the subsequent lines as
<*number of groups*>
<*basis set label 1*> $< n_1 >$,$< n_2 >$,...,$< n_k >$-$< n_l >$,...
<*basis set label 2*> $< m_1 >$,$< m_2 >$,...,$< m_k >$-$< m_l >$,...
...
where $n_i$'s, $m_i$'s, ...are the serial numbers of the atoms. Serial numbers separated by dash mean that atoms $< n_k >$ through $< n_l >$ are included.

`embed` A mixed basis set composed of two AO bases will be used in the case of an embedding calculation. It only works if keyword `embed` is also specified. The two basis sets must be given in the following two lines. The first basis will be used for the environment, while the second one is the AO basis for the embedded subsystem (see also the description of keyword `embed`).

`corembed` It is the same as `embed`, but the partitioning defined by keyword `corembed` will be used.

`special` In the general case, if different basis set are used for each atom, then one should give `basis=special` and specify the basis sets for each atom in the subsequent lines by giving the label of the corresponding basis sets in the order the atoms appear at the specification of the geometry.

Notes:

1. By default the following basis sets are available for elements H to Kr in MRCC:
   - Dunning's correlation consistent basis sets [176–181]: cc-pV$X$Z, cc-pCV$X$Z, aug-cc-pV$X$Z, aug-cc-pCV$X$Z, (aug-)cc-pwCV$X$Z, cc-pV($X$+d)Z, aug-cc-pV($X$+d)Z ($X$ = D, T, Q, 5, 6)
   - Gaussian basis sets of Pople and co-workers [182–190]: STO-3G, 3-21G, 6-31G, 6-311G, 6-31G*, 6-311G*, 6-31G**, 6-311G**, 6-31+G*, 6-31+G**, 6-31++G**, 6-311+G*, 6-311+G**, 6-311++G**
   - the def2 Gaussian basis sets of Weigend and Ahlrichs [191]: def2-SV(P), def2-SVP, def2-TZVP, def2-TZVPP, def2-QZVP, def2-QZVPP
   - the augmented def2 Gaussian basis sets of Rappoport and Furche [192]: def2-SVPD, def2-TZVPD, def2-TZVPPD, def2-QZVPD, def2-QZVPPD
   - F12 basis sets for explicitly correlated wave functions developed by Peterson *et al.* [193]: cc-pV$X$Z-F12 ($X$ = D, T, Q)
   - the Gaussian basis sets of Dunning and Hay (LANL2DZ) [194]
   - the auxiliary basis sets of Weigend *et al.* for correlation calculations using the density-fitting/resolution-of-identity approximation [195, 196]: cc-pV$X$Z-RI, aug-cc-pV$X$Z-RI ($X$ = D, T, Q, 5, 6); def2-SV(P)-RI, def2-SVP-RI, def2-TZVP-RI, def2-TZVPP-RI, def2-QZVP-RI, def2-QZVPP-RI
   - the auxiliary basis sets of Hellweg and Rappoport for the augmented def2 Gaussian basis sets [197]: def2-SVPD-RI, def2-TZVPD-RI, def2-TZVPPD-RI, def2-QZVPD-RI, def2-QZVPPD-RI
   - Weigend's Coulomb/exchange auxiliary basis sets for density fitting/resolution of the identity SCF calculations [198]: cc-pV$X$Z-RI-JK, aug-cc-pV$X$Z-RI-JK ($X$ = D, T, Q, 5), def2-QZVPP-RI-JK
   - Weigend's universal Coulomb fitting auxiliary basis sets for density fitting/resolution of the identity DFT calculations [199]: def2-QZVPP-RI-J
   - the complementary auxiliary basis sets (CABSs) for explicitly correlated calculations developed by Peterson, Hill, and co-workers [200–203]: cc-pV$X$Z-F12-OPTRI ($X$ = D, T, Q), cc-pCV$X$Z-F12-OPTRI ($X$ = D, T, Q), aug-cc-pV$X$Z-OPTRI ($X$ = D, T, Q, 5), aug-cc-pwCV$X$Z-OPTRI ($X$ = D, T, Q)

   From Na to Og the following basis sets are available, which must be used together with the corresponding ECP (see also the description of keyword `ECP`):
   - the LANL2DZ basis sets of Hay and Wadt [204–206]
   - the def2 Gaussian basis sets of Weigend and Ahlrichs [191]: def2-SV(P), def2-SVP, def2-TZVP, def2-TZVPP, def2-QZVP, def2-QZVPP
   - the augmented def2 Gaussian basis sets of Rappoport and Furche [192]: def2-SVPD, def2-TZVPD, def2-TZVPPD, def2-QZVPD, def2-QZVPPD
   - the correlation consistent PP basis sets of Peterson and co-workers [207–212]: cc-pV$X$Z-PP, aug-cc-pV$X$Z-PP, (aug-)cc-pwCV$X$Z-PP ($X$ = D, T, Q, 5)
   - the auxiliary basis sets of Hellweg and Rappoport for the augmented def2 Gaussian basis sets [197]: def2-SVPD-RI, def2-TZVPD-RI, def2-TZVPPD-RI, def2-QZVPD-RI, def2-QZVPPD-RI
   - the auxiliary basis sets of Hättig and Hill for correlation calculations with the core-valence and PP basis sets [213, 214]: aug-cc-pwCV$X$Z-RI, cc-pV$X$Z-PP-RI and aug-cc-pV$X$Z-PP-RI, (aug-)cc-pwCV$X$Z-PP-RI ($X$ = D, T, Q, 5)
   - the auxiliary basis sets of Kritikou and Hill for explicitly correlated calculations [215]: cc-pCV$X$Z-F12-RI ($X$ = D, T, Q)

   Please note that some of the above basis sets are not available for all elements.

2. If you need basis sets other than the default ones, you can, e.g., download them from the Basis Set Exchange [167–170] or from ccRepo [216]. Please choose format “`CFOUR`” when downloading the basis sets.
3. If you use your own basis sets, these must be copied to the end of the corresponding file in the `BASIS` directory. Alternatively, you can also create a file called `GENBAS` in the directory where Mrcc is executed, and then you should copy your basis sets to that file.
4. The labels of the basis sets must be identical to those used in the `BASIS/*` files (or the `GENBAS` file). For the default basis sets just type the usual name of the basis set as given above, e.g., `cc-pVDZ`, `6-311++G**`, etc. If you employ non-default basis sets, you can use any label.
5. For Dunnings's aug-cc-p(C)V$X$Z basis sets one, two, or three additional diffuse function sets can be automatically added by attaching the prefix `d-`, `t-`, or `q-`, respectively, to the name of the basis set. To generate a d-aug basis set one even tempered diffuse function is added to each primitive set. Its exponent is calculated by multiplying the exponent of the most diffuse function by the ratio of the exponents of the most diffuse and the second most diffuse functions in the primitive set. If there is only one function in the set, the exponent of the most diffuse function is divided by 2.5. To generate t-aug and q-aug sets this procedure is repeated.
6. For Dunnings's basis sets, to use the aug-cc-p(C)V$X$Z set for the non-hydrogen atoms and the corresponding cc-p(C)V$X$Z set for the hydrogens give `aug'-cc-p(C)VXZ`. Then the diffuse functions will be automatically removed from the hydrogen atoms.

7. Only the conventional AO basis set can be specified with this keyword. For the auxiliary basis sets used in density-fitting or F12 approximations see the description of keywords `dfbasis_*`.
8. The cc-pVDZ-RI-JK basis set has been generated from cc-pVTZ-RI-JK by dropping the functions of highest angular momentum. The aug-cc-pV$X$Z-RI-JK (def2-QZVPPD-RI-JK) basis sets are constructed automatically from the corresponding cc-pV$X$Z-RI-JK (def2-QZVPP-RI-JK) sets by adding diffuse functions as described above for the d-aug-cc-p(C)V$X$Z basis sets.
9. For Dunnings's and Pople's basis sets add the `-min` postfix to the basis set name to generate a minimal basis set dropping all the polarization (correlation) functions.
10. If the (aug-)cc-pV$X$Z-PP basis set does not exist for an element with $Z \leq 28$, the program will automatically attempt to use the corresponding (aug-)cc-pV$X$Z basis instead.
11. If the (aug-)cc-pV($X$+d)Z basis set does not exist for an element (i.e., $Z \leq 12$ or $Z \geq 37$ or $Z = 19$), the program will automatically attempt to use the corresponding (aug-)cc-pV$X$Z basis instead.

Default: none, that is, the basis set must be specified (excepting the case when Mrcc is used together with another code, that is, `iface` $\neq$ `none`).

Examples:

1. Consider any molecule and suppose that the cc-pVDZ basis set is used for all atoms. The input must include the following line:
```
basis=cc-pVDZ
```
2. To use Dunning's doubly augmented cc-pVDZ basis set (d-aug-cc-pVDZ) for all atoms the input must include the following line:
```
basis=d-aug-cc-pVDZ
```
3. Consider the water molecule and use the cc-pVDZ basis set for the hydrogens and cc-pVTZ for the oxygen. The input must include the following lines:
```
basis=atomtype
O:cc-pVTZ
H:cc-pVDZ
```
4. Consider water again and use the cc-pVQZ, cc-pVTZ, and cc-pVDZ basis sets for the oxygen atom, for the first hydrogen, and for the second hydrogen, respectively. Note that the order of the basis set labels after the `basis=special` statement must be identical to the order of the corresponding atoms in the Z-matrix/Cartesian coordinates:
```
geom
O
H 1 R
H 1 R 2 A

R=0.9575
A=104.51
```

```
basis=special
cc-pVQZ
cc-pVTZ
cc-pVDZ
```

5. Consider the water molecule and use the cc-pVTZ basis set for the hydrogens and aug-cc-pVTZ for the oxygen. The following two inputs are identical:
   ```
   basis=atomtype
   O:aug-cc-pVTZ
   H:cc-pVTZ
   ```
   or
   ```
   basis=aug'-cc-pVTZ
   ```
6. Consider the water molecule. If you specify
   ```
   basis=cc-pVTZ-min
   ```
   minimal basis sets generated from cc-pVTZ will be used for the atoms, that is, only one *s* function (two *s* and one *p* shells) will be retained from the *s*–*p* kernel of the H (O) cc-pVTZ basis set.
7. Consider the PbO molecule. If you want to use the cc-pVDZ basis set for O and the cc-pVDZ-PP basis with the corresponding ECP for Pb, you only need to set
   ```
   basis=cc-pVDZ-PP
   ```
   in the `MINP` file.
8. Mixed basis approach with two basis sets, the cc-pVTZ basis is used for atoms 1, 2, 3, and 5, while cc-pVDZ is employed for atoms 4, 6, 7, 8:
   ```
   basis=mixed
   2
   cc-pVTZ 1-3,5
   cc-pVDZ 4,6-8
   ```

`basis_sm` Specifies the small basis set used in dual basis-set calculations as well as for generating SCF initial guess (`scfiguess=small`).

Options: the options are the same as for keyword `basis`, but there is an additional one, `none`, which means that no small basis is defined.

Default: `basis_sm=none`

Examples:

1. To restart an SCF calculation with the cc-pVQZ basis set from the densities obtained with the cc-pVDZ basis give
   ```
   basis=cc-pVQZ
   basis_sm=cc-pVDZ
   scfiguess=small
   ```
2. To perform a dual basis set DF-HF calculation with the 6-311G** and 6-31G** basis sets you need:
   ```
   basis=6-311G**
   basis_sm=6-31G**
   dual=on
   calc=DF-HF
   ```

`basopt` Use this keyword to turn on/off basis set optimization. Besides setting this keyword a user supplied `GENBAS` file is also required for basis set optimization jobs. It is also possible to set the value of `basopt` to be equal to an appropriate energy. In this case the basis set parameters are optimized so that the absolute value of the difference between this value and the actual energy is minimized. This option comes handy when optimizing a density fitting basis set. In this case the difference between the actual and non-density-fitting energy (obtained from a previous calculation) will be minimized. See also Sect. 6.10.

Options: `on`, `off`, or <*any real number*>

Default: `basopt=off`

Examples:

1. To optimize a basis set variationally set `basopt=on`
2. To optimize a basis set minimizing the difference of the calculated energy and -76.287041 $E_h$ set `basopt=-76.287041`

`bfbasis` Specifies the bond function (BF) basis (see Ref. 27 for details).

Options:

`none` No BFs are used.

<*BF basis name*> name of the BF basis to be used.

Notes:

1. The format of the name of the BF basis, <*BF basis name*>, is <*AO basis name*>-<*BF type*>. E.g., 6-31G-1s1p is a BF basis optimized for the 6-31G AO basis and one s and one p function set are placed on the corresponding bonds.
2. The BF basis sets are stored in the `BASIS/Bond` file but the BF basis can also be specified in the `GENBAS` file similar to the AO basis sets (see the description of keyword `basis`). The format of the label of the BF basis in the file is `B`<*bond name*>`:`<*BF basis name*>. E.g., `BCH:6-31G-1s1p` is 6-31G-1s1p BF basis optimized for the C–H bond.
3. If BF bases are used, the geometry must be given in `mol` format (see the description of keyword `geom`)

Default: `bfbasis=none`

Example: hydrogen-fluoride molecule, the 6-31G basis and the 6-31G-1s1p bond function basis are used:

```
basis=6-31G
bfbasis=6-31G-1s1p
geom=mol
2 1
0.00000000 0.00000000 0.00000000 F
0.00000000 0.00000000 0.91690000 H
1 2 1
```

`bfgsmem` Specifies the number of gradient vectors used for the BFGS update in quadratic SCF calculations using the BFGS algorithm (`qscf=BFGS`).

Options: <*any positive integer*>

Default: `bfgsmem=10`

Example: to increase the number of vectors to 15 set `bfgsmem=15`

`bfgstol` Threshold (in $E_h$) for starting the BFGS algorithm in quadratic SCF calculations using the BFGS algorithm (`qscf=BFGS`). The calculation starts with the conventional DIIS-based convergence acceleration, and the BFGS algorithm will be switched on if the maximum norm of the gradient is smaller than `bfgstol`.

Options: <*any positive real number*>

Default: `bfgstol=1e-3`

Example: for a convergence threshold of $10^{-4}$ $E_h$ set `bfgstol=1e-4`

`boysalg` Specifies the optimization algorithm used for the Boys–Foster localization.

Options:

`Jacobi` Uses standard Jacobi rotations.

`Newton` Uses trust-region optimization technique.

Default: `boysalg=Jacobi`

Example: for the trust-region optimization set `boysalg=Newton`

`bpcompo` Boughton–Pulay completeness criterion [217] for occupied orbitals. In various local correlation approaches the Boughton–Pulay procedure is used to identify the atoms on which an LMO is localized. The least-squares residual of the parent LMO and the LMO truncated to the selected atoms is required to be less than one minus this criterion.

Options:

<*any real number in the [0,1] interval*> This number will be used as the completeness criterion.

Default: `bpcompo=0.95` for local excited-state calculations, `bpcompo=0.985` otherwise

Note: Atom domains determined by `bpcompo` are also utilized to construct local fitting domains in the case of `localcc`$\geq$`2016` according to Ref. 31.

Example: to set a threshold of 0.99 type `bpcompo=0.99`

`bpcompv` Boughton–Pulay completeness criterion [217] for virtual orbitals (projected atomic orbitals). See also keyword `bpcompo`.

Options:

<*any real number in the [0,1] interval*> This number will be used as the completeness criterion.

Default: `bpcompv=0.98`

Example: to set a threshold of 0.95 type `bpcompv=0.95`

`bpedo` Boughton–Pulay completeness criterion [217] for the occupied orbitals of an extended domain. See also keyword `bpcompo`.

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `bpedo=0.9999` is set as default in the case of `localcc`≥`2018` and `lcorthr=normal` for both LMP2 and LNO-CC computations, according to Refs. 31 and 38. See the description of `lcorthr` for further details on the predefined values of `bpedo` for other cases

Note: `bpedo=bpcompo` is set if `bpedo` is not specified and not employed in the local correlation calculation

Example: to set a threshold of 0.9998 type `bpedo=0.9998`

`bpedv` Boughton–Pulay completeness criterion [217] for the virtual orbitals (projected atomic orbitals) of an extended domain. See also keyword `bpcompo`.

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `bpedv=0.995` is set as default in the case of `localcc`≥`2016` according to Ref. 31.

Note: `bpedv=bpcompv` is set if `bpedv` is not specified and not employed in the local correlation calculation

Example: to set a threshold of 0.99 type `bpedv=0.99`

`bp_subsyso` Boughton–Pulay (BP) completeness criterion [217] for the occupied orbitals in the case of subsystem-based calculations, where the system is divided into environment and active parts. The keyword controls the number of atoms that are assigned to the occupied molecular orbitals. Note that the keyword affects numerical results only when the `bopu` selection scheme is utilized for the embedding methods (see the `embed` and `corembed` keywords), or the border atoms are automatically handled in the `oniom` approach. See also keywords `bpcompo` and `subsys_bopu`.

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `bp_subsyso=0.95`

Example: to set a threshold of 0.90, type `bp_subsyso=0.90`

`bp_subsysv` Boughton–Pulay (BP) completeness criterion [217] for the virtual orbitals in the case of subsystem-based calculations, where the system is divided into environment and active parts. The keyword controls the number of atoms that are assigned to the virtual molecular orbitals. Note that the keyword affects numerical results only when the `bopu` selection scheme is utilized for the embedding methods, where virtual space is also partitioned. See also keywords `bpcompv`, `embed`, and `subsys_bopu`.

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `bp_subsysv=0.95`

Example: to set a threshold of 0.90, type `bp_subsysv=0.90`

`cabscorr` Specifies whether the CABS correction, in addition to DBBSC (see keyword `denscorr`), is calculated.

Options:

`off` The CABS correction is not calculated.

`on` The CABS correction is calculated.

`exit` The F12-based calculations stop after the CABS correction.

Default: `cabscorr=on` if `denscorr=1` or 2, otherwise `cabscorr=off`.

Example: to calculate the CABS correction, the user should set `cabscorr=on`

`bppdo` Boughton–Pulay completeness criterion [217] for the occupied orbital of a primary domain. See also keyword `bpcompo`.

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `bppdo=0.999` is set as default in the case of `localcc`≥`2016` according to Ref. 31.

Note: `bppdo=bpcompo` is set if `bppdo` is not specified and not employed in the local correlation calculation

Example: to set a threshold of 0.99 type `bppdo=0.99`

`bppdv` Boughton–Pulay completeness criterion [217] for virtual orbitals (projected atomic orbitals) of a primary domain. See also keywords `bppdo` and `bpcompo`.

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `bppdv=bpcompv`

Example: to set a threshold of 0.99 type `bppdv=0.99`

`calc` Specifies the type of the calculation.

Options:

`SCF`, `HF`, or `KS`

Hartree–Fock or Kohn–Sham SCF calculation, the type of the calculation can be controlled by keyword `scftype` (see also keyword `scftype`).

`RHF`, `UHF`, `ROHF`, `RKS`, `UKS`, `ROKS`, `MCSCF`

Restricted, unrestricted, restricted open-shell Hartree–Fock SCF; restricted, unrestricted, restricted open-shell Kohn–Sham SCF; or multi-configurational SCF calculation, respectively. The type of the HF/KS/MCSCF procedure is also defined at the same time if these options are chosen, and it is not necessary to set `scftype`. That is, `calc=RHF` is equivalent to `calc=SCF` plus `scftype=RHF`, etc.

`B3LYP, PBE0, B3PW91, B3LYP-D3, B2PLYP-D3, ...`

Kohn–Sham SCF calculation with the specified density functional. The type of the Kohn–Sham procedure (i.e., RKS, UKS, or ROKS) can be controlled by keyword `scftype` (see also keyword `scftype`). The options are identical to those of keyword `dft` (except for `off`, `user`, `userd`, and `rsdh`), see the description of keyword `dft`. Note that for a correlated calculation with KS orbitals you can only select the functional with keyword `dft`, the value of keyword `calc` must be set to the desired correlation method. Note also that for DFT calculations the density fitting approximation is used by default, i.e., `dfbasis_scf` is set to `auto`. To run a conventional KS calculation set `dfbasis_scf=none`.

`TDHF`

Time-dependent HF (TD-HF, also known as random-phase approximation). If `calc=SCF` and number of the states is greater than one (set by keywords `nsing`, `ntrip`, or `nstate`), also TD-HF calculations are performed for the excited states. It is only available with density fitting.

`TDDFT`

Full time-dependent DFT (TD-DFT). The density functional must be set using keyword `dft`. Alternatively, if `calc` is set to the name of the functional, and the number of the states is greater than one (set by keywords `nsing`, `ntrip`, or `nstate`), also TD-DFT calculations are performed for the excited states using the given functional. For HF reference it is equivalent to TD-HF. It is only available with density fitting. For excited-state calculations with double hybrid functionals see also keyword `dhexc`.

`TDA`

TD-DFT in the Tamm–Dancoff approximation (TDA). For HF reference it is equivalent to CIS. It is only available with density fitting. For excited-state calculations with double hybrid functionals see also keyword `dhexc`.

`MP2`

Second-order Møller–Plesset (MP2) calculation. Note that efficient MP2 calculations are only possible with the density-fitting (resolution-of-identity) approximation, which can be performed with the `DF-MP2` or the `RI-MP2` keywords or by setting `dfbasis_cor≠none`. In the case of a DF-MP2 calculation, the spin-component scaled MP2 (SCS-MP2) [109] and the scaled opposite-spin MP2 (SOS-MP2) [110] energy will also be computed (see also keywords `scsps` and `scspt`).

`MP3`

Third-order Møller–Plesset (MP3) calculation [46]. Note that MP3 calculations without the density-fitting (resolution-of-identity) approximation are not possible with RHF reference wave functions.

`SOS-MP2`

Scaled opposite-spin second-order Møller–Plesset (SOS-MP2) calculation [110] using an $N^4$-scaling algorithm based on the Cholesky decomposition/Laplace transform of energy denominators (in practice one dRPA iteration is performed, see below). Note that it is only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-SOS-MP2 (≡ RI-SOS-MP2) calculation is performed (that is, options `SOS-MP2`, `DF-SOS-MP2`, and `RI-SOS-MP2` are synonyms).

`SCS-MP2`

For canonical calculations it is equivalent to option `MP2`. If a local correlation calculation is executed, only the spin-component scaled MP2 (SCS-MP2) energy will be computed.

`dRPA`

Direct random-phase approximation (dRPA) calculation (see Eqs. 7 and 8 in Ref. 218). Note that dRPA calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-dRPA (≡ RI-dRPA) calculation is performed (that is, options `dRPA`, `DF-dRPA`, and `RI-dRPA` are synonyms).

`RPA`

Random-phase approximation (RPA) calculation (see Eqs. 10 and 13 in Ref. 218, where it is referred to as RPAx-SO2). Note that RPA calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-RPA (≡ RI-RPA) calculation is performed (that is, options `RPA`, `DF-RPA`, and `RI-RPA` are synonyms).

`SOSEX`

Second-order screened exchange (SOSEX) [114] calculation (see Eqs. 7 and 9 in Ref. 218), the dRPA and the rPT2 [115] energies are also computed. Note that SOSEX calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-SOSEX (≡ RI-SOSEX) calculation is performed (that is, options `SOSEX`, `DF-SOSEX`, and `RI-SOSEX` are synonyms).

`sedRPA, seSOSEX, dsdRPA, dsSOSEX`

The scaled-equation and down-scaled dRPA and SOSEX methods of Ref. 44. See also the notes for options `dRPA` and `SOSEX` as well as the descriptions of keywords `scspe` and `scsph`.

`RPAX2`

RPAX2 calculation (see Eqs. 17 to 19 in Ref. 117). Note that RPAX2 calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-RPAX2 (≡ RI-RPAX2) calculation is performed (that is, options `RPAX2`, `DF-RPAX2`, and `RI-RPAX2` are synonyms).

`CIS`

Configuration interaction singles (CIS) calculation [40]. Efficient CIS calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-CIS (≡ RI-CIS) calculation is performed (that is, options `CIS`, `DF-CIS`, and `RI-CIS` are synonyms). If you are still interested in the CIS energy without DF, set `ccprog=mrcc`, `dfbasis_scf=none`, and `dfbasis_cor=none`.

`CIS(D)`

Configuration interaction singles with perturbative correction for double excitations [CIS(D)] calculation [41, 149]. Note that CIS(D) calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-CIS(D) [≡ RI-CIS(D)] calculation is performed [that is, options `CIS(D)`, `DF-CIS(D)`, and `RI-CIS(D)` are synonyms].

`CIS(Di)`

Iterative doubles correction to configuration interaction singles [CIS($D_\infty$)] calculation [35, 37, 149]. Note that CIS($D_\infty$) calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-CIS($D_\infty$) [≡ RI-CIS($D_\infty$)] calculation is performed [that is, options `CIS(Di)`, `DF-CIS(Di)`, and `RI-CIS(Di)` are synonyms].

`ADC(2)`

Second-order algebraic diagrammatic construction [ADC(2)] calculation [35, 37, 153, 219]. Note that ADC(2) calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-ADC(2) [≡ RI-ADC(2)] calculation is performed [that is, options `ADC(2)`, `DF-ADC(2)`, and `RI-ADC(2)` are synonyms].

`CC2`

Second-order coupled-cluster singles and doubles (CC2) calculation [35, 37, 118, 219]. Efficient CC2 calculations are only possible with the density-fitting (resolution-of-identity) approximation, and, by default, a DF-CC2 (≡ RI-CC2) calculation is performed (that is, options `CC2`, `DF-CC2`, and `RI-CC2` are synonyms). If you are still interested in the CC2 energy without DF, set `ccprog=mrcc`, `dfbasis_scf=none`, and `dfbasis_cor=none`.

`SOS-CC2, SOS-CIS(D), SOS-CIS(Di), SOS-ADC(2)`

Scaled opposite-spin CC2, CIS(D), CIS($D_\infty$), and ADC(2) [SOS-CC2, SOS-CIS(D), SOS-CIS($D_\infty$), SOS-ADC(2)] calculation [35, 37, 41, 118, 120]. An $N^4$-scaling algorithm based on the Cholesky decomposition/Laplace transform of energy denominators is executed. It is only available with density fitting. Note that two different parameterizations are implemented for SOS-CIS(D) (see Refs. 119 and 151). See also keywords `scsps`, `scspt`, `scsps_t`, and `scspt_t`.

`SCS-CC2, SCS-CIS(D), SCS-CIS(Di), SCS-ADC(2)`

Spin-component scaled CC2, CIS(D), CIS($D_\infty$), and ADC(2) [SCS-CC2, SCS-CIS(D), SCS-CIS($D_\infty$), SCS-ADC(2)] calculation [35, 37, 41, 118, 120]. It is only available with density fitting. Note that two different parameterizations are implemented for SCS-CIS(D) (see Refs. 119 and 151). See also keywords `scsps`, `scspt`, `scsps_t`, and `scspt_t`.

`CCS, CCSD, CCSDT, CCSDTQ, CCSDTQP, CC(<n>)`

The corresponding single-reference CC calculation if the number of active orbitals is zero (see Ref. 3); the corresponding SRMRCCSD, SRMRCCSDT, etc.

calculation otherwise (see Ref. 4).

`CCSD[T], CCSDT[Q], CCSDTQ[P], CC(<n-1>)[<n>]`

The corresponding single-reference CC calculation with perturbative corrections (see Ref. 9).

`CCSD(T), CCSDT(Q), CCSDTQ(P), CC(<n-1>)(<n>)`

The corresponding single-reference CC calculation with perturbative corrections (see Ref. 9).

`CCSD(T)_L, CCSDT(Q)_L, CCSDTQ(P)_L, CC(<n-1>)(<n>)_L`

The corresponding $\mathrm{CCSD(T)}_\Lambda$, $\mathrm{CCSDT(Q)}_\Lambda$, etc. calculation (see Ref. 9).

`CCSDT-1a, CCSDTQ-1a, CCSDTQP-1a, CC(<n>)-1a`

The corresponding iterative approximate single-reference CC calculation (see Ref. 9).

`CCSDT-1b, CCSDTQ-1b, CCSDTQP-1b, CC(<n>)-1b`

The corresponding iterative approximate single-reference CC calculation (see Ref. 9).

`CC2, CC3, CC4, CC5, CC<n>`

The corresponding iterative approximate single-reference CC calculation (see Ref. 9).

`CCSDT-3, CCSDTQ-3, CCSDTQP-3, CC(<n>)-3`

The corresponding iterative approximate single-reference CC calculation (see Ref. 9).

`CCSDT[Q]/A, CCSDTQ[P]/A, CC(<n-1>)[<n>]/A`

The corresponding single-reference CC calculation with perturbative corrections using ansatz A (see Ref. 15).

`CCSDT[Q]/B, CCSDTQ[P]/B, CC(<n-1>)[<n>]/B`

The corresponding single-reference CC calculation with perturbative corrections using ansatz B (see Ref. 15).

`CCSDT(Q)/A, CCSDTQ(P)/A, CC(<n-1>)(<n>)/A`

The corresponding single-reference CC calculation with perturbative corrections using ansatz A (see Ref. 15).

`CCSDT(Q)/B, CCSDTQ(P)/B, CC(<n-1>)(<n>)/B`

The corresponding single-reference CC calculation with perturbative corrections using ansatz B (see Ref. 15).

`CIS, CISD, CISDT, CISDTQ, CISDTQP, CI(<n>), FCI`

The corresponding single-reference CI calculation if the number of active orbitals is zero (see Ref. 3), the corresponding MRCISD, MRCISDT, etc. calculation otherwise (see Ref. 4).

`MP2-F12`

The explicitly correlated MP2 (MP2-F12) method using the 2B ansatz, the F + K commutator approximation, fixed amplitudes, spin-flipped geminals in open-shell calculations [53, 111, 112]. It is only available with DF. We note that

MP2-F12-based DH-DFT-F12 [220] calculations are also available (see keyword `dft`).

`CCSD-F12` or `CCSD(F12*)`, `CCSD(T)-F12` or `CCSD(F12*)(T)`

The explicitly correlated CCSD and CCSD(T) utilizing the CCSD(F12*) [133] and (T+) [53] models and the above MP2-F12 approximation. It is only available with DF.

Notes:

1. In the above options, $n$ is a positive integer, which is the excitation level of the highest excitation. $n$ is supposed to be equal to or greater than 6 since for smaller $n$'s the CC($<n>$) and similar options are equivalent to one of the other options, e.g., CC(5) is equivalent to CCSDTQP or CC(3)(4) is identical with CCSDT(Q).
2. For excited-state calculations with the TD-HF, TDA, TD-DFT (including double hybrid approaches), CIS, CIS(D), CIS($D_\infty$), ADC(2), CC2 and various CC and CI methods the number of states should be greater than one (keywords `nsing`, `ntrip`, or `nstate`). If more than one state is requested for CC calculations, the corresponding linear-response (LR) CC (for excitation energies it is equivalent to equation-of-motion CC, EOM-CC) calculation is performed automatically for the excited states. If more than one state is requested and `calc=SCF`, TD-HF (`dft=off`) or TD-DFT (`dft`$\neq$`off`) calculations will be carried out for the excited states.
3. The active orbitals can be selected and the MRCI/CC calculations can be controlled by keywords `nacto`, `nactv`, `active`, `maxex`, and `maxact`. Note that, by default, MRCI/CC calculations are executed using HF reference. To use MCSCF orbitals `scftype` must be set to `MCSCF`, and the MCSCF wave function must be defined by keywords `docc` and `mact`. In this case `nacto` and `nactv` are taken over from the MCSCF calculation, and a MRCI/CC calculation will be run automatically. You should only set the above keywords if you want to run the post-MCSCF calculation with an active space different from the MCSCF one.
4. In principle, all methods can be used with the density fitting (resolution-of-identity) approximation. It is possible in two ways. You can attach the prefix `DF-` or `RI-` to the corresponding option from the above list. Then, for a HF calculation keyword `dfbasis_scf` will be set to `auto`, while for a correlated calculation both `dfbasis_scf` and `dfbasis_cor` will be given the value `auto`. Alternatively, you can also set the values for keywords `dfbasis_scf` and `dfbasis_cor`, see their description.
5. Local correlation methods, for both ground and excited states, can be run if the prefix "`L`" is added to the corresponding option of the keyword, e.g., as `LMP2`, `LdRPA`, `LCCSD(T)`, `LADC(2)`, `LCC2`, etc. Additionally, the prefix "`LNO-`" can also be used as a synonym in the case of local coupled-cluster approaches, e.g., as `LNO-CCSD`, `LNO-CCSD(T)`, `LNO-CCSDT`, etc., and for ADC(2) [`LNO-ADC(2)`]. Both options are equivalent to setting `localcc=on`.
6. For the dRPA, RPA, and SOSEX methods the use of PBE orbitals is recom-

mended (this is set by default).

7. For the RPAX2 method the use of PBEx orbitals is recommended (this is set by default).
8. If `FNO-CCSD` or `FNO-CCSD(T)` is set, a CCSD or CCSD(T) calculation is performed utilizing the FNO and NAF approximations [48] and the PPL+ and (T+) corrections [55]. In these cases, `ovirt=ppl`, `naf_cor=on`, and `dfbasis_cor=auto` are set by default. `CCSD-PPL+` and `CCSD-PPL+(T+)` are equivalent to `FNO-CCSD` and `FNO-CCSD(T)`.
9. If the `FNO-` prefix is added to the acronym of an explicitly correlated CC method, that is, if `FNO-CCSD-F12`, `FNO-CCSD(T)-F12`, `FNO-CCSD(F12*)(T+)`, etc. is set, the corresponding calculation is performed utilizing the FNO, NAF, and NAB approximations [60]. In these cases, `ovirt=mp2`, `naf_cor=on`, `naf_f12=on`, `nab=on`, and `dfbasis_cor=auto` are set by default.

Default: `calc=SCF`

Examples:

1. To run a CCSD(T) calculation, the user should set `calc=CCSD(T)`
2. For DF-HF (RI-HF) calculations type:
   `calc=DF-HF`
   which is equivalent to the following input:
   `calc=SCF`
   `dfbasis_scf=auto`
3. For a local CCSD(T) calculation using the local natural orbital approximation set `calc=LCCSD(T)` or `calc=LNO-CCSD(T)`
4. For a RI-MP2 calculation set `calc=MP2`
5. For a DFT calculation with the B3LYP functional set `calc=B3LYP`
6. Direct RPA calculation with Kohn–Sham orbitals calculated with the PBE functional:
   `calc=dRPA`
   `dft=PBE`
7. TD-DFT calculation for the 3 lowest singlet excited states of a molecule using the PBE functional:
   `calc=TDDFT`
   `dft=PBE`
   `nsing=4`
   A somewhat less complicated input for the same purpose:
   `calc=PBE`
   `nsing=4`

`ccmaxit` Maximum number of iteration steps in correlated calculations (CC, CI, RPA, . . . ).

Options: <*any positive integer*>

Default: `ccmaxit=50`

Example: to increase the maximum number of CC iterations to 100 give `ccmaxit=100` `ccprog` Specifies the CC program to be used.

Options:

`mrcc` The automated, string-based CC program `mrcc` will be called.

`ccsd` The very fast, hand-coded CCSD(T) codes, `ccsd` or `uccsd`, will be executed (currently the spatial symmetry cannot be utilized).

`cis` The very fast, hand-coded, integral direct DF-CIS code `cis` will be executed (currently the spatial symmetry cannot be utilized).

Note: Please note that the `mrcc` code was optimized for high-order CC calculations, such as CCSDT(Q) and CCSDTQ, which require different algorithms than CCSD(T). Thus it is slow for CCSD(T), but optimal for high-order CC models.

Default: `ccprog=ccsd` for conventional and explicitly correlated CCSD and CCSD(T) calculations, `ccprog=cis` for CIS, CIS(D), CIS($D_\infty$), ADC(2), and CC2 calculations, `ccprog=mrcc` otherwise.

Example: to use the `mrcc` code for CCSD or CCSD(T) calculations give `ccprog=mrcc`

`ccsdalg` Specifies the CCSD algorithm employed in the `ccsd` or `uccsd` program if `ccprog=ccsd`.

Options:

`disk` An MO-based, hand-coded, OpenMP-parallel CCSD algorithm is invoked [24]. All the necessary four-center integrals (including the ones with four virtual orbitals) are stored on disk after the integral transformation (and integral assembly) steps. The four-center integrals are read in each CCSD iteration from these files. Careful optimization was performed significantly improving both the CPU usage and the OpenMP parallelization efficiency of the implementation (see Ref. 46 for `ccsd`).

`dfdirect` For `ccsd`: A highly-optimized, hand-coded, OpenMP-parallel, extremely low-memory, in-core, integral-direct $t_1$-transformed DF-CCSD algorithm is invoked [46]. The necessary four-center integrals and intermediates are constructed completely in memory in an integral direct manner in each iteration, and disk I/O is avoided. This implementation, including the cost of the repeated integral assembly, is still more efficient than the optimized `ccsdalg=disk` one, especially if the OpenMP parallelization is employed. For `uccsd`: currently the most storage-intensive four-external four-center integrals are assembled on-the-fly from the corresponding two- and three-center DF integrals in each CCSD iteration. The remaining four-center integrals are stored on disk.

Note: The `dfdirect` algorithm is currently only available with the `ccsd` or `uccsd` programs, i.e., via `ccprog=ccsd`.

Default: `ccsdalg=dfdirect` if density fitting is applied for the correlation energy calculation, e.g., for `calc=DF-CCSD`, `calc=DF-CCSD(T)`, `calc=LNO-CCSD(T)`, etc. If density fitting is not employed, i.e., `dfbasis_cor=none`, then `ccprog=disk` is selected automatically.

Example: `ccsdalg=disk` switches to the CCSD algorithm that stores the four-center integrals on disk.

`ccsdmkl` Controls the parallel execution of Intel MKL subroutines in program `ccsd`.

Options:

- `thr` Threaded/parallel version of MKL is used
- `seq` Sequential MKL procedures are called in parallel

Default: `ccsdmkl=seq`

Example: to run BLAS subroutines on more than 1 thread in program `ccsd` set `ccsdmkl=thr`

`ccsdrest` Use this keyword to restart canonical or reduced-cost (i.e., not local) CC calculations from previously calculated amplitudes if `ccprog=ccsd` and `talg=occ`. For restarting local correlation calculations, see keyword `lccrest`. For restarting canonical CI or CC calculations with `ccprog=mrcc`, see keyword `rest`.

Options:

- `off` Turns off checkpointing and performs the CC calculation normally
- `trf` Turns on checkpointing creating the `ccsd.rst` and `pt.rst` files and performs the CC calculation normally
- `ccsd` Turns on checkpointing and restarts the CC calculation from the last completed iteration for CCSD or the next unfinished $ijk$ index triplet for the (T) correction

Note: In the case of restarting a CC calculation, program `ccsd` is run directly skipping the SCF procedure and the integral transformation. Therefore, the `VARS`, `55`, and `DFINT_*` files, as well as the checkpoint files `ccsd.rst` and `pt.rst` from the previous run need to be present in the working directory.

Default: `ccsdrest=trf`

Example: to restart a CCSD/CCSD(T) calculation set `ccsdrest=ccsd`

`ccsdthreads` Sets the number of outer OpenMP threads in program `ccsd` while performing the CCSD iteration.

Options: <*any positive integer*>

Default: `ccsdthreads=2`

Example: to reduce the memory requirement of a CCSD calculation by turning off nested OpenMP set `ccsdthreads=1`

`cctol` Convergence threshold for the energy in correlated calculations (CC, CI, dRPA, RPA, etc.). The energy will be accurate to $10^{-\texttt{cctol}}$ $\mathrm{E}_h$.

Options: <*any integer*>

Default: `cctol=8` for property calculations, `cctol=`[-$\log_{10}$(`optetol`)]+2 for geometry optimizations, `cctol=5` for `localcc`≥`2016`, `cctol=6` otherwise

Example: for an accuracy of $10^{-8}$ $\mathrm{E}_h$ one must give `cctol=8`

`charge` Charge of the system.

Options: <*any integer*>

Default: `charge=0`

Example: for the $\mathrm{Cl}^-$ ion one should give `charge=-1`

`cialg` Specifies what type of algorithm is to be used in CIS, TDA, TD-HF, and TD-DFT calculations.

Options:

- `disk` Conventional algorithm, two-electron integrals are stored on disk
- `direct` Completely I/O-free, integral-direct algorithm, two-electron integrals are recalculated in each iteration step.
- `direct2` Partially I/O-free, integral-direct algorithm; recommended if the I/O is fast and/or few states are required.
- `direct3` Variant of `direct2`, but usually slower.
- `auto` Based on the size of the molecule the program will automatically select the most efficient one from the above options.

Note: The `cialg=direct` option can be used for iterative second-order methods, such as CC2, ADC(2), and CIS($D_\infty$). In this case, a partially integral-direct algorithm is executed.

Default: `cialg=auto`

Example: to use disk-based algorithm set `cialg=disk`

`ciguess` The initial guess vectors for CI and LR-CC calculations can be specified using this keyword.

Options:

- `on` The initial trial vectors are supplied by the user and should be given in the subsequent lines as follows. For each state the corresponding initial guess vector must given by the number of non-zero elements of the vector on the first line, followed by as many lines as the number of non-zero elements. In each line the corresponding excitation operator and the value for this element of the vector must be provided in the following format:
  $< n >< sp_1 >< sp_2 > \ldots < sp_n >< a_1 >< a_2 > \ldots < a_n >< i_1 >< i_2 > \ldots < i_n >< coeff >$
  where $< n >$ is the level of excitation, and the electrons are promoted from occupied orbitals $< i_1 >< i_2 > \ldots < i_n >$ to virtual orbitals $< a_1 >< a_2 > \ldots < a_n >$ with spins $< sp_1 >< sp_2 > \ldots < sp_n >$ ($< sp_k >$ is `1` for alpha and `0` for beta), respectively. $< coeff >$ is the corresponding coefficient.
- `off` Initial trial vectors are not specified, the program applies simple unit vectors as initial guess. The unit vectors are determined on the basis of the diagonal elements of the Hamiltonian: if $n$ roots are requested, $n$ unit vectors corresponding to the $n$ lowest diagonals will be used.

Default: `ciguess=off`

Example: Suppose that we have two excited states in a LR-CC calculation. Then the initial guess can be given as follows.

```
ciguess=on
1
1 1 6 4 1.0
3
1 1 7 3 0.1
2 1 0 7 7 5 5 1.0
2 1 1 7 6 3 4 0.1
```

For the first state there is only one entry, a single excitation of the alpha electron from orbital 4 to orbital 6 with a coefficient of 1.0. For the second root the initial guess vector contains three entries. A single excitation from orbital 3 to orbital 7 with alpha spin and a relative weight of 0.1, a double excitation from orbital 5 to orbital 7 with a weight of 1.0, and another double excitation of the alpha electrons from orbitals 3 and 4 to orbitals 6 and 7 with a weight of 0.1.

Notes:

1. For $M_S = 0$ states the vector is automatically spin-adapted, and you do not need to specify the coefficients for the corresponding spin-reversed excitations. E.g., in the above example, for root 1 the `1 0 6 4 1.0` entry is unnecessary.
2. The guess vector is not required to be normalized, it is done automatically.
3. In the case of four-component relativistic calculations (DIRAC interface) the serial numbers of the spinors should be specified. In addition, the second number in the above strings must be 1 (that is, all excitations are formally considered as excitations of alpha electrons).

`cmpgrp` Specifies the computational point group. All calculations will use the specified Abelian group. See Sect. 13 for more details.

Options:

`auto` The molecular symmetry is automatically recognized.

*<point group symbol>* Schönflies symbol of the Abelian point group such as `C1`, `Ci`, `Cs`, `C2`, `C2v`, `C2h`, `D2`, `D2h`

Note: `cmpgrp=C1` is equivalent to `symm=off`

Default: `cmpgrp=auto`

Example: to use $C_{2v}$ point group for benzene set `cmpgrp=C2v`

`comprest` Controls the restart procedure for composite calculations. Currently, only ONIOM calculations can be restarted with the keyword.

Options:

`0` The ONIOM calculation is restarted and halted after the layer boundaries are defined. This option is designed for the setup of layer boundaries after a successful calculation on the full system.

*<any positive integer n>* The composite calculation is restarted after the $n$th subcalculation is completed. This option is designed for the modification of composite calculations after successful subcalculations of a composite method. See the keyword `subminp` to modify the initial settings.

`auto` The composite calculation is restarted from the last successful subcalculation of a composite method. This option is designed, e.g., for power failures.

`off` No restart is requested

Notes:

1. The `compress=0` option requires the `MOCOEF` file for Mrcc, the corresponding auxiliary file (`aux`) for Mopac, and the `wbo` file for xTB calculations.
2. In the case of single point ONIOM calculations with option `compress` $>$ `0`, the `ONIOMSPEC` and `iface*` files of the corresponding subcalculations are checked if they are available. Additionally, the `GRAD*` and `VARS*` files are also searched for if gradient or electronic embedding calculations are requested. Calculations using the Mopac or the xTB programs are automatically checked by the external codes.
3. Except of `compress=0`, the boundary settings are read from the `ONIOMSPEC` file, while the boundary specification in the `MINP` file is ignored.
4. In the case of `compress=auto`, the keyword `scfiguess` is set to `restart`, and the `SCFDENSITIES` file is reused if the upcoming calculation uses Mrcc.

Default: `compress=off`

Example:

1. If a series of ONIOM calculations are stopped due to power failure at unknown points of the composite procedure, type `compress=auto` to continue from the last successful subcalculation.
2. To reconfigure the boundary settings after the first ONIOM subcalculation, type `compress=0`. After the setup is made, type `compress=1` to continue with the second subcalculation.

`core` Specifies whether the core electrons are correlated.

Options:

`frozen` Frozen core approximation

`corr` All core electrons are correlated

*<any non-negative integer n>* The lowest (according to orbital energy order) $n$ pieces of spatial orbitals (the lowest $n$ pieces of alpha and $n$ pieces of beta spin orbitals for UHF/semicanonical ROHF reference) will be dropped.

Default: `core=frozen`

Example: to correlate all core electrons set `core=corr` or `core=0`

`corembed` This keyword controls the models and subsystems selected for multi-level local correlation methods. Currently it is only available for closed-shell systems using density-fitting.

Options:

`off` Conventional case, a single model defined by `calc` is used for the entire system.

`on` Multi-level calculation is performed with different local correlation methods for the active (high-level) and the environmental (low-level) subsystems. The three input lines following `corembed` define the list of active atoms, the computational model for the environment level, and the number of embedded orbitals (if it is specified). The syntax for these three lines is analogous with that for keyword `embed`. (See the description of keyword `embed`.) The high-level method for the active region should be specified by the keyword `calc`.

Default: `corembed=off`

Notes:

1. Local correlation methods available with `localcc`≥`2015` (e.g., MP2 or arbitrary single-reference CC) can be chosen for both the active and the environmental subsystem. Additionally, HF or HF+LRC are also available choices for the low-level model. If the latter is set, the environment is treated at the HF level but the long-range correlation (LRC) between the active subsystem and its environment is also taken into account (see Ref. 33). Note that models with KS-DFT reference, such as dRPA, SOSEX, etc., are not available for multi-level local correlation calculations.
2. The threshold settings of the local correlation method chosen for the high-level model can be given (as in the case of `corembed=off`) by the keywords controlling the local correlation methods (see their list in Sect. 6.9). Default settings according to `lcorthr=normal` and `localcc=2018` or `2021` (or for previous versions according to `lcorthr=loose` and `localcc=2015` or `localcc=2016`) are employed for the low-level model of the environment.

Examples:

1. LNO-CCSD(T)-in-LMP2 scheme, where LNO-CCSD(T) is performed for the active orbitals with tight thresholds, atoms 1 and 2 are included in the high-level region, and the number of the active orbitals is determined automatically:
```
calc=LNO-CCSD(T)
lcorthr=tight
corembed=on
1-2
LMP2
0
```
2. LNO-CCSDT-in-LNO-CCSD scheme, where the local CCSDT calculation is performed with the `mrcc` program for the active orbitals and the local CCSD is calculation performed with the `ccsd` program for the environment:
```
calc=LCCSDT
corembed=on
1-2
LCCSD
0
```
3. LNO-CCSD(T)-in-HF+LRC embedding where only HF is used for the environment but the additional LRC term accounts for the interaction of the active and

environmental parts. Atoms 1, 2, 3, and 5 define the active subsystem, and 10 orbitals are included in the active region:

```
calc=LNO-CCSD(T)
corembed=on
1-3,5
HF+LRC
10
```

`csapprox` Switches on the approximate long-range spin polarization approach of Ref. 51 for the restricted open-shell local correlation methods with `localcc`≥`2021`. This option treats the orbital domains without singly occupied orbitals via a closed-shell formalism with negligible approximations and considerable gain in efficiency.

Options: `on` or `off`

Default: `csapprox=on`

`cvs` Switches on the core-valence separation (CVS) for excited-state calculations [221, 222]. Only excitations including at least one core orbital specified by this keyword will be included in the excited-state wave function.

Options:

`off` The CVS approximation will not be used.

<*serial no. of the orbitals*> The serial number of the core orbitals (i.e., absolute serial number according to orbital energy order) must be given separated by commas. Serial numbers separated by dash means that all the orbitals in the specified interval will be included. For unrestricted orbitals, only the serial numbers of the alpha orbitals must be given.

Notes:

1. A projected eigenvalue equation [221] is used for `ccprog=ccsd`. It is recommended for higher-order linear-response CC and CI calculations.
2. A decoupled Hamiltonian [222] is used for `ccprog=cis`. It is recommended for CIS, TDHF, CIS(D), ADC(2), and various (DH-)TDDFT calculations [62]. In these cases, the reduced-cost and reduced-scaling approximations are also available (see the description of keyword `redcost_exc` and Ref. 63).
3. If the CVS approximation is employed, it is recommended to give an appropriate initial guess using the `ciguess` keyword.

Default: `cvs=off`

Example: to consider orbitals 1, 3, 4, and 5 set:

```
cvs=1,3-5
```

`dboc` Diagonal Born–Oppenheimer correction (DBOC) (available only with Cfour).

Options: `on` or `off`

Default: `dboc=off`

Example: for a DBOC calculation set `dboc=on` `deltaf12` Specifies whether the ΔF12 correction [67] is calculated for the DF-CCSD or DF-CCSD(T) energies.

Options:

`off` The ΔF12 correction is not calculated.

`on` The ΔF12 correction is calculated.

Note: Reduced-cost techniques are also available via the corresponding keywords.

Default: `deltaf12=off`

Example: to calculate the ΔF12 correction, the user should set `deltaf12=on`

`dendec` Selects the algorithm for the decomposition of energy denominators, Cholesky-decomposition or Laplace transform, for canonical SOS-MP2, dRPA (also required for SOSEX), SOS-CC2, SOS-CIS(D), SOS-CIS($D_\infty$), and SOS-ADC(2) as well as for local MP2 and dRPA calculations. The dRPA calculation is performed using the modified algorithm of Heßelmann [117] based on the decomposition of energy denominators. For the calculation of the SOS-MP2 energy, in practice one dRPA iteration is performed with the aforementioned algorithm. In the case of local MP2 and dRPA calculations the correlation energy contributions are also evaluated with the aid of the decomposition of energy denominators (see Ref. 30). The algorithm for the decomposition can be set using this keyword in all of the above cases. The number of retained Cholesky vectors/quadrature points can be controlled by keyword `nchol`.

Options:

`off` for SOS-CC2, SOS-CIS(D), SOS-CIS($D_\infty$), and SOS-ADC(2) calculations the decomposition will not be used, but a fifth-power scaling algorithm will be executed

`Cholesky` Cholesky decomposition will be used

`Laplace` Laplace transform will be used

Default: `dendec=Laplace` for SOS-MP2, SOS-CC2, SOS-CIS(D) SOS-CIS($D_\infty$), and SOS-ADC(2); `dendec=Cholesky` otherwise

Notes:

1. The algorithms based on the Laplace-transformed technique use minimax quadratures obtained from Ref. 223.
2. The default quadratures are taken from the `Quad` file which is located in the `BASIS` directory created at the installation. In addition to the default quadratures, any further quadrature can be used by adding it to the `BASIS/Quad` file or alternatively to the `GENBAS` file to be placed in the directory where MRCC is executed. The format is as follows. On the first line give the label of the quadrature as `KNNRXXX`, where `NN` is the number of the quadrature points and `XXX` is the upper limit of the interval in which the Laplace transform is approximated (variable $R$ in Ref. 223). The subsequent `NN` lines must contain, respectively, the weights and quadrature points.

Example: to use Laplace transform give `dendec=Laplace` `dens` Construction of density, derivative density, and transition density matrices for property calculations. If mod(`dens`,2)=1, only one-particle, if mod(`dens`,2)=0, both one- and two-particle density matrices will be calculated and contracted with the available property integrals. See Refs. 5–7, 11, 13, 14 for more details.

Options:

`1, 2` Density-matrix calculation (for geometry optimizations, first-order properties, etc.)

`3, 4` Density-matrix first derivatives (for second-order property calculations, available only with CFOUR)

`5, 6` Transition density matrices (for transition moment calculations)

`7, 8` Second and third derivatives of the density-matrix (for third-order property calculations, available only with CFOUR)

Default: `dens=2` for geometry optimizations, QM/MM, and higher-order CC pair density DBBSC calculations, `dens=0` otherwise

Notes:

1. Transition moment as well as excited-state gradient calculations can be performed for only one excited state at a time, that is, `nsing`, `ntrip`, or `nstate` cannot exceed 2. To compute the transition moment or gradient for a higher excited state you need to converge the equations to that root. The best practice is to run a calculation with the desired number of excited states, and then restart the calculation selecting a higher solution (see the description of keyword `rest`). You can also try to start the calculation from a good initial guess (see the description of keyword `ciguess`).
2. If `dens` $\neq$ `0`, a population analysis is also performed, and Mulliken and Löwdin atomic charges as well as Mayer bond orders are computed.

Example: for the calculation of both one- and two-particle density matrices set `dens=2`

`denscorr` Specifies whether the density-based basis-set correction (DBBSC) proposed by Giner, Toulouse, and co-workers [136] is calculated.

Options:

`0` The DBBSC is not calculated.

`1` Both the CCSD(T) energy and the DBBSC using HF pair density are calculated.

`2` Only the DBBSC using HF pair density is calculated.

`3` The DBBSC using CC pair density and HF one-electron density is calculated.

`4` Similar to `denscorr=3`, but the CC one-electron density is used.

Notes:

1. To calculate the correction, even if the `denscorr=2` option is used, the following format should be used (see keyword `dft`):

```
calc=DF-CCSD(T)
dft=userd
1
```

```
1.00 HFx
1
1.0 ECMD_PBEc
```

2. The correction is also available for double-hybrid DFT functionals [92], for example, using the PBE0-2 functional:
   ```
   denscorr=2
   calc=DF-CCSD(T)
   dft=userd
   3
   0.2063 PBEx
   0.5000 PBEc
   0.7937 HFx
   1
   0.5000 ECMD_PBEc
   ```
   As shown, the MP2 contribution is not included in this calculation. This must be requested in a subsequent calculation using the `scfiguess=off` and `dft=user` keywords, where the MP2 contribution is also defined using the appropriate scaling factor.
3. DBBSC calculations using local domain approximations are also available [71]. In this case, the user should use the `calc=LMP2` or `calc=LNO-CCSD(T)` keywords. This option is only available with the `denscorr=1` setting and an RHF reference.
4. Within this framework, DBBSC-DH calculations are also available using the following format, for example, with the PBE0-2 functional:
   ```
   denscorr=1
   calc=LMP2
   dft=userd
   4
   0.2063 PBEx
   0.5000 PBEc
   0.7937 HFx
   0.5000 MP2
   1
   0.5000 ECMD_PBEc
   ```
5. DBBSC calculations using higher-order CC densities are also available [72]. In this case, the corresponding CC densities are used which is specified after the `calc` keyword, for example, `calc=CCSDT`. The calculation of the CABS correction is not available in this case.
6. In the case of ROHF calculations, the semicanonical orbitals are used by default (see keywords `rohfcore` and `rohftype`). When comparing the results from other programs, it might be important to verify the ROHF orbitals.

Default: `denscorr=0`

Example: for only the DBSSC using HF pair density, the user should set `denscorr=2`

`dfalg` Specifies how the inverse of the two-center Coulomb integral matrix is decomposed in density fitting direct SCF calculations.

Options:

`LinEq` The fitting coefficients are computed by solving the corresponding system of linear equations. It is efficient and numerically stable. It is the best choice for very large auxiliary basis sets for which the diagonalization of the two-center integral matrix is prohibitive.

`InvSqrt` Inverse square root of the two-center integral matrix is used. It is relatively stable numerically, but the diagonalization is slow and requires much memory.

`Cholesky` Cholesky decomposition of the inverse of the two-center integral matrix is used. It is an efficient algorithm but numerically unstable if the two-center matrix tends to be singular.

Default: `dfalg=InvSqrt` for property calculations, `dfalg=LinEq` otherwise

Example: to use Cholesky decomposition set `dfalg=Cholesky`

`dfbasis_cab` Specifies the complimentary auxiliary basis set (CABS) for explicitly correlated methods.

Options:

`none` No CABS.

<*basis set label*>, `atomtype`, `special` The specified basis set is used as CABS. For the specification of the basis the same rules apply as for keyword `basis`, see the description of keyword `basis`.

`auto` This option can only be used if Dunning's aug-cc-pV$X$Z or Peterson's cc-pV$X$Z-F12 basis sets are used as the normal basis set. In this case, if `dfbasis_cab=auto`, the corresponding "OPTRI" basis set will be used as CABS.

Note: For the available CABSs, see the notes for keyword `basis`.

Default: `dfbasis_cab=auto` for all the explicitly correlation methods, `dfbasis_cab=none` otherwise.

Example: To use the cc-pVTZ-F12-OPTRI CABS with the aug-cc-pVDZ AO basis set:

```
basis=aug-cc-pVDZ
dfbasis_cab=cc-pVTZ-F12-OPTRI
```

`dfbasis_cor` Specifies whether the density fitting approximation will be used in the correlated calculations and also specifies the fitting basis set.

Options:

`none` The density fitting approximation is not used for the correlated calculation.

<*basis set label*>, `atomtype`, `special` The density fitting approximation is invoked, and the specified basis set is used as fitting basis set. For the specification of the basis the same rules apply as for keyword `basis`, see the description of keyword `basis`.

`auto` This option can only be used if Dunning's (aug-)cc-pV$X$Z or (aug-)cc-pV($X$+d)Z, Weigend and Ahlrichs' def2, the augmented def2 basis sets of Rappoport and Furche, Peterson's cc-pV$X$Z-F12 or (aug-)cc-pV$X$Z-PP, or Pople's basis sets are used as the normal basis set. In this case, if `dfbasis_cor=auto`, the density fitting approximation is invoked. For the (aug-)cc-pV$X$Z(-PP) and (aug-)cc-pV($X$+d)Z basis sets the corresponding (aug-)cc-pV$X$Z(-PP)-RI basis sets will be used automatically as the fitting basis sets, while for a cc-pV$X$Z-F12 basis set the corresponding aug-cc-pV$X$Z-RI basis will be taken. For the (augmented) def2 basis sets also the corresponding RI basis sets will be used, e.g., def2-TZVPP-RI for def2-TZVPP, def2-QZVPP-RI for def2-QZVPP, def2-TZVPPD-RI for def2-TZVPPD, etc. For Pople-type minimal and double-$\zeta$ basis sets (i.e., STO-3G, 3-21G, 6-31G**, etc.) the cc-pVDZ-RI basis set, while for triple-$\zeta$ basis sets (i.e., 6-311G, 6-311G**, etc.) the cc-pVTZ-RI basis set will be used as the auxiliary basis; if the basis also includes diffuse functions (i.e., 6-31+G**, 6-311++G**, etc.) the aug-cc-pVDZ-RI and aug-cc-pVTZ-RI basis sets are employed by default.

Notes:

1. For the available fitting basis sets see the notes for keyword `basis` on page 43.
2. The density fitting approximation can also be invoked by attaching the prefix `DF-` or `RI-` to the corresponding option of keyword `calc`, see the description of `calc`.

Default: `dfbasis_cor=auto` for all the correlation methods that use the density fitting approximation by default as well as for local correlation calculations (i.e., `localcc` $\neq$ `off`), `dfbasis_cor=none` otherwise.

Examples:

1. To use the cc-pVTZ-RI fitting basis in the correlated calculation for all atoms the input must include `dfbasis_cor= cc-pVTZ-RI`
2. Consider the water molecule and use the cc-pVTZ-RI fitting basis set for the hydrogens and aug-cc-pVTZ-RI for the oxygen. The following inputs are equivalent:

```
dfbasis_cor=atomtype
O:aug-cc-pVTZ-RI
H:cc-pVTZ-RI
```

   or

```
dfbasis_cor=aug'-cc-pVTZ-RI
```

3. Consider the water molecule and use the cc-pVTZ (cc-pVTZ-RI) basis set (fitting basis set) for the hydrogens and aug-cc-pVTZ (aug-cc-pVTZ-RI.) for the oxygen in a local correlation calculation. The following inputs are equivalent:

```
calc=CCSD(T)
localcc=on
basis=aug'-cc-pVTZ
dfbasis_scf=aug'-cc-pVTZ-RI
dfbasis_cor=aug'-cc-pVTZ-RI
```

   or

```
calc=LCCSD(T)
basis=aug'-cc-pVTZ
```

4. To run a DF-HF calculation with the cc-pVTZ-F12 basis set and the aug-cc-pVTZ-RI auxiliary basis the input should only include the following lines:

```
basis=cc-pVTZ-F12
calc=DF-HF
```

`dfbasis_scf` Specifies whether the density fitting approximation will be used in the HF- or KS-SCF calculation and also specifies the fitting basis set. For the syntax see the description of keyword `dfbasis_cor`. The important difference is that, if `dfbasis_scf=auto`, the (aug-)cc-pV$XZ$-RI-JK basis sets will be used as auxiliary basis sets for Dunning's, Peterson's, and Pople's basis sets, while for the def2 basis sets the def2-QZVPP-RI-JK auxiliary basis is taken. For the augmented def2 as well as for the aug-cc-pV$XZ$-PP basis sets the def2-QZVPPD-RI-JK auxiliary basis will be used. Exceptions are the KS-DFT calculations with non-hybrid functionals, where the def2-QZVPPD-RI-J basis set will be used in the following cases: for H with the cc-pVDZ, aug-cc-pVDZ, cc-pVDZ-F12, def2-SV(P), def2-SVP, def2-TZVP, def2-SVPD, def2-TZVPD, and LANL2DZ AO basis sets; for other atoms with the cc-pVDZ, cc-pVTZ, cc-pV(D+d)Z, cc-pV(T+d)Z, aug-cc-pVDZ, aug-cc-pVTZ, aug-cc-pV(D+d)Z, aug-cc-pV(T+d)Z, cc-pVDZ-F12, def2-SV(P), def2-SVP, def2-TZVP, def2-TZVPP, def2-SVPD, def2-TZVPD, def2-TZVPPD, and LANL2DZ AO basis sets.

Default: `dfbasis_scf=auto` if `dfbasis_cor`≠`none` and for DFT calculations, `dfbasis_scf=none` otherwise.

`dfbasis_scf_sm` Specifies the small fitting basis set for dual auxiliary basis set calculations. Any basis set can be used as a small auxiliary basis set which is available in Mrcc or you can use your own sets using the GENBAS file. For a small set of atoms (H, C, N, O, F, P, S, and Cl), preoptimized cc-pV(T,Q)Z-RI-JK-dual basis sets are also available [50]. These are the recommended ones for calculations with the (aug-)cc-pV(T,Q)Z AO basis sets. See also the description of keyword `dual_df` for further details and examples.

Options: Same as `dfbasis_scf`.

Default: `dfbasis_scf_sm=auto` according to `basis_sm`

Notes: For more information on dual auxiliary basis calculations see the keyword `dual_df`.

`dfintran` Specifies the integral transformation program to be used for the transformation of three-center Coulomb integrals.

Options:

- `drpa` the `drpa` program will be called
- `ovirt` the `ovirt` program will be called

Default: `dfintran=ovirt` if `ovirt`≠`off`, `dfintran=drpa` otherwise.

Example: to use the `ovirt` code set `dfintran=ovirt`

`dft` Use this keyword to perform DFT calculations and to specify the functional.

Options:

- `off` No DFT calculation is carried out.
- <*functional name*> The name of the functional, see Table 4 for the available functionals.

*<Libxc identifier>* The identifier of a functional implemented in the LIBXC library (if installed), such as `LDA_X`, `LDA_C_VWN_1`, `GGA_X_B88`, etc. (see the homepage of the LIBXC project [173]).

`user` User-defined functional. Any combination of the following contributions can be defined:

- the available standalone functionals, see column "User" in Table 4.
- the functionals available in the LIBXC library (if installed), use simply the LIBXC identifier of the functionals (see the homepage of the LIBXC project [173]).
- the HF exchange, denoted by `HFx`
- the long-range HF exchange, denoted by `lrHFx`. The range-separation parameter ($\omega$) should be specified in a.u. after the `lrHFx` flag separated by space, see the example below. If the parameter is not set, $\omega = 0.15$ a.u. will be used.
- the short-range HF exchange, denoted by `srHFx`. The syntax is same as for `lrHFx`, see the example below.
- the MP2, MP3, dRPA, and SOSEX correlation, denoted, respectively, by `MP2`, `MP3`, `dRPA`, and `SOSEX`;
- the antiparallel- and parallel-spin components of the latter correlation corrections, add the `s` and `t` postfix to the above labels, respectively, e.g., instead of the `MP2` label, the `MP2s` and `MP2t` labels should be used.
- the long- and short-range dRPA correlation denoted, respectively, by `lrdRPA` and `srdRPA`. The corresponding spin components, `lrdRPAs`, `srdRPAs`, `lrdRPAt`, and `srdRPAt`, are also available; the syntax is same as for `lrHFx`.

Note that for hybrid functionals, such as B97, the HF exchange will be neglected. The combination should be specified in the subsequent lines as follows (see also the examples below):
*<number of entries>*
*<coefficient 1> <functional name 1>*
*<coefficient 2> <functional name 2>*
*<coefficient 3> <functional name 3>*
...

`userd` User-defined functional, but different functionals are used for the calculation of the density and the energy. It is useful for defining special double-hybrid functionals. The combination should be specified in the subsequent lines as follows (see also the examples below):
*<number of entries for density>*
*<coefficient 1> <functional name 1>*
*<coefficient 2> <functional name 2>*
*<coefficient 3> <functional name 3>*
...
*<number of entries for energy>*
*<coefficient 1'> <functional name 1'>*
*<coefficient 2'> <functional name 2'>*

<*coefficient 3'*> <*functional name 3'*>
. . .
See option `user` for the possible values of <*functional name n*> and <*functional name n'*>. The weight of the HF exchange (`HFx`), if any, can be different for the density and the energy, and, in contrast to previous versions of Mrcc, must be specified also in the second block.

`rsdh` Range-separated double hybrid functionals of Mester and Kállay [49] based on the ansatz of Kalai and Toulouse [107]. The entries should be specified in the subsequent lines as follows (see also the examples below):
`ssr`<*functional name 1*>
`ssr`<*functional name 2*>
. . .
<*coefficient 1*>
<*coefficient 2*>
Here, <*coefficient 1*> and <*coefficient 2*> stand for the parameters $\lambda$ and $\omega$, respectively. `ssr`<*functional name i*> is the range-separated variant of the corresponding functional evaluated utilizing the local-scaling approximation proposed by Scuseria and co-workers [224]. This framework works with arbitrary combination of LDA/GGA/meta-GGA exchange and correlation functionals using either the built-in ones or those from the Libxc library.

Default: `dft=PBE` for dRPA, RPA, SOSEX, and for scaled-equation and down-scaled dRPA and SOSEX; `dft=PBEx` for RPAX2; otherwise `dft=off`

Notes:

1. Most of the the built-in functionals implemented in Mrcc were obtained from the Density Functional Repository [225, 226]. Other functionals are available via the Libxc interface [171, 173] and require the Libxc library, see Sect. 7.2 for the installation of Libxc.
2. Empirical dispersion corrections can be calculated for particular functionals and also for the HF energy using the DFT-D3 approach of Grimme and co-workers [227, 228] by attaching the `-D3` postfix to the corresponding options: `BLYP-D3`, `BHLYP-D3`, `B3LYP-D3`, `B3PW91-D3`, `BP86-D3`, `PBE-D3`, `PBE0-D3`, `HCTH120-D3`, `B2PLYP-D3`, `mPW1B95-D3`, `TPSS-D3`, `TPSSh-D3`, `B2GPPLYP-D3`, `DSDPBEP86-D3`, `DSDPBEhB95-D3`, `CAM-B3LYP-D3`, `wB97X-D3`, `M06-L-D3`, `MN15-D3`, `M06-2X-D3`, `LC-wPBE-D3`, `PW91-D3`, `SCAN-D3`, `dRPA75-D3`, `HF-D3`. See also the description of keyword `edisp`.
3. For a simple DFT calculation (i.e., without subsequent correlation calculations) the value of keyword `calc` can be `SCF`, `HF`, `RHF`, or `UHF`. Note that you do not need to set its value since it is set to `SCF` by default. Alternatively, you can select the DFT functional using keyword `calc`, and in this case you do not have to set keyword `dft` (see the description of `calc`).
4. For a correlated calculation with KS orbitals you should select the functional with this keyword, and the value of keyword `calc` must be set to the desired correlation method. Note that you can also accelerate the post-KS calculation using local correlation schemes (e.g., local dRPA). See the examples below.

Table 4: Functionals (the options for keyword `dft`) implemented in MRCC. The rightmost column shows if the functional can be used with the `user`/`userd` options.

| Functional | Description | User |
|---|---|---|
| LDA exchange functionals | | |
| `LDA` | Slater–Dirac exchange (local density approximation) [102, 229, 230] | Yes |
| LDA correlation functionals | | |
| `VWN1` | functional I of Vosko, Wilk, and Nusair [231] | Yes |
| `VWN2` | functional II of Vosko, Wilk, and Nusair [231] | Yes |
| `VWN3` | functional III of Vosko, Wilk, and Nusair [231] | Yes |
| `VWN4` | functional IV of Vosko, Wilk, and Nusair [231] | Yes |
| `VWN5` | functional V of Vosko, Wilk, and Nusair [231] | Yes |
| `PZ` | Perdew–Zunger 1981 correlation functional [232] | Yes |
| `PW` | Perdew–Wang 1992 correlation functional [233] | Yes |
| GGA exchange functionals | | |
| `B88` | Becke's 1988 exchange functional [234] | Yes |
| `PBEx` | functional of Perdew, Burke, and Ernzerhof [235] | Yes |
| `PBEh` | 1988 revision of PBEx by Ernzerhof and Perdew [236] | Yes |
| `PW91x` | Perdew–Wang 1991 exchange functional [237] | Yes |
| `G96` | exchange functional of Gill [235] | Yes |
| `mPW91x` | modified PW91x functional of Adamo and Barone [238] | Yes |
| `NCAP` | nearly correct asymptotic potential functional of Carmona-Espíndola *et al.* [239] | Yes |
| GGA correlation functionals | | |
| `LYP` | correlation functional of Lee, Yang, and Parr [240] | Yes |
| `P86` | Perdew's 1986 correlation functional [241] | Yes |
| `PBEc` | functional of Perdew, Burke, and Ernzerhof [235] | Yes |
| `PW91c` | Perdew–Wang 1991 correlation functional [237] | Yes |
| GGA exchange-correlation functionals | | |
| `BLYP` | Becke's 1988 exchange functional [234] and the correlation functional of Lee, Yang, and Parr (`B88` + `LYP`) [240] | No |
| `BP86` | BP86 exchange-correlation functional (`B88` + `P86`) [234, 241] | No |
| `PBE` | exchange-correlation functional of Perdew, Burke, and Ernzerhof (`PBEx` + `PBEc`) [235] | No |
| `PW91` | Perdew and Wang 1991 exchange-correlation functional (`PW91x` + `PW91c`) [237] | No |
| `HCTH120` | HCTH120 exchange-correlation functional of Boese and co-workers [242] | Yes |
| `HCTH147` | HCTH147 exchange-correlation functional of Boese and co-workers [242] | Yes |
| `HCTH407` | HCTH407 exchange-correlation functional of Boese and Handy [243] | Yes |

| | | |
|---|---|---|
| `XLYP` | exchange-correlation functional of Xu and Goddard [244] | Yes |
| `mPWLYP1w` | exchange-correlation functional of Dahlke and Truhlar optimized for water [245] | Yes |
| Hybrid GGA exchange-correlation functionals | | |
| `BHLYP` | Becke's half-and-half exchange in combination with the LYP correlation functional (0.5 `B88` + 0.5 HF exchange + `LYP`) [234, 240, 246] | No |
| `B3LYP` | Becke's three-parameter hybrid functional including the correlation functional of Lee, Yang, and Parr (0.08 `LDA` + 0.72 `B88` + 0.2 HF exchange + 0.19 `VWN5` + 0.81 `LYP`) [229–231, 234, 240, 247] | No |
| `B3LYP3` | Becke's three-parameter hybrid functional including the correlation functional of Lee, Yang, and Parr (0.08 `LDA` + 0.72 `B88` + 0.2 HF exchange + 0.19 `VWN3` + 0.81 `LYP`) [229–231, 234, 240, 247, 248]. Note that this is equivalent to the B3LYP functional of the GAUSSIAN package. | Yes |
| `B3PW91` | Becke's three-parameter hybrid functional including the 1991 correlation functional of Perdew and Wang (0.08 `LDA` + 0.72 `B88` + 0.2 HF exchange + 0.19 `VWN5` + 0.81 `PW91c`) [229–231, 234, 237, 247] | No |
| `B1LYP` | modified B3LYP functional of Adamo and Barone [249] | Yes |
| `O3LYP` | modified B3LYP functional of Cohen and Handy [250] | Yes |
| `B97` | Becke's 1997 exchange-correlation functional (including 0.1943 HF exchange) [251] | Yes |
| `PBE0` | hybrid functional of Perdew, Burke, and Ernzerhof (0.75 `PBEx` + 0.25 HF exchange + `PBEc`) [235, 252] | No |
| `X3LYP` | hybrid functional of Xu and Goddard [244] | Yes |
| Meta-GGA exchange functionals | | |
| `TPSSx` | exchange functional of Tao, Perdew, Staroverov, and Scuseria [253] | Yes |
| `revTPSSx` | revised TPSS exchange of Perdew *et al.* [254] | Yes |
| `SCANx` | exchange functional of Sun, Ruzsinszky, and Perdew [255] | Yes |
| Meta-GGA correlation functionals | | |
| `B95` | Becke's 1995 correlation functional [256] | Yes |
| `TPSSc` | correlation functional of Tao, Perdew, Staroverov, and Scuseria [253] | Yes |
| `revTPSSc` | revised TPSS correlation of Perdew *et al.* [254] | Yes |
| `SCANc` | correlation functional of Sun, Ruzsinszky, and Perdew [255] | Yes |
| `revSCANc` | revised SCAN correlation functional of Mezei, Csonka, and Kállay [257] | Yes |
| Meta-GGA exchange-correlation functionals | | |
| `TPSS` | exchange-correlation functional of Tao, Perdew, Staroverov, and Scuseria [253] | No |
| `revTPSS` | revised TPSS functional of Perdew *et al.* [254] | No |
| `M06-L` | 2006 exchange-correlation functional of Zhao and Truhlar [258, 259] | No |

| | | |
|---|---|---|
| `B97M-V` | exchange-correlation functional of Mardirossian and Head-Gordon (with self-consistent VV10) [260] | Yes |
| `SCAN` | exchange-correlation functional of Sun, Ruzsinszky, and Perdew [255] | No |
| `revSCAN` | revised SCAN exchange-correlation functional of Mezei, Csonka, and Kállay [257] | No |
| Hybrid meta-GGA exchange-correlation functionals | | |
| `M06-2X` | 29-parameter exchange-correlation functional of Zhao and Truhlar including 0.54 HF exchange [258] | No |
| `M06-HF` | hybrid meta-GGA functional of Zhao and Truhlar including 100% HF exchange [261] | No |
| `M08-HX` | 47-parameter exchange-correlation functional of Zhao and Truhlar including 0.5223 HF exchange [262] | Yes |
| `M08-SO` | 44-parameter exchange-correlation functional of Zhao and Truhlar including 0.5679 HF exchange [262] | Yes |
| `TPSSh` | hybrid version of TPSS including 0.1 HF exchange [263] | Yes |
| `revTPSSh` | revised TPSSh of Csonka, Perdew, and Ruzsinszky including 0.1 HF exchange [263, 264] | Yes |
| `mPW1B95` | mixture of mPW91x and B95 by Zhao and Truhlar [265] | Yes |
| `PW6B95` | mixture of PW91x and B95 by Zhao and Truhlar [266] | Yes |
| `MN15` | the MN15 hybrid of Peverati and Truhlar (MN15 exchange-correlation + 0.44 HF exchange) [267] | No |
| `SCAN0` | hybrid version of SCAN including 0.25 HF exchange [255, 268] | No |
| `revSCAN0` | hybrid version of revised SCAN including 0.25 HF exchange [257] | No |
| Range-separated GGA correlation functionals | | |
| `srPBEc` | short-range PBE correlation functional of Goll *et al.* [269]; only energy is implemented, it can be used only in the second block of `userd` | Yes |
| `ECMD_PBEc` | PBEc-based ECMD functional of Giner *et al.* [136]; only energy is implemented, it can be used only in the second block of `userd` | Yes |
| Range-separated hybrid GGA exchange-correlation functionals | | |
| `HSE06` | the 2006 version of the range-separated hybrid of Heyd, Scuseria, and Ernzerhof (modified PBE + 0.25 short-range HF exchange, $\omega$ = 0.11 a.u.) [270, 271] | No |
| `CAM-B3LYP` | Coulomb-attenuating B3LYP functional of Yanai *et al.* (short-range B3LYP + 0.19 HF exchange + 0.46 long-range HF exchange, $\omega$ = 0.33 a.u.) [145] | No |
| `LC-wPBE` | the LC-$\omega$PBE range-separated hybrid of Vydrov and Scuseria (short-range PBE + long-range HF exchange, $\omega$ = 0.40 a.u.) [272] | No |
| `wB97` | the $\omega$B97 range-separated hybrid of Chai and Head-Gordon ($\omega$B97 exchange-correlation + 1.0 long-range HF exchange, $\omega$ = 0.44 a.u.) [273] | No |

| | | |
|---|---|---|
| `wB97X` | the $\omega$B97X range-separated hybrid of Chai and Head-Gordon ($\omega$B97X exchange-correlation + 0.157706 HF exchange + 0.842294 long-range HF exchange, $\omega = 0.3$ a.u.) [273] | No |
| `wB97X-V` | the $\omega$B97X-V range-separated hybrid of Mardirossian and Head-Gordon ($\omega$B97X-V exchange-correlation + self-consistent VV10 + 0.167 HF exchange + 0.833 long-range HF exchange, $\omega = 0.3$ a.u.) [274] | No |
| Range-separated hybrid meta-GGA exchange-correlation functionals | | |
| `M11` | the M11 range-separated hybrid of Peverati and Truhlar (M11 exchange-correlation + 1.0 HF exchange - 0.572 short-range HF exchange, $\omega = 0.25$ a.u.) [275] | No |
| `MN12-SX` | the MN12-SX range-separated hybrid of Peverati and Truhlar (MN12-SX exchange-correlation + 0.25 short-range HF exchange, $\omega = 0.11$ a.u.) [276] | No |
| `wB97M-V` | the $\omega$B97M-V range-separated hybrid of Mardirossian and Head-Gordon ($\omega$B97M-V exchange-correlation + self-consistent VV10 + 0.15 HF exchange + 0.85 long-range HF exchange, $\omega = 0.3$ a.u.) [277] | No |
| Double hybrid functionals | | |
| `B2PLYP` | Grimme's two-parameter double hybrid functional including MP2 correction (0.47 `B88` + 0.53 HF exchange + 0.73 `LYP` + 0.27 MP2 correlation) [105] | No |
| `B2GPPLYP` | two-parameter double hybrid functional including MP2 correction of Martin and co-workers (0.35 `B88` + 0.65 HF exchange + 0.64 `LYP` + 0.36 MP2 correlation) [278] | No |
| `DSDPBEP86` | dispersion corrected, spin-component scaled double hybrid functional of Kozuch and Martin (0.30 `PBEx` + 0.70 HF exchange + 0.43 `P86` + 0.53 MP2 antiparallel-spin correlation + 0.25 MP2 parallel-spin correlation) [279, 280]. Note that the dispersion correction is only included if the `-D3` postfix is added (see the note below). | No |
| `DSDPBEhB95` | dispersion corrected, spin-component scaled double hybrid functional of Kozuch and Martin (0.34 `PBEh` + 0.66 HF exchange + 0.55 `B95` + 0.47 MP2 antiparallel-spin correlation + 0.09 MP2 parallel-spin correlation) [280]. Note that the dispersion correction is only included if the `-D3` postfix is added (see the note below). | No |
| `XYG3` | double hybrid functional of Zhang, Xu, and Goddard (0.2107 `B88` - 0.014 `LDA` + 0.8033 HF exchange + 0.6789 `LYP` + 0.3211 MP2 correlation evaluated with B2LYP orbitals) [281, 282] | No |
| `SCAN0-2` | SCAN-based double-hybrid of Hui and Chai (0.793701 HF exchange + 0.206299 `SCANx` + 0.5 `SCANc` + 0.5 MP2 correlation [255, 268] | No |

| | | |
|---|---|---|
| `dRPA75` | the dual-hybrid random phase approximation (dRPA75) method of Mezei *et al.* [28]. The KS orbitals are obtained with the "0.25 `PBEx` + 0.75 HF exchange + `PBEc`" functional, while the energy is calculated using the "0.25 `PBEx` + 0.75 HF exchange + dRPA correlation" expression. Dispersion correction [283] can be included if the `-D3` postfix is added. | No |
| `SCS-dRPA75` | the spin-component scaled dual-hybrid random phase approximation (SCS-dRPA75) method of Mezei *et al.* [28, 32]. The KS orbitals are obtained with the "0.25 `PBEx` + 0.75 HF exchange + `PBEc`" functional, while the energy is calculated using the "0.25 `PBEx` + 0.75 HF exchange + 1.5 dRPA antiparallel-spin correlation + 0.5 dRPA parallel-spin correlation" expression. | No |
| Range-separated double hybrid functionals | | |
| `RS-PBE-PBE` | PBE exchange- and PBE correlation-based range-separated double hybrid functional of Mester and Kállay ($\lambda = 0.5$ and $\omega = 0.7$ a.u.) [49] | No |
| `RS-PBE-P86` | PBE exchange- and P86 correlation-based range-separated double hybrid functional of Mester and Kállay ($\lambda = 0.5$ and $\omega = 0.7$ a.u.) [49] | No |
| `RS-B88-LYP` | B88 exchange- and LYP correlation-based range-separated double hybrid functional of Mester and Kállay ($\lambda = 0.5$ and $\omega = 0.7$ a.u.) [49] | No |
| `RS-PW91-PW91` | PW91 exchange- and PW91 correlation-based range-separated double hybrid functional of Mester and Kállay ($\lambda = 0.5$ and $\omega = 0.7$ a.u.) [49] | No |
| van der Waals density functionals | | |
| `VV10` | the nonlocal part of the 2010 van der Waals density functional of Vydrov and Van Voorhis [284], both self-consistent and non-self-consistent implementations are available, see also the comment below | Yes |
| `VV10NL` | same as `VV10` but the $\beta N$ term is ignored | Yes |

5. For a correlated calculation with KS orbitals (excluding calculations with double hybrid functionals) the HF energy computed with KS orbitals is used as reference energy.
6. For the B2PLYP, B2GPPLYP, DSDPBEP86, DSDPBEhB95, dRPA75, etc. double hybrid functionals as well as for user-defined double hybrid functionals including MP2 (SCS-MP2), dRPA, etc. correlation `calc` is automatically set to `MP2`, `dRPA`, etc. Note that you can accelerate the MP2, dRPA, . . . part of a double hybrid DFT calculation for large molecules using local correlation approaches. For the built-in double hybrid functionals just add the "L" prefix, while for the user-defined functionals set `localcc=on`. See the examples below.
7. The DSDPBEP86, DSDPBEhB95, and dRPA75 functionals use special param-

eters for the calculation of the D3 correction which are read by the `DFT-D3` program from the `.dftd3par.$HOST` file located in your home directory. This file will be created by the program, but you must be sure that the program is able to access your home directory. Also note that, if you already have this file in your home, it will be overwritten, so please do not forget to save it before executing MRCC.

8. For the VV10 van der Waals functional you can modify parameters $b$ and $C$ (see Ref. 284) if it is used with the `user` or `userd` options. For that purpose the two parameters should be specified after the `VV10` or `VV10NL` flag separated by spaces, see the example below. If the parameters are not set, those of Ref. 284 will be used.
9. The range-separated variant of a functional is available with the `ssr` prefix utilizing the local-scaling approximation proposed by Scuseria and co-workers [224]. This framework works for arbitrary LDA/GGA/meta-GGA exchange and correlation functionals using the either built-in ones or those from the LIBXC library.
10. ECMD functionals proposed by Giner and co-workers [136] are available with the `ECMD_` prefix. This framework works for the built-in PBEc functional or arbitrary LDA/GGA/meta-GGA correlation functionals from the LIBXC library.
11. MP2-F12-based DH-DFT-F12 [220] calculations are also available using the `dft=user` option, with the proper definition of both the `MP2s` and `MP2t` contributions. Please note that spin-scaled DH functionals are available only for UHF and semicanonical ROHF references. See the example below.

Examples:

1. To perform a DFT calculation with the B3LYP functional give `dft=B3LYP` or `calc=B3LYP`
2. The B3LYP functional can also be defined using the `user` option as
   ```
   calc=scf
   dft=user
   5
   0.08 LDA
   0.72 B88
   0.20 HFx
   0.19 VWN5
   0.81 LYP
   ```
3. The B2PLYP double-hybrid functional can also be defined using the `user` option as
   ```
   calc=scf
   dft=user
   4
   0.47 B88
   0.73 LYP
   0.53 HFx
   0.27 MP2
   ```
4. The DSDPBEP86 double-hybrid functional can also be defined using the `user`

option as

```
calc=SCF
dft=user
5
0.30 PBEx
0.43 P86
0.70 HFx
0.53 MP2s
0.25 MP2t
```

5. SOSEX calculation with Kohn–Sham orbitals calculated with the LDA exchange functional:

   ```
   calc=SOSEX
   dft=LDA
   ```

6. To perform a DFT calculation with the B2PLYP double-hybrid functional and add the D3 dispersion correction set `dft=B2PLYP-D3` or `calc=B2PLYP-D3`
7. B2PLYP calculation, the MP2 contribution is evaluated using local MP2 approximation:

   ```
   calc=LB2PLYP
   ```

8. User-defined functional, different functionals are used for the calculation of the density (0.25 `PBEx` + 0.75 HF exchange + `PBEc`) and the energy (0.50 `PBEx` + 0.50 HF exchange + MP2 correlation).

   ```
   dft=userd
   3
   0.75 HFx
   0.25 PBEx
   1.00 PBEc
   3
   0.50 HFx
   0.50 PBEx
   1.00 MP2
   ```

9. The dRPA75 dual-hybrid functional can also be defined using the `userd` option as

   ```
   dft=userd
   3
   0.75 HFx
   0.25 PBEx
   1.00 PBEc
   3
   0.75 HFx
   0.25 PBEx
   1.00 dRPA
   ```

10. Local dRPA calculation with Kohn–Sham orbitals calculated with the PBE functional:

    ```
    calc=LdRPA
    dft=PBE
    ```

11. To perform a DFT calculation with the B3LYP functional using its Libxc implementation set `calc=HYB_GGA_XC_B3LYP5`
12. The B3LYP functional can also be defined using the `user` option and the functionals implemented in the Libxc library as
    ```
    dft=user
    5
    0.08 LDA_X
    0.72 GGA_X_B88
    0.20 HFx
    0.19 LDA_C_VWN
    0.81 GGA_C_LYP
    ```
13. DFT calculation with a user-defined PBE0-VV10 functional. Parameter $b$ of VV10 is modified, while for $C$ its default value, 0.0093, is used. If you do not want to modify either $b$ or $C$, simply drop the two numbers for the `VV10` entry.
    ```
    dft=userd
    3
    0.75 PBEx
    0.25 HFx
    1.00 PBEc
    4
    0.75 PBEx
    0.25 HFx
    1.00 PBEc
    1.00 VV10 8.0 0.0093
    ```
14. DFT calculation with a user-defined CAM-B3LYP functional. A range-separation parameter ($\omega$) of 0.33 a.u. is set. If you want to use the default $\omega = 0.15$ a.u., simply drop the last number for the `lrHFx` entry.
    ```
    dft=user
    3
    1.00 HYB_GGA_XC_CAM_B3LYP
    0.19 HFx
    0.46 lrHFx 0.33
    ```
15. An alternative implementation of CAM-B3LYP:
    ```
    dft=user
    3
    1.00 HYB_GGA_XC_CAM_B3LYP
    0.65 HFx
    -0.46 srHFx 0.33
    ```
16. Alternative definition for the RS-PBE-P86 functional utilizing built-in functionals:
    ```
    dft=rsdh
    ssrPBEx
    ssrP86
    0.5
    0.7
    ```

17. Alternative definition for the RS-PBE-P86 functional utilizing functionals from the Libxc library:
```
dft=rsdh
ssrGGA_X_PBE
ssrGGA_C_P86
0.5
0.7
```
18. To define third-order double hybrid functionals containing a perturbative third-order correction [285, 286], the `MP3` option can be used. For example, the XYG9[f2]@B44LYP functional of Ref. 285 can be specified as
```
dft=userd
4
0.56 B88
0.44 HFx
0.19 VWN5
0.81 LYP
7
0.0881 LDA
-0.0644 B88
0.9692 HFx
0.2730 VWN5
0.6447 MP2s
0.6467 MP2t
0.2761 MP3
```
19. Density-based basis-set correction, see keyword **`denscorr`**, using the built-in PBEc-based functional:
```
denscorr=2
calc=DF-CCSD(T)
dft=userd
1
1.00 HFx
1
1.0 ECMD_PBEc
```
20. Density-based basis-set correction, see keyword **`denscorr`**, utilizing correlation functionals from the Libxc library:
```
denscorr=2
calc=DF-CCSD(T)
dft=userd
1
1.00 HFx
1
1.0 ECMD_GGA_C_PW91
```
21. MP2-F12-based PBE0-2-F12 calculation:
```
calc=MP2-F12
dft=user
```

```
5
0.2063 PBEx
0.5000 PBEc
0.7937 HFx
0.5000 MP2s
0.5000 MP2t
```

`dhexc` This keyword can be used to select the wave function method for the evaluation of the second-order contributions in the case of excited-state double hybrid TD-DFT approaches. See Ref. 41 for further details.

Options:

`ADC(2)`, `CIS(D)`, `SOS-ADC(2)`, `SOS-CIS(D)`, `SCS-ADC(2)`, `SCS-CIS(D)` The corresponding methods will be used for the evaluation of second-order terms.

Default: `dhexc=ADC(2)`

Note: If a spin-scaled DH is employed, the corresponding spin-scaled ADC(2) or CIS(D) method will be used even if `ADC(2)` or `CIS(D)` is set. For a non-spin-scaled DH, if a spin-scaled approach is selected, the default spin-scaling parameters described in Ref. 41 will be set. To change the spin-scaling parameters use keywords `scsps` and `scspt`. For the CIS(D)-based ansatz, the keywords `scsps_t` and `scspt_t` can also be varied.

Examples:

1. To use the CIS(D)-based TD-B2PLYP approach of Ref. 147 for the lowest excited state of a molecule set:
```
calc=B2PLYP
nstate=2
dhexc=CIS(D)
```
2. To use the SOS-ADC(2)-based DSDPBEP86 approach of Ref. 41 for the lowest excited state of a molecule set:
```
calc=DSDPBEP86
nstate=2
dhexc=SOS-ADC(2)
```

`diag` Type of diagonalization algorithm used for the CI and LR-CC calculations.

Options:

`david` Standard Davidson diagonalization

`olsen` Another algorithm proposed by Olsen using only two expansion vectors (see Refs. 287, 288, and 3), useful for very large CI/LR-CC vectors

`follow` Davidson diagonalization with root-following, recommended for excited-state calculations if the initial guess is given manually or the calculation is restarted

Default: `diag=david`

Example: for root-following type `diag=follow` `docc` Specifies the number of doubly occupied orbitals in an MCSCF calculation. See also the description of keyword `mact`.

Options: The number of doubly occupied orbitals per irrep should be given in the following format:
`docc=`$< n_1 >, < n_2 >, \ldots, < n_{N_{ir}} >$
where $< n_i >$ is the number of doubly occupied orbitals in irrep $i$, and $N_{ir}$ is the number of irreps.

Default: There is no default, the occupation must be set in the case of an MCSCF calculation.

Examples:

1. Water, 2 × 2 CAS calculation, the active space includes an $A_1$ and a $B_1$ orbital:
```
docc=3,1,0,0
mact=1,0,1,0
```
2. Oxygen, 4 × 4 CAS calculation, the active space includes four orbitals of $B_{2g}$, $B_{3g}$, $B_{2u}$, and $B_{3u}$ symmetry. Note that the multiplicity and the symmetry of the state must also be specified, and, if the MCSCF calculation is restarted from HF MOs, the HF occupation should also be given.
```
occ=3,0,1,1,0,2,1,1/3,0,0,0,0,2,1,1
docc=3,0,0,0,0,2,0,0
mact=0,0,1,1,0,0,1,1
mult=3
symm=2
```

`domrad` Radius of atom domains for the local correlation method of Ref. 24 (`localcc=2013`). For each localized MO (LMO), using the Boughton–Pulay procedure [217], we assign those atoms to the LMO on which it is localized. Then, for each LMO an atom domain is constructed in two steps, the LMO is called the central LMO of the domain. In the first step, those atoms are included in the domain whose distance from the atoms assigned to the central LMO is smaller than `domrad`. In the second step, those LMOs are identified which are localized on the atoms selected in the first step, and the domain is extended to include all atoms assigned to these LMOs.

Options:

<*any positive real number*> In the first step of the construction of atom domains all atoms whose distance from the atoms assigned to the central LMO is smaller than this number (in bohr) will be included in the domain.

`inf` Infinite radius will be applied, i.e., there is only one atom domain including all atoms.

Default: `domrad=10.0`

Note: To use the local CC methods as defined in Ref. 22 set `domrad=inf`, that is, use only one atom domain including all atoms.

Example: to set a threshold of 12.0 bohr type `domrad=12.0` `drpaalg` Specifies the type of the algorithm for the solution of the dRPA equations or the calculation of SOS-MP2 energies. See Ref. 30 for more details.

Options:

`fit` The algorithm of Ref. 117 will be used, the fitting of integral lists will be performed before the dRPA iterations (SOS-MP2 calculation).

`nofit` The algorithm of Ref. 30 will be executed, the fitting of the integrals is not performed. This algorithm is efficient for large molecules.

`plasmon` The dRPA correlation energy is calculated using the plasmon formula.

`auto` The algorithm is automatically selected on the basis of the size of the molecule (canonical dRPA) or the HOMO-LUMO gap (local dRPA).

Notes:

1. For SOSEX calculations `drpaalg=fit` is the only option, which is forced by the program.
2. For canonical dRPA the algorithm using the plasmon formula scales as $N^6$, it is only competitive for smaller molecules but inefficient for bigger ones. It avoids, however, the problems of the other algorithms, that is, convergence problems and unphysical solutions. Thus, it is useful for testing.
3. For local dRPA `drpaalg=plasmon` is also linear scaling but typically 2- to 4-times slower than `drpaalg=fit`. It is advantageous for the aforementioned reasons. If `drpaalg=auto`, the plasmon formula-based algorithm is executed if the HOMO-LUMO gap is lower than 0.05 $\mathrm{E}_h$.

Default: `drpaalg=fit` and `drpaalg=auto` for canonical and local dRPA, respectively.

Example: to set the second option give `drpaalg=nofit`

`dual` Activates dual basis set calculations. For these calculations two basis sets must be specified: a smaller one by keyword `basis_sm` (see the description of the keyword) and a bigger basis defined by keyword `basis`. The energy evaluated with the bigger basis set is estimated from a small-basis calculation. See Ref. 39 for more details. This feature can be combined with dual auxiliary basis, dual metric, dual grid, etc. approximations, see keywords `grid_sm` and `dual_df`.

Options:

`off` No dual basis calculation.

`on` Dual basis set calculation for conventional SCF and correlated methods. First, an SCF calculation will be performed using the small basis set. Second, one iteration of a SCF calculation is carried out with the large basis, and the energy is extrapolated using a first-order formula. If a correlation calculation is requested, the orbitals obtained in the second SCF step will be used for that purpose.

`e1` Dual basis set embedding, Ansatz 1 of Ref. 39. A Huzinaga-embedding calculation is performed with the small basis set. The steps of the large-basis Huzinaga-embedding calculation are non-iterative. See also the description of keyword `embed`.

`e2` Dual basis set embedding, Ansatz 2 of Ref. 39. Similar to `e1`, but there is also an iterative step with the large basis.

Default: `dual=off`

Examples:

1. To perform a dual basis set PBE calculation with the cc-pVTZ and cc-pVDZ basis sets you need:
```
basis=cc-pVTZ
basis_sm=cc-pVDZ
dual=on
calc=PBE
```
2. Dual basis set PBE-in-LDA calculation with Ansatz 1 using the cc-pVTZ and cc-pVDZ basis sets as large and small basis set, respectively; atoms 1 to 5 are included in the embedded subsystem:
```
basis=cc-pVTZ
basis_sm=cc-pVDZ
calc=PBE
dual=e1
embed=Huzinaga
1-5
LDA
0
```
3. Dual basis set PBE-in-LDA calculation with Ansatz 1 using a mixed large basis (cc-pVTZ for atoms 1 to 5, cc-pVDZ otherwise) and the cc-pVDZ basis sets as the small basis set; atoms 1 to 5 are included in the embedded subsystem:
```
basis=embed
cc-pVTZ
cc-pVDZ
basis_sm=cc-pVDZ
calc=PBE
dual=e1
embed=Huzinaga
1-5
LDA
0
```

`dual_df` Set this keyword to perform dual auxiliary basis, dual metric DF-SCF, or dual grid COSX calculations [50]. For dual grid DFT calculations, see keyword `grid_sm`.

Options:

`dab` Performs dual auxiliary basis SCF calculations. To specify the small auxiliary basis, set the `dfbasis_scf_sm` keyword.

`dm` Performs a dual metric DF-SCF calculation.

`grid` Performs a dual grid COSX calculation. To specify the small angular integration grid, set the `agrid_pssp_sm` keyword.

`off` Turns off this feature.

Default: `dual_df=off` for all calculations except COSX, where `dual_df=grid` is the default. For more information about COSX calculations, see the keyword `pssp`.

Note: The `dm` option can be used in conjunction with the non-local direct SCF method only. The dual auxiliary basis algorithm can be applied to both local and non-local SCF calculations.

Examples:

1. Dual auxiliary basis calculation with local fitting with the cc-pVTZ AO basis set, the default large fitting basis set, and the cc-pVTZ-RI-JK-dual small fitting basis set:
```
basis=cc-pVTZ
dfbasis_scf_sm=cc-pVTZ-RI-JK-dual
calc=DF-HF
scfalg=locfit2
dual_df=dab
```

2. Dual metric calculation with the cc-pVTZ AO basis set and the default fitting basis set:
```
basis=cc-pVTZ
calc=DF-HF
scfalg=direct
dual_df=dm
```

`dyson` If set, Dyson orbitals for ADC(2)-based IP/EA states are written (see also Sect. 14).

Options:

`on` Dyson orbitals are saved to the corresponding files (see Notes). These files can be handled using the same programs as file `MOLDEN` (see keyword `molden`).

`off` Dyson orbitals are not calculated.

Note: For the Dyson orbitals, the corresponding `MOLDEN_DYSON_IP.`$n$ and `MOLDEN_DYSON_EA.`$n$ files are generated. The files are labeled according to the order of the IP/EA energies. We note that this option is also available for double-hybrid calculations.

Default: `dyson=off`

Example: if you need the Dyson orbitals, add `dyson=on`

`ecp` Specifies the effective core potential (ECP) used in all calculations. By default the ECPs are taken from the files named by the chemical symbol of the elements, which can be found in the `BASIS` directory created at the installation. The ECPs are stored in the format used by the CFOUR package. In addition to the ECPs provided by default, any ECP can be used by adding it to the corresponding files in the `BASIS` directory. Alternatively, you can also specify your own ECP in the file `GENBAS` which must be copied to the directory where MRCC is executed.

Options:

`none` No ECPs will be used.

`auto` The ECPs will be automatically selected: no ECP will be used for atoms with all-electron basis sets, while the ECP adequate for the basis set of the atom will be selected otherwise.

*<ECP label>* If the same ECP is used for all atoms, the label of the ECP can be given here.

`atomtype` If different ECPs are used or no ECP is used for particular atoms, but the atoms of the same type are treated in the same way, `ecp=atomtype` should be given, and the user must specify the ECP for each atomtype (for which an ECP is used) in the subsequent lines as *<atomic symbol>*`:`*<ECP label>* .

`special` In the general case, if different ECPs are used for each atom, then one should give `ecp=special` and specify the ECP for each atom in the subsequent lines by giving the label of the corresponding ECP (or `none` if no ECP is used for that atom) in the order the atoms appear at the specification of the geometry.

Notes:

1. By default, the following ECP are available for elements Na to Rn in Mrcc:
   - the LANL2DZ ECPs of Hay and Wadt [204–206]: LANL2DZ-ECP-10, LANL2DZ-ECP-18, LANL2DZ-ECP-28, LANL2DZ-ECP-36, LANL2DZ-ECP-46, LANL2DZ-ECP-60, LANL2DZ-ECP-68, LANL2DZ-ECP-78
   - the Stuttgart–Köln ECPs for the def2 basis sets [289–291]: def2-ECP-28, def2-ECP-46, def2-ECP-60
   - the Stuttgart–Köln multiconfiguration Dirac–Hartree–Fock-adjusted ECPs [208, 210, 211, 292–295]: MCDHF-ECP-10, MCDHF-ECP-28, MCDHF-ECP-60

   Please note that some of the above ECPs are not available for all elements.
2. If you need ECPs other than the default ones, you can, e.g., download them from the Basis Set Exchange [167–170]. Please choose format “`CFOUR`” when downloading the ECPs.
3. If you use your own ECPs, these must be copied to the end of the corresponding file in the `BASIS` directory. Alternatively, you can also create a file called `GENBAS` in the directory where Mrcc is executed, and then you should copy your ECPs to that file.
4. The labels of the ECPs must be identical to those used in the `BASIS/*` files (or the `GENBAS` file). For the default ECPs just type the name of the ECPs as given above, e.g., `LANL2DZ-ECP-10`, `def2-ECP-28`, etc. If you employ non-default ECPs, you can use any label.

Default: `ecp=auto`

Examples:

1. To use the MCDHF-ECP-10 pseudopotential for all atoms the input must include `ecp=MCDHF-ECP-10`
2. Consider the PbO molecule and use the def2-SVP basis set for both elements as well as the def2-ECP-60 pseudopotential for Pb. The following inputs are equivalent.

Input 1:

```
basis=def2-SVP
geom
Pb
O 1 R

R=1.921813
```

Input 2:

```
basis=def2-SVP
ecp=atomtype
Pb:def2-ECP-60
geom
Pb
O 1 R

R=1.921813
```

Input 3:

```
basis=def2-SVP
ecp=special
def2-ECP-60
none
geom
Pb
O 1 R

R=1.921813
```

`edisp` This keyword controls the calculation of empirical dispersion corrections for DFT and HF calculations using the DFT-D3 approach of Grimme and co-workers [227, 228]. The corrections are evaluated by the DFTD3 program of the latter authors, which is available at `https://www.chemie.uni-bonn.de/grimme/de/software/dft-d3` and interfaced to MRCC. You need to separately install this code and add the directory where the `dftd3` executable is located to your `PATH` environment variable.

Options:

- `off` No dispersion correction will be computed.
- `auto` The dispersion correction will be automatically evaluated to the KS or HF energy. Note that it is only possible for particular functionals listed in the description of keyword `dft` (and the HF method). For these methods, however, you can also turn on the calculations of the dispersion corrections by attaching the `-D3` postfix to the corresponding options, e.g., as `BLYP-D3`, `B3LYP-D3`, `B2PLYP-D3`, etc. (see the description of keyword `dft`).
- <*any options of the* DFTD3 *program*> You can directly give any options of the DFTD3

code. The options will be passed over to DFTD3 without any consistency check, the user should take care of the compatibility of these options with the calculation performed by MRCC. Note that the coordinate file name must not be specified here, the coordinates will be taken from the `COORD.xyz` file generated by MRCC.

Note: If `edisp=auto` or the `-D3` postfix is added to the corresponding options, the empirical dispersion correction is by default evaluated with the Becke and Johnson (BJ) damping function [228].

Default: `edisp=off`

Example:

1. to calculate the D3 dispersion correction including BJ damping to the B3LYP energy give `calc=B3LYP-D3`
2. to calculate the D3 dispersion correction to the B3LYP energy without the BJ damping the input should include:
   ```
   calc=B3LYP
   edisp=-func b3-lyp -zero
   ```

`edisp_embed` This keyword controls the calculation of empirical dispersion corrections for low-level methods (environment) in embedding calculations using the DFT-D3 approach of Grimme and co-workers [227, 228]. See the keywords `edisp` and `embed` for more details. Note that the empirical corrections are only calculated with the embedded atom list in the case of the subsystem calculations (see keyword `embed`). Note also that `edisp_embed` is not usable in the case of energy-based subsystem separation.

Options:

`off` No dispersion correction will be computed.

`auto` The dispersion energy correction will be automatically evaluated for the low-level theory of the embedding approach. Other considerations are same as for `edisp=auto`.

<*any options of the* DFTD3 *program*> You can directly give any options of the DFTD3 code. Other considerations are same as for `edisp=auto`. Note that the coordinates will be taken from the `COORD.xyz` and the `COORD_SUBSYSA.xyz` files generated by MRCC for the full system and the subsystem calculations, respectively.

Note: If `edisp_auto=auto` or the `-D3` postfix is added to the corresponding options, the empirical dispersion correction is by default evaluated with the Becke and Johnson (BJ) damping function [228].

Default: `edisp_embed=off`

Example:

1. To calculate the D3 dispersion correction (including BJ damping) for the interaction of the embedded subsystem and the environment in the case of a LNO-CCSD(T)-in-BLYP type embedding give
   `calc=LNO-CCSD(T)`

```
embed=huzinaga
1-2
BLYP-D3
0
```

2. To calculate the D3 dispersion correction to the BLYP energy without the BJ damping, the input should include:
   ```
   calc=LCCSD(T)
   edisp_embed=-func b-lyp -zero
   embed=huzinaga
   1-2
   BLYP
   0
   ```

`etemp` The electronic temperature in K at which the Fermi-smearing of the occupied orbitals is made in the case of `scfiguess=gfn1` and `scfiguess=gfn2`.

Options: <*any positive real number*>

Default: `temp=300.0`

Example: for 10000 K set `etemp=10000.0`

`embed` This keyword controls DFT embedding calculations. Currently, it is only available using density-fitting. Both restricted closed-shell determinants and unrestricted open-shell determinants are applicable for embedding calculations, however, only those open-shell cases are supported where the environmental subsystem has singlet multiplicity. See also keywords `corembed` and `oniom` for further embedding approaches.

Options:

`off` No embedding.

`Huzinaga` The Huzinaga-equation-based embedding approach [33, 39] will be used. The embedded atoms and the low-level DFT approach (or HF) used for the embedding must be specified and the number of embedded orbitals can be given in the subsequent lines as follows. The embedded atoms must be given by their serial numbers in the first line as $< n_1 >$,$< n_2 >$,...,$< n_k >$-$< n_l >$,..., where $n_i$'s are the serial numbers of the atoms. Serial numbers separated by dash mean that $< n_k >$ through $< n_l >$ are embedded atoms. Note that the numbering of the atoms must be identical to that used in the Z-matrix or Cartesian coordinate specification, but dummy atoms must be excluded. The low-level DFT (LDA, GGA, or hybrid) or HF approach must be specified in the second line using the corresponding option of keyword `dft` (for HF simply `HF`). Note that the `user` option of keyword `dft` is also supported. The high-level method (any DFT, HF, or any correlation method) for the active region should be specified by the keyword `calc` (or keywords `calc` and `dft` if a KS reference is used in a correlation calculation). In the third line an integer, $i$, should be given which is the number of the embedded orbitals or zero. If zero is given, then the number of embedded orbitals are determined automatically. In addition, the algorithm for the selection of the orbitals can also be given in the third line after the integer. The options are `amul`, `nmul`, `spad`, `nspad`, `ecore`, `bopu`, and `serialno`. The `amul` and `nmul` mean the Mulliken population-based orbital selection scheme of Refs. 33 and 39. For `amul`, you can also specify the Mulliken population threshold (see the appendix of Ref. 39), the default threshold is 0.3. The `nmul` scheme selects $i$ pieces of active orbitals with the largest Mulliken populations. The `spad` and `nspad` techniques employ the orbital selection scheme of Ref. 296, where the active orbitals are selected based on the difference of the MO populations. Note that `spad` does not require a threshold for the selection, algorithm `nspad` selects $i$ orbitals which have the largest eigenvalues of the SPADE projector, and both schemes require `orbloce=spade`. The energy-based `ecore` algorithm is also supported, which does not require orbital localization and selects the high-energy orbitals as active orbitals: the `ecore` algorithm automatically selects the valence orbitals as active, while if $i$ is nonzero, then $i$ pieces of orbitals with the largest orbital energies are selected. Note that in the case of the `ecore` algorithm, the orbitals are not localized and the embedded atom list is not used in the calculation. `bopu` means the Boughton–Pulay algorithm-based scheme, see Ref. 33 for more details. For the default MO completeness criteria of the `bopu` algorithm, see keywords `bp_subsyso` and `bp_subsysv`. Lastly, if `serialno` is specified, the active orbitals are selected manually by giving the appropriate serial numbers of the orbitals after the keyword `serialno`. Note that the ordering of the orbitals are based on their orbital energies and this scheme does not require a list of active atoms. The default automatic selection scheme is `spad` if the `orbloce=spade`, and `amul` otherwise. If $i$ is not zero, `nspad` is the default scheme when `orbloce=spade`, and `nmul` is the default algorithm otherwise. Note also that the `bopu` scheme is the only selection approach that requires the execution of the Boughton–Pulay algorithm. In the other cases, it can be omitted for efficiency considerations, see keyword `subsys_bopu` for more details. Finally, the embedding calculations can also be executed with selected active virtual orbitals if `orblocv` is defined by using `orbloce=special`. The orbital selection scheme of the virtual orbitals is specified in the fifth line, and all the previously described schemes are supported with a single modification: the `ecore` algorithm does not have an automatic selection mode and it will select the $i$ number of virtual orbitals with the lowest eigenvalues. In this case, a modified Huzinaga operator is constructed which includes the projector of the inactive virtual orbitals as well. In addition to the virtual orbital localization procedures, the projected atomic orbitals-based (PAOs) virtual space separation approach is also supported [297]. The PAO-based algorithm constructs PAOs utilizing the low-level supersystem density matrix, and subsequently, screening steps are applied to the redundant PAOs using the `pao_tol` and `pao_subsys_tol` truncation parameters which results in the nonredundant PAOs of the active subsystem. The projector of the virtual environment orbitals is constructed by subtracting the projector of the occupied space and the projector of the active virtual space from the identity matrix, and finally, the resulting projector is included in the modified Huzinaga operator. Note that the subsequent correlation calculations will only use the active virtual subspace. See also examples below.

`project` The projector-based embedding approach of Manby and co-workers [82] will be used. The embedded atoms and the low-level DFT approach can be specified as described above.

Default: `embed=off`

Examples:

1. CCSD(T)-in-PBE embedding with the Huzinaga-equation-based approach, atoms 1 and 2 are included in the embedded region, the number of the embedded orbitals is determined automatically, the `amul` algorithm is used for the selection of the orbitals with the default threshold, and the Pipek–Mezey localization scheme is utilized when the system is split up:

   ```
   calc=DF-CCSD(T)
   orbloce=pm
   embed=huzinaga
   1-2
   PBE
   0
   ```

2. The same as example 1, but the `amul` algorithm is used with a threshold of 0.25. Here, the `amul` selection scheme is enforced regardless of the orbital localization scheme and the Boughton–Pulay algorithm is not executed:

   ```
   calc=DF-CCSD(T)
   subsys_bopu=off
   embed=huzinaga
   1-2
   PBE
   0 amul 0.25
   ```

3. The same as example 1, but the `bopu` algorithm is used with 0.93 MO completeness criteria:

   ```
   calc=DF-CCSD(T)
   bp_subsyso=0.93
   embed=huzinaga
   1-2
   PBE
   0 bopu
   ```

4. Same as example 1, but a mixed basis set is used, cc-pVDZ for the environment and cc-pVTZ for the embedded subsystem:

   ```
   calc=DF-CCSD(T)
   basis=embed
   cc-pVDZ
   cc-pVTZ
   orbloce=pm
   embed=huzinaga
   1-2
   PBE
   0
   ```

5. dRPA@PBE-in-PBE embedding with the Huzinaga-equation-based approach, atoms 1, 2, 3, and 5 as well as 10 orbitals are included in the active region. Here, the `nmul` scheme is applied to the selection of the orbitals:
   ```
   calc=dRPA
   dft=PBE
   orbloce=pm
   embed=huzinaga
   1-3,5
   PBE
   10
   ```
6. B3LYP-in-BLYP embedding with the Huzinaga-equation-based approach, the number of the embedded orbitals will be the same as the number of valance orbitals. Note that the active atoms are not specified because of the `ecore` algorithm, and a blank line is used in the second line after the keyword `embed`:
   ```
   calc=B3LYP
   embed=huzinaga

   BLYP
   0 ecore
   ```
7. B3LYP-in-BLYP embedding with the Huzinaga-equation-based approach, the number of the embedded orbitals will be the 4 MOs with the highest orbital energies:
   ```
   calc=B3LYP
   embed=huzinaga

   BLYP
   4 ecore
   ```
8. B3LYP-in-BLYP embedding with the Huzinaga-equation-based approach, orbitals with the 1, 2, 3, 4, and 13 serial numbers are selected manually as active orbitals. Again, note that the list of active atoms are not required in this case:
   ```
   calc=B3LYP
   embed=huzinaga

   BLYP
   0 serialno 1,2-4,13
   ```
9. CCSD(T)-in-PBE embedding with the Huzinaga-equation-based approach, atoms 1 and 2 are included in the embedded region. The `spade` algorithm is used for the automatic selection of the active occupied orbitals, while the PAO-based approach is applied to construct the projector of the virtual environment. After the PAOs of the active subsystem are obtained, 4 active virtual orbitals are selected by the `nmul` scheme. Note that the embedded occupied MOs of the second SCF run are only allowed to mix with 4 virtual MOs, and subsequently, the CCSD(T) calculation is carried out with the active occupied and active virtual MOs:
   ```
   calc=DF-CCSD(T)
   ```

```
orbloce=special
spade
spade
pao-subsys
embed=huzinaga
1-2
PBE
0
4 nmul
```

10. Same as example 9, but in this case, the SPADE localization technique is applied to obtain the projector of the virtual subspace.
    ```
    calc=DF-CCSD(T)
    orbloce=special
    spade
    spade
    spade
    embed=huzinaga
    1-2
    PBE
    0
    4 nmul
    ```
11. Same as example 10, but after the embedded HF solution is obtained, projected atomic orbitals (PAO) are constructed for the correlation calculation. Note that the atomic orbitals of atoms 1 and 2 will be used to calculate the norm of the PAOs, and those PAOs that have a norm lower than `pao_subsys_tol` will be omitted in the subsequent correlation calculation. Note the blank line below `orblocv`.
    ```
    calc=DF-CCSD(T)
    orblocv=pao-subsys

    orbloce=special
    spade
    spade
    spade
    embed=huzinaga
    1-2
    PBE
    0
    4 nmul
    ```
12. Same as example 11, but this time, the norm of the PAOs are calculated using an extended number of atoms thus generating more virtual orbitals.
    ```
    calc=DF-CCSD(T)
    orblocv=pao-subsys
    1-2,8-10
    orbloce=special
    ```

```
spade
spade
spade
embed=huzinaga
1-2
PBE
0
4 nmul
```

13. Same as example 12, except that all virtual orbitals are used in the high-level SCF run, and only the CCSD(T) calculation utilizes a reduced number of virtual orbitals. Again, note the blank line below `orblocv`, which can be used to override the atom list which is used for the norm calculation of PAOs. Here, the atom list is not overridden.

```
calc=DF-CCSD(T)
orblocv=pao-subsys

orbloce=special
spade
spade
off
embed=huzinaga
1-2
PBE
0
```

`epert` Use this option to add an external perturbation to the Hamiltonian, e.g., an external electric dipole field.

Options:

`none` No perturbations are added.

*<any integer in the [0,9] interval>* the number of the operators added to the Hamiltonian. The operators and the corresponding coefficients (in a.u.) should be specified in the subsequent lines as follows:
*<operator 1> <coefficient 1>*
*<operator 2> <coefficient 2>*
*<operator 3> <coefficient 3>*
...
where the operator can be `x`, `y`, `z`, `xx`, `yy`, `zz`, `xy`, `xz`, `yz`, `xxx`, `xxy`, `xxz`, `xyy`, `xyz`, `xzz`, `yyy`, `yyz`, `yzz`, `zzz`.

Note: The symmetry of the perturbation is not taken care of automatically. If the perturbation lowers the symmetry of the system, you must change the computational point group (keyword `cmpgrp`) or turn off symmetry (`symm=off`).

Default: `epert=none`

Example: to add the $\hat{y}$ and $\hat{z}$ dipole length operators to the Hamiltonian with coefficients 0.01 and 0.001 a.u., respectively, the input should include the following lines

```
epert=2
y 0.01
z 0.001
```

`eps` Threshold for the cumulative populations of MP2 natural orbitals (NOs) or optimized virtual orbitals (OVOs), to be used together with keyword `ovirt`. The cumulative population for an MO is calculated by summing up the occupation number of that particular MO and all the MOs with larger occupation numbers, and then this number is divided by the number of electrons. If unset, keyword `lnoepsv` is used as the threshold for MP2 NOs. The use of keyword `lnoepsv` is preferred instead of keyword `eps`. See Ref. 21 for more details.

Options:

*<any real number in the [0,1] interval>* Virtual orbitals with cumulative populations of higher than this number will be dropped.

Default: None. By default keyword `lnoepsv` is used.

Notes:

1. This keyword can not be used with explicitly correlated calculations.
2. If `eps` is specified, the spatial symmetry is not used in correlation calculations.

Example: to set a threshold of 0.95 type `eps=0.95`

`espcharge` Select the desired method for calculating atomic charges based on the electrostatic potential around the molecule.

Options:

`off` No atomic charge calculation based on electrostatic potential.

`chelpg` Calculate CHELPG charges [298].

`mk` Calculate Merz–Singh–Kollman charges [299].

Default: `espcharge=off`

Notes:

1. If `espcharge`$\neq$`off` and `dens=0`, then it will be changed to `dens=1`.
2. Atomic radii used for the determination of the charges are listed in Table 5. The values are based on the original articles and the MULTIWFN implementation. For atoms not shown, the UFF radii divided by 1.2 are utilized.

Example: to calculate CHELPG charges use `espcharge=chelpg`

`excrad` Sets the radius of local fitting domains for the exchange contribution in direct density-fitting SCF calculations [31, 158]. In direct DF-SCF calculations, in each iteration step, the MOs are localized. For each localized MO Löwdin atomic charges are computed, and all atoms are selected which have a charge greater than 0.05. All further atoms will be included in the fitting domain of the MO for which the electron repulsion integrals including the corresponding AOs and the basis functions residing on the atoms selected in the first step are estimated to be greater than a threshold.

Options:

Table 5: Atomic radii in Å used for the determination of atomic charges based on electrostatic potential.

| Atomic symbol | CHELPG | Merz–Singh–Kollman |
|---|---|---|
| H | 1.45 | 1.20 |
| He | 1.45 | 1.20 |
| Li | 1.50 | 1.37 |
| Be | 1.50 | 1.45 |
| B | 1.50 | 1.45 |
| C | 1.50 | 1.50 |
| N | 1.70 | 1.50 |
| O | 1.70 | 1.40 |
| F | 1.70 | 1.35 |
| Ne | 1.70 | 1.30 |
| Na | 2.00 | 1.57 |
| Mg | 2.00 | 1.65 |
| Al | 2.00 | 1.65 |
| Si | 2.00 | 1.80 |
| P | 2.00 | 1.80 |
| S | 2.00 | 1.75 |
| Cl | 2.00 | 1.70 |
| Ar | 2.00 | – |

<*any positive real number*> Threshold for the integrals (in $E_h$).

`0` A threshold of zero will be applied, i.e., a conventional direct DF-SCF calculation will be executed.

Notes:

1. Local fitting domains are available for RHF, UHF, ROHF, RKS, UKS, and ROKS calculations.
2. For average organic molecules with localized electronic structure `excrad=1.0` is a good choice. For more complicated systems other thresholds may be necessary. For `excrad=1.0` `excrad_fin` is set to $10^{-3}$, which is fine with basis sets excluding diffuse functions. For basis sets with diffuse functions `excrad_fin=1e-5` or tighter is recommended.

Default: `excrad=1.0` if `scfalg=locfit1` or `scfalg=locfit2`, `excrad=0` otherwise

Example: to set a threshold of 0.1 $E_h$ type `excrad=0.1`

`excrad_f12` During the CABS correction calculation, the Fock matrix is expanded to include the complementary auxiliary basis set. `excrad_f12` specifies the radius of local fitting domains for the exchange contribution in this step if the local DF algorithm is used for the construction of the Fock matrix.

Options: See the description of keyword `excrad`.

Default: `excrad=5.0` if `scfalg=locfit1` or `scfalg=locfit2`, `excrad=0` otherwise

Example: to set a threshold of 1.0 $E_h$ type `excrad=1.0`

`excrad_fin` In density-fitting SCF calculations, if `excrad` and `excrad_fin` differ, an extra iteration is performed to get an accurate SCF energy. `excrad_fin` specifies the radius of local fitting domains for the exchange contribution in this iteration step. See also notes for keyword `excrad`.

Options: See the description of keyword `excrad`.

Default: `excrad_fin=excrad`/1000

Example: to avoid the use of local fitting domains in the extra iteration step give `excrad_fin=0.0`

`fitting` Specifies the fitting metric for direct DF-SCF calculations.

Options:

`Coulomb` Coulomb-metric DF-SCF calculation.

`Overlap` Overlap-metric DF-SCF calculation.

Default: `fitting=Coulomb`

Example: to use the overlap metric, set `fitting=Overlap`

`fmm` Controls the multipole method for three-center Coulomb integrals for density fitting SCF calculations. See also keyword `fmmord`.

Options:

`off` No multipole approximation is used for the integrals.

`on` The exchange contribution will be evaluated using multipole approximations.

`Coulomb` Both the Coulomb and the exchange contributions will be evaluated using multipole approximations.

Note: The multipole method can be employed either with or without local fitting approximations (see keyword `scfalg`). If no local fitting is used, the speedup is quite good, the computation time can roughly be halved. With local fitting approximations, the speedup is moderate.

Default: `fmm=off`

Example: To use the multipole method to speed up a DF-HF calculation, set `fmm=on`

`fmmord` The keyword controls the order of the multipole approximation if the multipole method is used for three-center Coulomb integrals (see also keyword `fmm`). If the multipole approximation is invoked to calculate an integral of the product of two Gaussians, the contribution of multipole moments up to order $l_1 + l_2 +$ `fmmord` will be considered, where $l_1$ and $l_2$ are the angular momentum quantum numbers of the Gaussians.

Options: <*any positive integer*>

Default: `fmmord=8`

Example: for lower accuracy but increased speed, give `fmmord=6` `freq` Requests harmonic vibrational frequency calculation (numerical). Ideal gas thermodynamic properties will also be evaluated in the rigid-rotor harmonic-oscillator approximation. If geometry optimization is also carried out, i.e., `gopt` $\neq$ `off`, the frequencies are calculated at the optimized geometry.

Options: `on` or `off`

Note: You should also set this keyword if you are interested in thermodynamic properties of atoms.

Default: `freq=off`

Example: for a frequency calculation set `freq=on`

`gamma` Exponent for the Slater-type correlation factor ($\gamma$) used in explicitly correlated calculations [112, 300].

Options:

*<any positive real number>* Value of the correlation factor in bohr$^{-1}$.

Note: In MRCC, six Gaussians are used to fit the Slater-type correlation factor. The exponents and coefficients were taken from Ref. 300.

Default: `gamma=0.9` for cc-pVDZ-F12, `gamma=1.1` for cc-pVQZ-F12 or aug-cc-pVTZ, `gamma=1.4` for aug-cc-pVQZ or aug-cc-pV5Z, `gamma=1.0` otherwise.

Example: to set an exponent of 1.3 bohr$^{-1}$ give `gamma=1.3`

`gauss` Specifies whether spherical harmonic or Cartesian Gaussian basis functions will be used.

Options:

`spher` Spherical harmonic Gaussians will be used

`cart` Cartesian Gaussians will be used

Notes:

1. For calculations using the density fitting (DF) approximation, if `intalg=os` or `intalg=auto`, the Coulomb integrals are evaluated by algorithms [34, 301] which only enable the use of spherical harmonic Gaussians. Consequently, Cartesian Gaussians are only available with `intalg=rys` in DF calculations (see the description of keyword `intalg`).
2. The derivative integrals are evaluated by the solid-harmonic Hermite scheme [302] (see the description of keyword `intalg`), consequently, differentiated integrals, and thus energy derivatives cannot be evaluated with Cartesian Gaussian basis sets.

Default: `gauss=spher`

Example: for Cartesian Gaussians the user should set `gauss=cart`

`geom` Specifies the format of molecular geometry. The geometry must be given in the corresponding format in the subsequent lines.

Options:

`zmat` Usual Z-matrix format. In the Z-matrix the geometrical parameters can only be specified as variables, and the variables must be defined after the matrix, following a blank line. Another blank line is required after the variables. This Z-matrix format is compatible to that of CFOUR and nearly compatible to that for GAUSSIAN and MOLPRO. Z-matrices can be generated by MOLDEN (see also Sect. 14.1), then the GAUSSIAN-style Z-matrix format must be chosen. The symbol for dummy atoms is “X”.

`xyz` Cartesian coordinates in xyz format, that is, the number of atoms, a blank line, then for each atom the atomic symbol or atomic number and the $x$, $y$, and $z$ components of Cartesian coordinates. Cartesian coordinates in xyz format can also be generated by MOLDEN (see also Sect. 14.1).

`tmol` Cartesian coordinates in a format similar to that used by the TURBOMOLE package, that is, the number of atoms, a blank line, then for each atom the $x$, $y$, and $z$ components of Cartesian coordinates and the atomic symbol or atomic number.

`mol` Cartesian coordinates and connectivity in .mol format, that is, the number of atoms and number of bonds in the first line, then for each atom the $x$, $y$, and $z$ components of Cartesian coordinates and the atomic symbol, then for each bond the serial number of the atoms connected by the bond and the type of the bond (1 for single bond, etc.). This geometry specifications is needed if the specified method requires the connectivity.

Note: For the use of ghost atoms see the description of keyword `ghost`.

Default: `geom=zmat`, which is equivalent to `geom`, i.e., if it is not specified whether the geometry is supplied in Z-matrix format or in other formats, Z-matrix format is supposed. Nevertheless, the coordinates must be given in the subsequent lines in any case.

Examples: the following five geometry inputs for $H_2O_2$ are equivalent

1. Z-matrix format, bond lengths in Å:
```
geom
H
O 1 R1
O 2 R2 1 A
H 3 R1 2 A 1 D

R1=0.967
R2=1.456
A=102.32
D=115.89
```

2. xyz format, coordinates in bohr, atoms are specified by atomic symbols:
```
unit=bohr
geom=xyz
4
```

```
H 0.00000000 0.00000000 0.00000000
O 1.82736517 0.00000000 0.00000000
O 2.41444411 2.68807873 0.00000000
H 3.25922198 2.90267673 1.60610134
```

3. xyz format, coordinates in bohr, atoms are specified by atomic numbers:

```
unit=bohr
geom=xyz
4

1 0.00000000 0.00000000 0.00000000
8 1.82736517 0.00000000 0.00000000
8 2.41444411 2.68807873 0.00000000
1 3.25922198 2.90267673 1.60610134
```

4. TURBOMOLE format, coordinates in bohr, atoms are specified by atomic symbols:

```
unit=bohr
geom=tmol
4

0.00000000 0.00000000 0.00000000 H
1.82736517 0.00000000 0.00000000 O
2.41444411 2.68807873 0.00000000 O
3.25922198 2.90267673 1.60610134 H
```

4. .mol format, coordinates in bohr:

```
unit=bohr
geom=mol
4 3
0.00000000 0.00000000 0.00000000 H
1.82736517 0.00000000 0.00000000 O
2.41444411 2.68807873 0.00000000 O
3.25922198 2.90267673 1.60610134 H
1 2 1
2 3 1
3 4 1
```

`ghost` Ghost atoms can be specified using this keyword, e.g., for the purpose of basis set superposition error (BSSE) calculations with the counterpoise method.

Options:

`none` There are no ghost atoms.

`serialno` Using this option one can select the ghost atoms specifying their serial numbers. The latter should be given in the subsequent line as $< n_1 >$,$< n_2 >$,...,$< n_k >$-$< n_l >$,..., where $n_i$'s are the serial numbers of the atoms.

Serial numbers separated by dash mean that $< n_k >$ through $< n_l >$ are ghost atoms. Note that the numbering of the atoms must be identical to that used in the Z-matrix or Cartesian coordinate specification, but dummy atoms must be excluded.

Default: `ghost=none`

Examples:

1. Rectangular HF dimer, the atoms of the second HF molecule are ghost atoms:

```
geom
H
F 1 R1
H 2 R2 1 A
F 3 R1 2 A 1 D

R1=0.98000000
R2=2.00000000
A=90.00000000
D=0.000000000

ghost=serialno
3-4
```

2. Ammonia, the third hydrogen is a ghost atom (note that the serial number of the hydrogen is 4 instead of 5 because of the dummy atom):

```
geom
X
N 1 R
H 2 NH 1 AL
H 2 NH 1 AL 3 A
H 2 NH 1 AL 3 B

R=1.00000000
NH=1.01000000
AL=115.40000000
A=120.00000000
B=-120.00000000

ghost=serialno
4
```

`gopt` Requests geometry optimization. Currently only the full geometry optimization is supported, geometrical parameters cannot be frozen.

Options:

`off` no geometry optimization.

`full` full geometry optimization.

Notes:

1. The coordinates in the `MINP` file are replaced by the converged ones at the end of the geometry optimization. The initial `MINP` file is saved as `MINP.init`.
2. The Abelian symmetry of the molecule is utilized at the calculation of gradients and update of the coordinates, thus, the computational point group is preserved during the optimization.

Default: `gopt=off`

Example: to carry out a full geometry optimization set `gopt=full`

`grdens` This keyword is useful for the analysis of HF, KS, or correlated (MP2, CI, CC, ...) one-electron density and its derivatives. The one-electron density, its gradient, and Laplacian will be calculated on a grid used for DFT calculations (see keywords `agrid` and `rgrid`) and saved for external use. In the case of correlated calculations the densities are evaluated using the relaxed density matrices.

Options:

`off` Densities are not evaluated.

`on` The one-electron density and its derivatives are calculated in the grid points. These values together with the grid are written to the unformatted Fortran file `DENSITY`. If a correlation calculation is performed the densities calculated with the correlation method are stored in the `DENSITY` file, while the SCF densities are saved to the file `DENSITY.SCF`. For restricted orbitals the files use the following format:

$$\begin{array}{l} N \\ \mathbf{r}_1\ w_1\ \rho(\mathbf{r}_1)\ \nabla\rho(\mathbf{r}_1)\ \nabla^2\rho(\mathbf{r}_1) \\ \mathbf{r}_2\ w_2\ \rho(\mathbf{r}_2)\ \nabla\rho(\mathbf{r}_2)\ \nabla^2\rho(\mathbf{r}_2) \\ \qquad\qquad\vdots \\ \mathbf{r}_N\ w_N\ \rho(\mathbf{r}_N)\ \nabla\rho(\mathbf{r}_N)\ \nabla^2\rho(\mathbf{r}_N) \end{array}$$

where $N$ is the number of grid points, $\mathbf{r}_i = (x_i,\ y_i,\ z_i)$ is the Cartesian coordinate of grid point $i$ with $w_i$ as the corresponding weight, and $\rho(\mathbf{r}_i)$ is the density in point $i$. For unrestricted calculations the corresponding $\alpha$ and $\beta$ quantities are stored separately, and the lines of the files change as
$\mathbf{r}_i\ w_i\ \rho_\alpha(\mathbf{r}_i)\ \rho_\beta(\mathbf{r}_i)\ \nabla\rho_\alpha(\mathbf{r}_i)\ \nabla\rho_\beta(\mathbf{r}_i)\ \nabla^2\rho_\alpha(\mathbf{r}_i)\ \nabla^2\rho_\beta(\mathbf{r}_i)$

Default: `grdens=off`

Example: to save the densities give `grdens=on`

`grid` Specifies the integration grid for DFT calculations. The grid construction follows the design principles of Becke [174] in any case.

Options:

`auto` An automatically constructed adaptive integration grid will be used. The adaptive scheme is motivated by Ref. 303. The important difference is that the grids are optimized for each atom separately to avoid discontinuous potential energy surfaces. Both the angular and the radial grids are selected automatically to ensure the accuracy of the angular and the radial integrations as described at keywords `agrid` and `rgrid`. See also the description of keyword `grtol`. The smoothing function for the Voronoi polyhedra are adopted from Ref. 304 with $m_\mu = 10$.

`TA<n>` Grid no. *n* of Treutler–Ahlrichs [175] using the Log3 radial quadrature of Mura and Knowles [305].

Default: `grid=TA3` for KS-DFT calculations with non-hybrid LDA and GGA functionals, `grid=auto` otherwise

Example: to use grid 2 of Treutler–Ahlrichs, set `grid=TA2`

`grid_sm` If this keyword is set and different from keyword `grid`, a dual grid DFT calculation will be carried out. The SCF iterations will be performed using the grid specified by `grid_sm`, and an extra iteration will be carried out with a larger grid set by `grid`. A first-order correction is also evaluated [50]. This can also be combined with dual (auxiliary) basis, dual metric, etc. approximations, see keywords `dual` and `dual_df`. Additionally, this keyword controls the grid size used for the DBBSC calculations.

Options: see keyword `grid`.

Default: `grid=TA1` for DBBSC calculations using local domain approximations, `grid_sm=grid` otherwise

Examples:

1. To use grid 1 and 3 of Treutler–Ahlrichs in the SCF cycles and in the final iteration, respectively, (recommended for KS-DFT calculations with non-hybrid functionals) set:
```
grid_sm=TA1
grid=TA3
```
2. To use grid 1 of Treutler–Ahlrichs and a Coulomb fitting basis in the SCF cycles and the default adaptive grid and a Coulomb-exchange fitting basis in the final iteration (recommended for DFT calculations with a hybrid functional; note that for the latter, you do not need to set `dfbasis_scf` with def2-type AO basis sets, because this is the default) set:
```
grid_sm=TA1
dual_df=DAB
dfbasis_scf_sm=def2-QZVPP-RI-J
dfbasis_scf=def2-QZVPP-RI-JK
```

`grtol` The keyword controls the fineness of the angular and radial integration grids employed in DFT calculations. The tolerance for the accuracy of angular integrals will be $10^{-\texttt{grtol}}$, while the number of radial grid points increases linearly with `grtol`. See also the description of keywords `agrid` and `rgrid`.

Options: <*any positive integer*>

Note: meta-GGA functionals or molecules with special bonding characteristics may require larger integration grids, and it is recommended to run test calculations to decide if the default value of `grtol` is sufficient.

Default: `grtol=10`

Example: for a fine integration grid give `grtol=12`

`gtol` Threshold for automatic point group recognition. Two atoms will be considered symmetry-equivalent if the difference in any component of their Cartesian coordinates after the symmetry operation is less than $10^{-\texttt{gtol}}$ bohr.

Options: <*any integer*>

Default: `gtol=7`

Example: for a tolerance of $10^{-4}$ bohr give `gtol=4`

`guido_ct` Use this keyword to calculate the Guido-type $\Delta\mathbf{r}$ charge-transfer descriptor (see Ref. 306).

Options:

`on` Calculate the corresponding descriptor.

`off` The calculation of the descriptor is turned off.

Default: `guido_ct=off`

Example: To calculate the descriptor use `guido_ct=on`

`hamilton` Specifies what type of Hamiltonian is used in relativistic calculations. This keyword has only effect if `iface=Dirac`.

Options:

`X2Cmmf` exact 2-component molecular-mean-field Hamiltonian [307]

`DC` other types of relativistic Hamiltonians such as the full Dirac–Coulomb Hamiltonian or the exact 2-component Hamiltonian

Default: `hamilton=DC`

Example: if you use the exact 2-component molecular-mean-field Hamiltonian, set `hamilton=X2Cmmf`

`iface` Specifies whether MRCC is used together with another program system. In this case the transformed MO integrals are calculated by that program and not by MRCC. See Sect. 5 for the description of various interfaces.

Options:

`none` Transformed MO integrals are calculated by MRCC.

`Cfour` MRCC is interfaced to CFOUR.

`Columbus` MRCC is interfaced to COLUMBUS.

`Dirac` MRCC is interfaced to DIRAC.

`Molpro` MRCC is interfaced to MOLPRO.

Notes:

1. If you use Mrcc together with Cfour or Molpro, you do not need to use this keyword. The Mrcc input file is automatically written and Mrcc is automatically called by these program systems. The user is not required to write the Mrcc input file, most of the features of Mrcc can be controlled from the input files of these programs. With Cfour the user has the option to turn off the automatic construction of the Mrcc input file by giving `INPUT_MRCC=OFF` in the Cfour input file `ZMAT`. In the latter case one should use this keyword.
2. If you use Mrcc together with Columbus or Dirac, this keyword must be always given.

Default: `iface=none`, that is, all calculations will be performed by Mrcc.

Example: to carry out four-component relativistic calculations using the Dirac interface give `iface=Dirac`

`incore` Specifies whether the three-center Coulomb integrals should be stored in main memory during the solution of SCF or coupled-perturbed SCF equations (MP2 only). This option works only with `scfalg=disk`. Please note that Mrcc does not check if the memory is enough for storing the three-center integrals before running the `scf` program.

Options:

`on` The three-center Coulomb integrals are stored in the main memory during SCF calculations.

`off` The three-center Coulomb integrals are not stored in the main memory.

Default: `incore=off`

`intalg` Specifies the algorithm used for the evaluation of two-electron integrals over primitive Gaussian-type orbitals.

Options:

`os` The $(\mathbf{e0}|\mathbf{f0})$ integrals are evaluated by the Obara–Saika procedure using the vertical and transfer recurrence relations [308, 309].

`rys` The $(\mathbf{e0}|\mathbf{f0})$ integrals are evaluated by the Rys quadrature scheme [309–311].

`auto` Depending on the angular momenta the program automatically determines which of the two algorithms is executed. For integrals of low angular momentum functions the Rys procedure is used, while the Obara–Saika algorithm is executed otherwise.

`herm` The integrals over contracted Gaussians are evaluated by the solid-harmonic Hermite scheme of Reine *et al.* [302].

Notes:

1. For calculations using the density fitting (DF) approximation `intalg=auto` is equivalent `intalg=os` since the Obara–Saika algorithm is more efficient for any integrals.
2. For DF methods option `herm` is not available.

3. For DF methods, if `intalg=os` or `intalg=auto` the Coulomb integrals are evaluated by the algorithm of Ref. 34, which only enables the use of spherical harmonic Gaussians. Consequently, Cartesian Gaussians are only available with `intalg=rys` in DF calculations (see the description of keyword `gauss`).
4. The derivative integrals are evaluated by the solid-harmonic Hermite scheme even if another option is used for the undifferentiated integrals. Consequently, differentiated integrals, and thus energy derivatives cannot be evaluated with Cartesian Gaussian basis sets.

Default: `intalg=auto`

Example: to use the Obara–Saika scheme for all angular momenta add `intalg=os`

`ip_ea` Specifies whether standard excitation energies, ionization potentials, or electron affinities are calculated. This option is available at the ADC(2) and CIS(D) levels, as well as for the corresponding TDDFT functionals [65].

Options:

`off` Standard excited-state calculations are carried out.

`ip` Ionization potentials are calculated.

`ea` Electron affinities are calculated.

Default: `ip_ea=off`

Example: for ionization potentials, the user should set `ip_ea=ip`

`itol` Threshold for integral calculation. Integrals less than $10^{-\texttt{itol}}$ $E_h$ will be neglected.

Options: <*any integer*>

Default: `itol`=max(10, `scftol`+4, `scfdtol`), but `itol` is changed to `itol+1` if basis functions are dropped because of linear dependence (see keyword `ovltol`)

Example: for an accuracy of $10^{-15}$ $E_h$ one must give `itol=15`

`laptol` Specifies the accuracy of the numerical Laplace transform in the (T) correlation energy term of local CCSD(T) calculations with `localcc`≥`2016`. See also the description of keyword `talg` and Ref. 36.

Options:

<*any positive real number*> The (T) energy denominator will be approximated using its numerical Laplace transform. The number of quadrature points ($n_\mathrm{q}$), and hence the accuracy is determined by this number. The minimum value of $n_\mathrm{q}$ is also set by this number as $n_\mathrm{q} > |\log_{10}(\texttt{laptol})|$.

Default: `laptol=`$10^{-2}$, but `laptol=`$10^{-3}$ is set if `lcorthr=tight`. See also the description of `lcorthr` for further details.

Example: to use a threshold of $10^{-3}$ type `laptol=1e-3`

`lccoporder` Specifies the order of operations in LNO-CC calculations.

Options:

`lccfirst` The integral transformation steps yielding the LNO integral lists of a LIS are immediately followed by the CC calculation in the same LIS. The three-center LNO integrals are stored in memory and passed directly to the `ccsd` program (unless `ccsdalg=disk` was requested). Hence, in combination with the default `ccsdalg=dfdirect`, the LNO integrals can be kept in memory throughout the entire LNO-CC calculation, and disk I/O is completely avoided in these steps. This option is only available with `localcc`$\geq$`2018` and `ccprog=ccsd`.

`trffirst` The integral transformation steps to the LNO bases of the LISs are completed first for all LISs, which is followed by the CC calculations performed for each LIS in their respective LNO basis. This option requires storing the integral lists of all LISs on disk. It is usually not limiting to store the three-center LNO integrals required for `ccprog=ccsd` or the four-center LNO integrals required for `ccprog=mrcc`.

Default: `lccoporder=lccfirst` for `localcc`$\geq$`2018`,
`lccoporder=trffirst` for `localcc`$<$`2018` or `ccprog=mrcc`.

Example: to force storing the integrals on disk for `localcc`$\geq$`2018` set `lccoporder=trffirst`

`lccrest` Use this keyword to restart local CC calculations in the case of `localcc`$\geq$`2015`, e.g., after power failure. If `localcc`$\geq$`2018` (and `lccoporder=lccfirst`) the calculation can be restarted at any stage of the local correlation calculation via `lccrest=restart`. In this case all the intermediate files located in the folder of execution have to be kept intact, and only the updated `MINP` file (containing `lccrest=restart`) should be overwritten before the calculation is restarted. The HF calculation and the orbital localization is skipped via the `scfiguess=off` and the `orblocguess=read` options, therefore it is important not to modify the `VARS`, `SCFDENSITIES`, `FOCK`, and `MOCOEF*` files either. Note that the beginning of the computation up to the pair energy evaluation is repeated in the restarted run, and the loop over the extended domains will continue from the index of the first unfinished domain.

For the restart with `lccrest=on`, compatible with options `localcc=2015`, `localcc=2016`, or `localcc`$\geq$`2018` and `lccoporder=trffirst`, the LMP2 calculation and the integral transformations have to be completed, and for the remaining domains the `localcc.restart`, `DFINT_AI.*`, `DFINT_AB.*`, `DFINT_IJ.*`, `ajb.*`, `55.*`, `56`, and `VARS` files are required, and only the updated `MINP` file (containing `lccrest=on`) should be overwritten before the calculation is restarted. In the case of `localcc`$\geq$`2016`, `talg=lapl` or `topr`, and `lccoporder=trffirst` the `laplbas.*` files are also needed.

In the case of an LMP2 calculation or the LMP2 part of an LNO-CC calculation with `lccoporder=trffirst` and `localcc`$\geq$`2016`, if the loop for the extended domains is started but not finished, the calculation can be restarted with the `lccrest= domain` option. In this case all the intermediate files located in the folder of execution have to be kept intact, and only the updated `MINP` file (containing `lccrest=domain`) should be overwritten before the calculation is restarted. Note that the beginning of the computation up to the pair energy evaluation is repeated in the restarted run, and the loop over the extended domains will continue from the index of the first unfinished domain.

In the case of restricted open-shell local correlation computations, the `FOCK.OD` file is also needed. For `qro=on`, the `MOCOEF.QRO file` should also be preserved.

Options: `restart`, `on`, `off`, or `domain`.

Default: `lccrest=off`

Example: to restart a local CCSD(T) calculation set `lccrest=restart` if `localcc`≥`2018` (and `lccoporder=lccfirst`). In the case of `localcc<2018` and/or `lccoporder=trffirst` set `lccrest=on` or `lccrest=domain` depending on the point of interruption.

`lcorthr` Controls the accuracy of local correlation calculations by setting the relevant thresholds: `bp*`, `lnoepso`, `lnoepsv`, `laptol`, `naf_cor`, `osveps`, `wpairtol`, `spairtol` (see also Refs. 30, 31, and 38 for details).

Options:

`vLoose` Very loose thresholds will be used, see Table 7.

`Loose` Relatively loose thresholds will be used, see Tables 6 and 7.

`Normal` Default threshold set if `localcc`≥`2018`. `lcorthr=Normal` is a synonym for `lcorthr=Loose` if `localcc=2015` or `2016`. See Tables 6 and 7.

`Tight` Tight thresholds will be used, see Tables 6 and 7.

`vTight` Very Tight thresholds will be used, see Table 6.

`vvTight` Even tighter or very very Tight thresholds will be used, see Table 6.

`0` The truncation thresholds will be set so that the canonical energy be reproduced, it is only useful for testing.

Notes:

1. The values of the thresholds controlled by `lcorthr` are summarized in Table 6 for the `localcc`≥`2018` case and in Table 7 for the previous schemes. The performance of `lcorthr=Loose` to `vTight` settings for `localcc=2018` is extensively benchmarked in Ref. 42. The `vLoose` and `vvTight` settings may be employed for convergence studies but their accuracy/cost performance is mostly not practical for production calculations.
2. Expected accuracy. Using the `Normal` settings for local MP2 and CCSD(T), if `localcc`≥`2018`, 1 kJ/mol (1 kcal/mol) average (maximum) errors are expected in energy differences even for relatively complicated or sizable systems (see Refs. 31, 38, and 42). Note that molecules with complicated electronic structure can exhibit slower convergence with the local approximations and higher local errors [42, 163]. Thus, converge tests with increasingly tighter `lcorthr` settings are suggested at least for some representative systems to determine appropriate local correlation settings and for the robust estimation of the local approximation errors [42]. In the case of monotonic convergence with respect to `lcorthr`, the corresponding LNO approximations of Ref. 42 are also recommended to further increase the accuracy.
   In the case of `localcc=2015`, considering local dRPA and dRPA related methods, the expected average (maximum) errors for energy differences are 2 kJ/mol (2 kcal/mol) with `Loose` and 1 kJ/mol (1 kcal/mol) with `Tight` settings (see Ref. 30).

Table 6: Keyword values set by `lcorthr` if `localcc>=2018`.

| `calc` | LMP2[a] or LNO-CC | | | | | | all |
|---|---|---|---|---|---|---|---|
| `localcc` | ≥ 2018 | | | | | | |
| `lcorthr` | `vLoose` | `Loose` | `Normal` | `Tight` | `vTight` | `vvTight` | `0` |
| `bpedo` | `0.9999` | `0.9999` | `0.9999` | `0.99995` | `0.99999` | `0.999995` | `1.0` |
| `wpairtol` | `1e-4` | `3e-5` | `1e-5` | `3e-6` | `1e-6` | `3e-7` | `0.0` |
| `lnoepso` | `1e-4` | `3e-5` | `1e-5` | `3e-6` | `1e-6` | `3e-7` | `0.0` |
| `lnoepsv` | `1e-5` | `3e-6` | `1e-6` | `3e-7` | `1e-7` | `3e-8` | `0.0` |
| `laptol` | `1e-1` | `1e-1` | `1e-2` | `1e-3` | `1e-3` | `1e-3` | `1e-6` |
| `naf_cor` | `1e-2` | `1e-2` | `1e-2` | `1e-2` | `1e-3` | `1e-3` | `off` |
| `bpcompo` | `0.985` | `0.985` | `0.985` | `0.985` | `0.985` | `0.985` | `1.0` |
| `bpcompv` | `0.98` | `0.98` | `0.98` | `0.98` | `0.98` | `0.98` | `1.0` |
| `bppdo` | `0.999` | `0.999` | `0.999` | `0.999` | `0.999` | `0.999` | `1.0` |
| `bppdv` | `0.98` | `0.98` | `0.98` | `0.98` | `0.98` | `0.98` | `1.0` |
| `bpedv` | `0.995` | `0.995` | `0.995` | `0.995` | `0.995` | `0.995` | `1.0` |
| `osveps` | `off` | `off` | `off` | `off` | `off` | `off` | `off` |
| `spairtol` | `off` | `off` | `off` | `off` | `off` | `off` | `off` |

[a] See note 4 for `lcorthr` for additional details of LMP2.

3. For the correlation of sub-valence electrons a relatively slow convergence of the natural orbital approximations was observed (e.g., for systems with s-block or 3d metals). At the moment, for such cases, tighter `lcorthr` options are advised.
4. See additional details in the discussion of the `localcc` keyword below.

Default: `lcorthr=Normal`

Example: to use tight thresholds set `lcorthr=Tight`

`lmp2dens` Determines whether the MP2 density matrix fragments are calculated using the "correct" expressions derived for the general type of orbitals, or using the expressions derived for the canonical case (as described in Ref. 22).

Options:

`on` The MP2 density matrix fragments are calculated using the correct, non-canonical expressions.

`off` The MP2 density matrix fragments are calculated using the approximate canonical expressions (as defined in Ref. 22).

Notes:

1. To reproduce the method described in Ref. 22 use `lmp2dens=off`.
2. The use of `lmp2dens=on` is recommended since in this case the local CC energy can be corrected by the difference of the local MP2 energy and the approximate local MP2 energy calculated in the local interacting subspaces (see `Total CC... energy + correction` in the output). This correction usually improves the local CC energy.

Default: `lmp2dens=on`

Table 7: Keyword values set by `lcorthr` for `localcc=2015` or `2016`. Note that identical settings are used for both `lcorthr=Normal` and `lcorthr=Loose` if `localcc=2015` or `2016`.

| calc | MP2, dRPA, CC | | MP2 | | CC | | all |
|---|---|---|---|---|---|---|---|
| localcc | 2015 | | 2016 | | 2016 | | |
| lcorthr | Normal | Tight | Normal | Tight | Normal | Tight | 0 |
| bpedo | - | - | 0.9998 | 0.9999 | 0.9999 | 0.99995 | 1.0 |
| wpairtol | 1e-6 | 1e-7 | 1.5e-5 | 1e-5 | 1e-5 | 3e-6 | 0.0 |
| spairtol | 1e-4 | 1e-5 | off | off | off | off | 0.0 |
| osveps | 1e-3 | 1e-4 | off | off | off | off | 0.0 |
| lnoepso | 3e-5 | 1e-5 | - | - | 2e-5 | 1e-5 | 0.0 |
| lnoepsv | 1e-6 | 3e-7 | - | - | 1e-6 | 5e-7 | 0.0 |
| laptol | - | - | - | - | 1e-2 | 1e-3 | - |
| naf_cor | 1e-2 | 8e-3 | 2e-3 | 2e-3 | 1e-2 | 1e-2 | off |
| bpcompo | 0.985 | 0.985 | 0.985 | 0.985 | 0.985 | 0.985 | 1.0 |
| bpcompv | 0.98 | 0.98 | - | - | - | - | 1.0 |
| bppdo | - | - | 0.999 | 0.999 | 0.999 | 0.999 | 1.0 |
| bppdv | - | - | 0.98 | 0.98 | 0.98 | 0.98 | 1.0 |
| bpedv | - | - | 0.995 | 0.995 | 0.995 | 0.995 | 1.0 |

Example: to use the canonical expressions give `lmp2dens=off`

`lnoepso` Threshold for the occupation numbers of occupied local natural orbitals (LNOs) in the case of local correlation calculations, for state-averaged MP2/CIS(D) occupied natural orbitals for reduced-cost excited-state calculations, or for MP2 occupied natural orbitals for reduced-cost correlation calculations, see also keyword `lnoepsv`. See Ref. 24 as well as Refs. 35 and 37 for more details.

Options:

*<any real number in the [0,1] interval>* Orbitals with occupation numbers greater than 1-`lnoepso` will be frozen.

Notes:

1. For default settings with options other than `lcorthr=Normal`, see the description of `lcorthr`.
2. For FNO-CC calculations, occupied NOs are only implemented for closed-shell references with density fitting.
3. Frozen occupied NOs are not implemented for explicitly correlated methods.

Default: `lnoepso=1e-5` for local correlation, `lnoepso=0` for excited-states

Example: to set a threshold of $5 \cdot 10^{-6}$ type `lnoepso=5e-6`

`lnoepsv` Threshold for the occupation numbers of virtual local natural orbitals (LNOs) in the case of local correlation calculations, for state-averaged MP2/CIS(D) virtual natural orbitals for reduced-cost excited-state calculations, or for MP2 virtual natural orbitals for reduced-cost correlation calculations, see also keyword `lnoepso`. See Ref. 24 as well as Refs. 35, 37, 48, and 60 for more details.

Options:

*<any real number in the [0,1] interval>* Orbitals with occupation numbers smaller than this number will be dropped.

Note: For default settings with options other than `lcorthr=Normal`, see the description of `lcorthr`.

Default: `lnoepsv=1e-6` for local correlation; `lnoepsv=7.5e-5` for excited states if no spin scaling is applied; `lnoepsv=3e-5` for spin-scaled excited-state models (such as SCS-CC2, SOS-CC2, . . . ); `lnoepsv=5e-5` for reduced-cost (FNO) ground-state calculations if no explicit correlation is used. For explicitly correlated CC calculations, `lnoepsv=1e-4` with the cc-pVDZ-F12 and aug-cc-pVDZ bases, `lnoepsv=1e-6` with cc-pVQZ-F12 and aug-cc-pVQZ, and `lnoepsv=1e-5` otherwise.

Example: to set a threshold of $5 \cdot 10^{-7}$ type `lnoepsv=5e-7`

`localcc` Specifies if local correlation calculation is performed. See Refs. 22, 24, 30, 31, 36, and 38 for more details.

Options:

`off` No local correlation calculation is performed.

`2013` The algorithm of Ref. 24 is used.

`2015` An algorithm based on Ref. 30 is used.

`2016` The local MP2 algorithm of Ref. 31 or the local CC algorithm of Ref. 36 is used.

`2018` The local MP2 algorithm of Ref. 31 (with tighter settings) or the local CC algorithm of Refs. 38 and 42 is used.

`2021` The local MP2 algorithms of Ref. 31 (with tighter settings) and 51 (for restricted open-shell references) or the closed-shell local CC algorithm of Refs. 38 and 42 are used.

`2024` The local CC algorithms of Ref. 66 are used for restricted open-shell references. The closed-shell local CC algorithm of Refs. 38 and 42 are slightly modified (see notes below) to ensure compatibility with the open-shell algorithm, e.g., so that the closed- and open-shell local CC algorithms provide identical results for closed-shell species.

`on` Equivalent to `localcc=2024` in the case of MP2 or CC, and equivalent to `localcc=2015` otherwise

Default: `localcc=off`

Example: for local correlation calculations give `localcc=on`

Notes:

1. Local correlation methods can also be run if the prefix “L” (or “LNO-” in the case of CC methods) is added to the corresponding option of keyword `calc`, see the description of `calc`. For example, `calc=LNO-CCSD(T)` or `calc=LMP2`. Local SCS-MP2 (SOS-MP2) calculations can be run with `calc=LSCS-MP2` (`calc=LSOS-MP2`).

2. The restricted open-shell local MP2 and coupled cluster approaches are available via `localcc`≥`2024` [51, 66]. ROHF, ROKS, and in combination with unrestricted references, also QRO references can be employed. With UHF (UKS) references, `qro=on` is set for `localcc`≥`2021`.
3. For `localcc>2018`, the NAF approximation is not employed for the local MP2 correlation energy of `calc=LMP2`, thus this local MP2 result is exactly the same as the byproduct local MP2 energy of an LNO-CCSD(T) computation with `localcc>2018`. The NAF approximation was employed in Ref. 31, thus `naf_cor=2e-3` is set in combination with `localcc=2018` and `calc=LMP2`, leading to a slightly different local MP2 correlation energy than the one obtained with a `localcc=2018` type LNO-CCSD(T) computation (where the NAF approach is only employed at the CC level but not for MP2). For local MP2, `naf_cor=off` is set if `localcc=2015`. Keywords `lnoepso`, `lnoepsv`, and `laptol` are irrelevant for local MP2 calculations.
4. The closed-shell `localcc=2018` and `localcc=2021` variants are very close but not always identical. In the case off-center atomic orbitals employed via the `ghost` keyword, the domain construction scheme was improved, and the `localcc=2021` version is considerably more accurate than `localcc=2018`. For `localcc`≥`2024`, a more advanced version of this approach is turned on by default. It can expand both the atomic orbital and the projected atomic orbital domains when it detects potentially important orbitals outside of the previous domain definitions. We find this most helpful when ghost or non-atom centered orbitals are used, but this extension via `localcc`≥`2024` is turned on for all other computations too.
5. Minor updates were made to the closed-shell local CC algorithms moving from `localcc=2021` to `localcc=2024` to ensure consistency between the closed- and open-shell local CC formulae. These updates introduce a small change to the closed-shell LNO-CC energies (compared to `localcc`≤`2021`), which is well below the uncertainties corresponding to the LNO approximations. Namely, the expression of the virtual domain density matrices providing the virtual LNOs and the closed-shell (T) correction were updated [66]. Moreover, when warranted, a small number of additional strong pair occupied localized orbitals are added to the extended domains according to their estimated overlap with the domains' central localized orbital as described in Ref. 66.
6. For `localcc`≥`2024`, the selection of the Cholesky vectors/quadrature points for the Laplace integral used for the decomposition of energy denominators in the MP2 energies of the extended domains is slightly updated. In most cases, the number of Cholesky vectors/quadrature points is not changed, however, for `localcc`≥`2024`, it is controlled via an extended expression for the tolerance of the selection, that is, min($10^{-5}$,10·`cctol`,`wpairtol`,`chtol`) for the Laplace quadrature points and min($10^{-4}$,100·`cctol`,10·`wpairtol`,`chtol`) for the Cholesky vectors, with `chtol` being adjustable in the `MINP` file and set to 100·`cctol` by default.

`localcorrsymm` Controls the point group symmetry recognition feature of ground-state local correlation methods with `localcc`≥`2018` as discussed in Sect. 3.4 of Ref. 38.

Options:

`off` Point group symmetry is not exploited for the local correlation computation.

`on` The symmetry equivalent domains are recognized by analyzing both the diagonal and off-diagonal elements of the Fock matrix transformed to the localized MO basis.

`diag` The symmetry equivalent domains are determined only based on the diagonal Fock matrix elements. This is a less strict criterion than `localcorrsymm=on`.

`read` The degeneracy levels of each localized MO can be read from the `MINP` input file. The domain correlation energy contribution of each localized MO will be multiplied by the positive integer degeneracy level in the MO order determined by the MO localization program. Orbitals without symmetry equivalent MOs should have a degeneracy level of 1. Negative integers denote that the correlation energy contribution of the corresponding MO is already accounted for in the domain of a symmetry equivalent MO. The correctness of these degeneracy levels is not checked by the program. The `orblocguess=read` or `restart` setting is recommended with the option to retain the order of the localized MOs.

Default: `localcorrsymm=off`

Notes:

1. While Abelian point group symmetry is recognized and exploited by the HF/KS-DFT programs, the orbital localization algorithms are not yet symmetry adapted. Hence the localized MOs may not exhibit (all of) the symmetry properties of the molecular structure and canonical MOs. Because of that, the `localcorrsymm` feature is turned off by default even if `localcc`≠`off`. If `localcorrsymm`≠`off` is set, it is recommended to manually check the degeneracy level determination in the output section "`Determine symmetry equivalent LMO lists`".
2. This point group symmetry identification approach [38] can also take advantage of non-Abelian symmetry if the canonical and localized MOs retain the corresponding symmetry properties even if (non-Abelian) symmetry is not enforced by the SCF and localization algorithms.

Example: to read the list of symmetry-equivalent MOs for $C_{2v}$ water:

```
localcorrsymm=read
2 -2 2 -2
```

`mact` Specifies the number of active orbitals in an MCSCF calculation. See also the description of keyword `docc`.

Options: The number of active orbitals per irrep should be given in the following format:

`mact=`$<n_1>, <n_2>, \ldots, <n_{N_{ir}}>$

where $<n_i>$ is the number of active orbitals in irrep $i$, and $N_{ir}$ is the number of irreps.

Default: There is no default, the active orbitals must be given in the case of an MCSCF calculation.

Example: See the description of keyword `docc` for examples.

`maxact` Maximum number of inactive labels. One can impose restrictions on the cluster operator using this keyword. The maximum number of virtual/occupied inactive labels on the singly, doubly, ... excited clusters can be specified.

Options: `on` or `off`. If `maxact=on`, the maximum number of virtual and occupied inactive labels must be specified in the subsequent line as an integer vector. The integers must be separated by spaces. The vector should contain as many elements as the excitation rank of the highest excitation in the cluster operator. The integers are maximum number of virtual/occupied inactive labels allowed on amplitudes of single, double, ... excitations, respectively.

Default: `maxact=off`

Example: Suppose that we have up to quadruple excitations, and the single, double, triple, and quadruple excitations are allowed to have maximum of 1, 2, 2, and 1 inactive virtual and occupied labels, respectively. Then the input file should include the following lines:

```
maxact=on
1 2 2 1
```

`maxdim` Maximum number of trial vectors in the Davidson procedure in the case of CIS, TDA, TD-HF, and TD-DFT calculations. The procedure will be restarted if the dimension of the reduced subspace reaches `maxdim`.

Options: <*any positive integer*>

Default: `maxdim`=max(200, 3*max(`ntrip`, `nsing`))

Example: for a maximum of 50 expansion vectors set `maxdim=50`

`maxex` Level of highest excitation included in the cluster operator in the case of MRCI/CC calculations. In an MR calculation all single, double (or higher) excitations out of the reference determinants are included in the cluster operator (see the description of keyword `nacto`), however, the very high excitations are frequently irrelevant. Using this option the latter can be dropped. If `maxex` is set to a positive integer $n$, only up to $n$-fold excitations will be included in the cluster operator. The excitation manifold can be further selected by imposing constrains on the number of active/inactive labels of the excitations (see keyword `maxact`). See Refs. 4 and 5 for more details.

Options:

`0` The excitation manifold is not truncated.

<*any positive integer*> The excitation manifold is truncated at $n$-fold excitations, see above.

Default: `maxex=0`

Example: to truncate the excitation manifold at triple excitations set `maxex=3`

`maxmicroit` Maximum number of microiterations in a Newton step in the case quadratic SCF calculations (see keyword `qscf`).

Options: <*any positive integer*>

Default: `maxmicroit=100`

Example: to increase the maximum number of microiterations to 200 give `maxmicroit=200`

`mcscfiguess` Initial guess for the MCSCF calculation.

Options: The `sad`, `ao`, `core`, `mo`, and `restart` options can be used as described for keyword `scfiguess`. In addition, there is one further option:

`HF` An RHF or ROHF calculation is run before the MCSCF calculation, and the HF MOs will be used as initial guess in the MCSCF calculation.

Note: To restart the MCSCF calculation using MOs from a previous MCSCF run, e.g., when calculating potential energy surfaces, use `mcscfiguess=mo`

Default: `mcscfiguess=HF`

Example: set `mcscfiguess=sad` to start the MCSCF calculation from SAD initial guess.

`mem` Specifies the core memory used.

Options:

*<any positive integer>*`MB` The amount of memory to allocate is specified in megabytes

*<any positive integer>*`GB` The amount of memory to allocate is specified in gigabytes

Default: `mem=256MB`

Example: to allocate 8 GB core memory the user should set `mem=8GB`

`mmprog` Specifies the molecular mechanics program to be used for ONIOM calculations. See the description below for the supported methods and programs. Currently, it is not necessary to specify the keyword as MRCC automatically checks the program availability for the specified molecular mechanics method.

Options:

`off` molecular mechanics method is not requested.

`xtb` The XTB package of Grimme [86].

Default: The default for GFN-FF is `xtb`, otherwise `mmprog=off`.

Example: to select the XTB program for a calculation, set `mmprog=xtb`.

`molden` Specifies whether input file for the MOLDEN program and an xyz-file containing the Cartesian coordinates are written (see also Sect. 14).

Options:

`on` Cartesian coordinates, basis set information, and MO coefficients are saved to file `MOLDEN`. This file can be opened by MOLDEN and used to visualize the structure of the molecule and the MOs. In addition, Cartesian coordinates are also written to file `COORD.xyz` in xyz (XMol) format, which can be processed by many molecular visualization programs.

`off` The construction of the `MOLDEN` input and the `COORD.xyz` file is turned off.

Default: `molden=on`

Example: if you do not need MOLDEN input and the `COORD.xyz` file, add `molden=off`

`mpitasks` Specifies the number of MPI tasks spawned by `dmrcc`. Currently only executables `scf`, `mrcc`, and `ccsd` can be spawned for MPI parallel execution. All of the current MPI implementations replicate all arrays stored in memory for each MPI task. The `mem` keyword sets the available memory for a single MPI task. Hence, `mem` should be set to the total available memory of the compute node divided by the number of MPI tasks used on a single compute node.

Options: <*any positive integer*>

Default: `mpitasks=1`

Note: For optimal performance, please set `mpitasks` to the total number of available CPUs/NUMA nodes/nodes/cores, etc., as the additional number of processes on top of `mpitasks` are driver processes running mostly in the background and do not require dedicated resources.

Example: to run an MPI-parallel CC calculation using 2 MPI tasks set `mpitasks=2`

`mulmet` Specifies the multipole approximation used for the evaluation of pair energies of distant pairs.

Options:

`0` The simplified dipole-dipole estimate of Ref. 312 will be used.

`1` Full dipole-dipole estimate [313].

`2` All the terms are included in the multipole expansion up to the contributions of the quadrupole moment [31].

`3` All the terms are included in the multipole expansion up to the contributions of the octapole moment [31].

Default: `mulmet=3` if `localcc`$\geq$`2016`, `mulmet=0` otherwise

Example: to use the octapole approximation set `mulmet=3`

`mult` Spin multiplicity ($2S+1$) of the Hartree–Fock or Kohn–Sham wave function. If a CI or CC calculation is also performed, the same multiplicity is supposed for the ground-state wave function. For excited states the multiplicity will be arbitrary, only $M_S$ is conserved. For closed-shell reference determinants the multiplicity (strictly speaking the parity of $S$) can be controlled by keywords `nsing` and `ntrip`, see below.

Options: <*any positive integer*>

Default: for atoms the corresponding experimental multiplicity is set, for molecules `mult=1` (singlet) for an even number of electrons, `mult=2` (doublet) otherwise

Example: for a triplet state one should give `mult=3`

`nab` Specifies whether the natural auxiliary basis (NAB) approximation will be used for explicitly correlated CC calculations and also specifies the threshold for the occupation numbers of NAB functions (see Ref. 60).

Options:

`off` the NAB approach will not be used.

*<any real number in the [0,1] interval>* A NAB will be constructed and auxiliary functions with occupation numbers smaller than this number will be dropped.

`on` Sets `nab=0.1`

Default: `nab=0.1` for reduced-cost explicitly correlated calculations (that is, if the `FNO-` prefix is added to the acronym of the selected method), `nab=off` otherwise.

Example: to use the NAB approximation and set a threshold of 0.2, type `nab=0.2`

`nacto` Number of active occupied spinorbitals for multi-reference (active-space) CI/CC calculations. By default, nacto pieces of spinorbitals under the Fermi level are supposed to be active. This can be overwritten using keyword `active`, which enables the user to select the active orbitals manually (see the description of keyword `active`). In a MRCI/CC calculation a complete active space (CAS) is supposed defined by keywords `nacto` and `nactv` (or alternatively by `active`) and up to $n$-fold excitations from the reference determinants of this space are included in the excitation manifold, where $n$ is determined by keyword `calc` (2 for CCSD, 3 for CCSDT, ...). See Ref. 4 for more details. See also keywords `nactv`, `maxex`, and `maxact`. Note also that this keyword only sets the active orbitals for the post-SCF calculation, the MCSCF active orbitals can be specified by keyword `mact`.

Options: *<any positive integer>* or `0`

Default: If MCSCF reference is used, `nacto` is set so that the MCSCF active orbitals (given by keyword `mact`) will also be the active orbitals in the post-SCF calculation. `nacto=0` in the general case.

Example: for two active occupied spin-orbitals give `nacto=2`

`nactv` Number of active virtual spinorbitals for multi-reference (active-space) CI/CC calculations. By default, nactv pieces of spinorbitals above the Fermi level are supposed to be active, which can be overwritten using keyword `active`. For a detailed description see keyword `nacto`.

Options: *<any positive integer>* or `0`

Default: If MCSCF reference is used, `nactv` is set so that the MCSCF active orbitals (given by keyword `mact`) will also be the active orbitals in the post-SCF calculation. `nactv=0` in the general case.

Example: for two active virtual spin-orbitals give `nactv=2`

`nafalg` Specifies how natural auxiliary functions (NAFs) will be constructed in the case spin-unrestricted MOs. NAFs can be calculated by diagonalizing $(\mathbf{W}^\alpha + \mathbf{W}^\beta)/2$ or $\mathbf{W}^\alpha$ (see Ref. 26 for the definitions). The latter option is somewhat more efficient but can be dangerous for processes involving atoms.

Options:

`albe` NAFs are constructed from $(\mathbf{W}^\alpha + \mathbf{W}^\beta)/2$.

`alpha` NAFs are constructed from $\mathbf{W}^\alpha$.

Default: `nafalg=albe`

Example: to use $\mathbf{W}^{\alpha}$, set `nafalg=alpha`

`nafdens` This keyword can be used to control the state-averaged density construction for local ADC(2) or CVS-ADC(2) calculations. See Ref. 43 for further details.

Options:

`off` The NAF approximation is not used for the density construction.

`on` A fast, local domain-based algorithm is used for the density construction invoking the NAF approximation.

Default: `nafdens=off`

Note: If `redcost_exc=on` and `localcc=2016` settings are used, `nafdens=on` is set by default.

Example: to switch on the NAF approximation at the construction of density matrices, set `nafdens=on`

`naf_cor` Specifies whether natural auxiliary functions (NAFs) will be used for density-fitting correlated calculations and also specifies the threshold for the occupation numbers of NAFs (see Ref. 26).

Options:

`off` NAFs will not be constructed.

<*any real number in the [0,1] interval*> A NAF basis will be constructed and NAFs with occupation numbers smaller than this number will be dropped.

`on` Sets `naf_cor=5e-2` when `ccprog=ccsd` is set, `naf_cor=1e-2` for NAF-MP2 analytical gradient calculations or `naf_cor=5e-3` in other cases if the use of NAFs is not specified as default. If NAFs are used as default, e.g., in the cases of certain local correlation and reduced-cost excited-state approaches, `naf_cor=on` should not be written to the input file.

Note: If `ccprog=mrcc` or `dfbasis_cor=none`, no correction for the dropped NAFs is calculated.

Default: according to the value of `lcorthr` for local correlation methods; `naf_cor=0.1` for the reduced-cost excited-state approaches of Refs. 35 and 37 (see keyword `redcost_exc`) without spin scaling; `naf_cor=0.075` for the spin-scaled version of the latter methods (such as SCS-CC2, SOS-CC2, . . . ); `naf_cor=0.05` for reduced-cost ground-state calculations (that is, if the`FNO-` prefix is added to the acronym of the selected method); `naf_cor=off` otherwise.

Example: to use NAFs and set a threshold of $10^{-2}$, type `naf_cor=1e-2`

`naf_f12` Specifies whether natural auxiliary functions (NAFs) will be used for explicitly correlated CC calculations at the calculation of the F12-dependent intermediates and also specifies the threshold for the occupation numbers of NAFs (see Ref. 60).

Options:

`off` NAFs will not be constructed.

<*any real number in the [0,1] interval*> A NAF basis will be constructed and NAFs with occupation numbers smaller than this number will be dropped.

`on` Sets `naf_f12=5e-2`

Default: `naf_f12=5e-2` for reduced-cost explicitly correlated calculations (that is, if the `FNO-` prefix is added to the acronym of the selected method), `naf_f12=off` otherwise.

Example: to use NAFs and set a threshold of $10^{-2}$, type `naf_f12=1e-2`

`naf_scf` Specifies whether NAFs will be used for density-fitting SCF calculations and also specifies the threshold for the occupation numbers of NAFs (see Ref. 26). The syntax is analogous with that for keyword `naf_cor`.

`naftyp` Specifies how NAFs will be constructed in the case of local correlation calculations. NAFs are constructed by diagonalizing the $\mathbf{W} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$ matrix where $\mathbf{J}$ is a particular block of the three-center Coulomb integral matrix (see Refs. 26 and 38 for details).

Options:

`Jai` NAFs are constructed from $J_{ai}^{P}$

`Jpi` NAFs are constructed from $J_{pi}^{P}$

`Jpq` NAFs are constructed from $J_{pq}^{P}$ using the $p \geq q$ values.

`Jpq_inv` NAFs are constructed from all $J_{pq}^{P}$ integrals. This option generates NAFs invariant to orbital rotations among the MOs but it overweighs the $p \neq q$ integrals compared to the diagonal $p = q$ integrals, leading to a minor numerical deviation compared to the `naftyp=Jpq` option.

`Jmi` NAFs are constructed from $J_{\mu i}^{P}$

Default: `naftyp=Jpq` for local CC and `localcc`≥`2016` as well as for the reduced-cost excited-state approaches of Refs. 35 and 37 (see keyword `redcost_exc`), `naftyp=Jai` for local MP2 and `localcc`≥`2016`

Example: to construct NAFs using $J_{pq}^{P}$ set `naftyp=Jpq`

`nchol` Number of Cholesky vectors/quadrature points for the Laplace integral in the case methods based on the decomposition of energy denominators. See also the description of keyword `dendec`.

Options:

`auto` The number of Cholesky vectors/quadrature points will be automatically determined to achieve the required precision.

<*any positive integer*> The number of Cholesky vectors/quadrature points will also be automatically determined but the maximum number of the vectors cannot exceed this number.

Default: `nchol=auto`

Example: to use ten Cholesky vectors/quadrature points give `nchol=10`

`ndeps` Step size for the numerical differentiation in atomic units.

Options: <*any positive real number*>

Default: `ndeps=1e-3`

Example: for a step size of $5 \cdot 10^{-4}$ a.u. for numerical Hessian evaluation set `ndeps=5e-4`

`nstate` Number of electronic states including the ground state and excited states. In non-relativistic calculations, for closed-shell reference determinants `nstate` is supposed to be the number of singlet states. See also keywords `nsing`, `ntrip`, and `optex`.

Options: <*any positive integer*>

Default: `nstate`=max(1, `nsing+ntrip`)

Example: for three states give `nstate=3`

`nsing` Number of singlet electronic states (strictly speaking the number of of states with $M_S = 0$ and $S$ is even) including the ground state and excited states. Use this option only for non-relativistic calculations and closed-shell reference determinants, it should be zero otherwise. In the case of closed-shell reference determinants a partial spin-adaptation is possible, see Ref. 3. This enables us to search for singlet and triplet roots separately. See also keywords `nstate` and `ntrip`.

Options: <*any positive integer*>

Default: `nsing=1` for closed-shell reference determinants, `nsing=0` otherwise

Example: for two singlet states give `nsing=2`

`ntrip` Number of triplet electronic states (strictly speaking the number of of states with $M_S = 0$ and $S$ is odd) including the ground state and excited states. Use this option only for non-relativistic calculations and closed-shell reference determinants, it should be zero otherwise. See the description of keywords `nstate` and `ntrip`.

Options: <*any positive integer*>

Default: `ntrip=0`

Example: for two triplet states give `ntrip=2`

`nto` If set, natural transition orbitals (NTOs) or state-averaged natural orbitals (NOs) are written (see also Sect. 14).

Options:

- `on` NTOs or NOs are saved to the corresponding files (see Notes). These files can be handled using the same programs as file `MOLDEN` can (see keyword `molden`).
- `off` The construction of the corresponding NTO or NO files is turned off.

Notes:

1. For the reduced-cost CC2, CIS($D_\infty$), and ADC(2), methods of Refs. 35 and 37, and for their spin-scaled versions, the pseudo-canonicalized state-averaged NOs will be saved to files `MOLDEN_NO.`$n$, where $n$ is the serial number of the excited state.
2. For the NTOs, in the case of CIS/TDHF/TDDFT/TDA-TDDFT calculations, the `MOLDEN_CIS_NTO.`$n$ files are generated for each state, where $n$ is the serial number of the excited state according to energy order.

3. For the NTOs, in the case of ADC(2)/CC2/CIS($D_\infty$) calculations, the `MOLDEN_CORR_NTO.`$n$ files are generated. For a canonical calculation, the files are labeled according to the order of the excitation energies of the second-order method. In contrast, if a state-specific, for example, reduced-cost calculation is performed, the files are labeled according to the order of the CIS excitation energies. Note that the ADC(2) and CIS roots can be swapped. You can check at the end of the output in the summary if such rearrangement occurred.
4. For the NTOs, in the case of double hybrid calculations, the `MOLDEN_DH_NTO.`$n$ files are generated. In the case of the ADC(2)-based ansatz, the files are labeled according to the order of the excitation energies. In contrast, if the CIS(D)-based ansatz is used, the files are labeled according to the order of the TDA excitation energies.
5. For the second-order methods, NTOs are calculated using only single excitation coefficients.

Default: `nto=off`

Example: if you need the NTO or NO files, add `nto=on`

`numgrad` This keyword controls the calculation of the numerical gradient of the energy with respect to nuclear positions.

Options:

`on` Calculates the numerical gradient with a four point central difference scheme with step size of 0.01 bohr. Please note that these settings may not be suitable for all types of calculations and molecules. Currently there is no way to control the step size or the numerical stencil from the input file.

`off` MRCC does not calculate numerical gradients.

Default: `numgrad=off`

`occ` Specifies the occupation of the Hartree–Fock determinant.

Options:

1. If this keyword is not given, the occupation is automatically determined in the SCF calculations.
2. For RHF calculations the occupation should be given in the following format:
   `occ=`$< n_1 >, < n_2 >, \ldots, < n_{N_{ir}} >$
   where $< n_i >$ is the number of occupied orbitals in irrep $i$, and $N_{ir}$ is the number of irreps.
3. For ROHF and UHF calculations the occupation should be given as
   `occ=`$< n_1^\alpha >, \ldots, < n_{N_{ir}}^\alpha > / < n_1^\beta >, \ldots, < n_{N_{ir}}^\beta >$
   where $< n_i^\sigma >$ is the number of occupied $\sigma$ spinorbitals in irrep $i$.

Default: `occ` is not specified, that is, the occupation is set by the SCF program.

Examples:

1. Water, RHF calculation:
   `occ=3,1,1,0`

2. Water, UHF calculation:
   `occ=3,1,1,0/3,1,1,0`
3. Carbon atom, ROHF or UHF calculation:
   `occ=2,0,0,0,0,1,0,1/2,0,0,0,0,0,0,0`

`occri` Turns on the OCC-RI-K [50, 159] algorithm. It can be used together with local and non-local direct DF-SCF calculations. It is available for both Coulomb- and overlap-metric DF-SCF methods. The desired fitting metric can be selected with the `fitting` and `dual_df` keywords.

Options:

`on` Turns on the OCC-RI-K algorithm.

`off` Turns off the OCC-RI-K algorithm.

Default: It is turned off except for overlap-metric SCF calculations.

Example: to turn of the OCC-RI-K algorithm, set `occri=on`

`oniom` Initiates an ONIOM calculation [83, 93] and specifies the number of layers. In an ONIOM calculation, the system is split up into several layers described at different levels of theory. Both mechanical and electronic embeddings are supported. In the case of the mechanical embedding, the presence of the lower layers are omitted in upper-layer calculations, while a point charge representation of the lower layers is used in electrostatic embedding. The top layer (model system, active subsystem) is treated at the highest level, and the corresponding method, basis set, . . . are set by the other keywords in the `MINP` file, `calc`, `basis`, . . . , as usual. The atoms and the boundary handling schemes of the upper layers, and the methods of the lower layers must be specified in the subsequent lines as follows, starting from the lowest-level layer. For each layer three lines are required.

The atoms of the layer must be given by their serial numbers in the first line as $< n_1 >$,$<n_2>$,. . . ,$< n_k >$-$< n_l >$,. . . , where $n_i$'s are the serial numbers of the atoms. Serial numbers separated by dash mean that atoms $< n_k >$ through $< n_l >$ are part of the layer. Note that the numbering of the atoms must be identical to that used in the Z-matrix or Cartesian coordinate specification, but dummy atoms must be excluded. Atoms of an upper-level layer have to be a subset of the lower-level layers.

The method for the lower-level layer must be specified in the second line using the corresponding options of keyword `calc`, or the supported methods of external programs (see keywords `sqmprog` and `mmprog`). The highest-level method is specified by keyword `calc` (or keywords `calc` and `dft` if a KS reference is used in a correlation calculation). The specification of the basis set (for the low-level method), charge, and multiplicity of the lower-level layer is optional but can be given in the second line separated by forward slashes and spaces, e.g., as `PBE / cc-pVDZ / -1 / 1`. See keywords `basis`, `charge`, and `mult` for the available options, respectively. Note that keywords `basis`, `charge`, and `mult` specify the basis of the last calculation, the charge, and the multiplicity of the top layer, respectively. If the basis set is not specified under `oniom`, the basis set of the upper-level layer will be used, while the default options will be applied for the charge and the multiplicity if they are not set here.

In the third line, an integer should be given which specifies the border handling scheme: the automatic (0), the semi-automatic (1), and the manual (2) border handling schemes are supported.

In the case of the automatic handling of layer borders, the following procedure is executed if the method of the lowest layer is not calculated by an external program. The MOs are localized after the first, full-system calculation (see `orbloce` for the available options for the localization scheme), and the localized MOs are assigned to maximum of two atoms by the Boughton–Pulay algorithm [217]. If the completeness criteria of a truncated LMO is greater than 0.8, the bond order between the corresponding two atoms is increased by one. The borders of the layers are then determined by those bonds which are located on a layer and a non-layer atom. Depending on their integer bond order, a hydrogen, an oxygen, or a nitrogen atom is selected as link atom. The bond distance is set based on the type of the link atom and the layer atom (see the average bond distances at https://cccbdb.nist.gov/expbondlengths1.asp), and the link atom is placed in the direction of the original bond. Note that the implementation follows the convention of the AMBER program, where the distances between the link atoms and the layer atoms are fixed independently of the geometry. Note also that quadruple bonds are not separable automatically, and the separation of aromatic structures should be avoided. If the method of the lowest-level layer is handled by an external program, the same procedure is executed except that the the bonding table is not built by using the Boughton–Pulay algorithm. Instead, the bond table is based on the bond order table of the external program.

In the case of the manual border handling, the user has to specify the atom pairs of the borders, the link atoms, and the distance between the link atoms and the layer atoms. These specifications can be given in three input sections separated by forward slashes and spaces on the same line as follows:
$2\ /\ <B_1>:<x_1>-<y_1>,<B_2>:<x_2>-<y_2>,\ldots,<B_n>:<x_n>-<y_n>\ /\ <B_1>:<L_1>,<B_2>:<L_2>,\ldots,<B_n>:<L_n>\ /\ <B_1>:<R_1>,<B_2>:<R_2>,\ldots,<B_n>:<R_n>$ ,
where $B_n$, $x_n$, $y_n$, $L_n$, and $R_n$ are the serial number of the border, the layer atom of the border bond, the non-layer atom of the border bond, the atomic symbol of the link atom, and the distance between the layer atom and the link atom, respectively. Note that the serial numbers of the borders are always followed by a colon, different specifications of the same section are separated by a comma, and different specifications of the same border hold the same border serial number in all input sections. All the border parameters have to be specified in the case of manual border handling, and $B_n$ and $n$ have to be positive integers smaller than the number of the atoms in the system. Note that the unit of $R_n$ has to match that set by keyword `unit` as there are no consistency checks in this handling scheme. Note also that, currently, the manual border handling scheme is the only option if one considers using the GFN-FF method via the `Mrcc-xtb` interface.

The semi-automatic border handling (1) is the combination of the automatic and manual schemes: the borders are selected and set by the automatic approach, but these selections can be modified manually. Any automatically selected $B_m$ border can be deleted by writing a colon and an $X$ after the serial number of the border in the border atom pair section: $<B_m>:<X>$. New borders can also be defined in the same manner as in the case of the manual border handling, however, all the parameters of the newly defined borders have to be specified. Please be aware that the simple overwriting of single parameters of the automatic approach is not supported by this scheme.

The ONIOM implementation allows arbitrary number of layers, thus the input requirement of an ONIOM calculation with $N$ layers is $(N-1)*3$ lines: the atoms of the second layer are specified in the first line, while atoms of the third layer are specified in the fourth line after the keyword `oniom`, and so on. To allow full control over the keywords of the subcalculations, the `subminp` keyword is advised for use. Note that that projection-based embedding and multi-level local correlation are supported as the top-level calculations (see the keywords `embed` and `corembed` for the description). Note also that the restart of the ONIOM procedure is controlled by the `comprest` keyword. See also the examples below.

Options:

`off` No ONIOM calculation is performed.

<*any positive integer*>`-me` An ONIOM calculation with mechanical embedding will be carried out with this integer specifying the number of layers. The further parameters should be given in the subsequent lines as described above.

<*any positive integer*>`-ee` An ONIOM calculation with electronic embedding will be carried out with this integer specifying the number of layers. The further parameters should be given in the subsequent lines as described above. See also keywords `oniom_eechg` and `oniom_qcorr` for options of the point charge representation of the lower layers.

Default: `oniom=off`, while mechanical embedding is used if only a positive integer is given as input.

Examples:

1. A two-layer ONIOM calculation is requested where the model system (layer 2) consists of atoms 8 and 9 (using the automatic border handling scheme). The subcalculations will be executed in the following order: PBE/STO-3G (full system), PBE/STO-3G (model system), and CCSD/aug-cc-pVDZ (model system):
```
basis=aug-cc-pVDZ
calc=CCSD
oniom=2
8-9
PBE / STO-3g
0
```
2. The same settings are applied as in the previous case except that the borders are selected manually: layer 2 will have two borders between atoms 8-5 and 8-1. The first (second) link atom is a hydrogen (fluorine), which will be placed 1.09 Å (1.01 Å) away from atom 8 in the direction of atom 5 (1):
```
basis=aug-cc-pVDZ
calc=CCSD
oniom=2
8-9
PBE / STO-3g
2 / 1:8-5,2:8-1 / 1:H,2:F / 1:1.09,2:1.01
```

3. The same settings are applied as in the previous case except that the borders are selected semi-automatically: the first border of the automatic selection is deleted and an additional third border is specified which is located between atoms 8 and 2. In this case, chlorine will be used as link atom, and it will be placed 1.01 Å away from atom 5 in the direction of atom 2:
   ```
   basis=aug-cc-pVDZ
   calc=CCSD
   oniom=2
   8-9
   PBE / STO-3G
   1 / 1:X,3:8-2 / 3:Cl / 3:1.01
   ```
4. Four-layer ONIOM calculation, LNO-CCSD(T):LMP2:PBE:LDA (aug-cc-pVDZ:cc-pVDZ:6-31G*:STO-3G). Each layer uses automatic border handling scheme and has a different negative charge, except the top layer, which has zero charge:
   ```
   basis=aug-cc-pVDZ
   calc=LNO-CCSD(T)
   charge=0
   mult=1
   oniom=4
   1-8
   LDA / STO-3G / -3 / 1
   0
   1-5
   PBE / 6-31G* / -2 / 1
   0
   1-2
   LMP2 / cc-pVDZ / -1 / 1
   0
   ```
5. Two-layer ONIOM calculation, LNO-CCSD(T)-in-LMP2:PBE (aug-cc-pVTZ:6-31G*). The automatic border handling scheme is utilized. The multilevel local correlation approach is used as the top-level calculation with an automatic selection of active occupied molecular orbitals.
   ```
   basis=aug-cc-pVTZ
   calc=LNO-CCSD(T)
   corembed=on
   1-4
   LMP2
   0
   charge=0
   mult=1
   oniom=2
   1-8
   PBE / 6-31G* / 0 / 1
   0
   ```
6. Two-layer ONIOM calculation, LNO-CCSD(T)-in-PBE0:PBE (aug-cc-pVTZ:6-

31G*). The automatic border handling scheme is utilized. The projection-based embedding is used as the top-level calculation with an automatic selection of active occupied molecular orbitals.

```
basis=aug-cc-pVTZ
calc=LNO-CCSD(T)
embed=huzinaga
1-4
PBE0
0
charge=0
mult=1
oniom=2
1-8
PBE / 6-31G* / 0 / 1
0
```

7. Two-layer ONIOM calculation, PBE0:PM6-D3H4 (aug-cc-pVTZ:minimal basis). The automatic border handling scheme is utilized. The Mopac semi-empirical program is used for the PM6-D3H4 calculations.

```
basis=aug-cc-pVTZ
calc=PBE0
charge=0
mult=1
sqmprog=mopac2016
oniom=2
1-8
PM6-D3H4 / / 0 / 1
0
```

8. Two-layer ONIOM calculation, LNO-CCSD(T)-in-LMP2-in-PBE-D3:GFN2-xTB (aug-cc-pVTZ:minimal basis). The automatic border handling scheme is utilized for the ONIOM calculation, while the `amul` scheme is used for the selection of occupied orbitals in the embedding and multilevel local correlation calculations. The xTB semi-empirical program is used for the GFN2-xTB calculations.

```
basis=aug-cc-pVTZ
calc=LNO-CCSD(T)
charge=0
mult=1
orbloce=pm
sqmprog=xtb
corembed=on
1-4
0
embed=huzinaga
1-8
PBE-D3
0
```

```
oniom=2
1-12
GFN2-xTB / / 0 / 1
0
```

**`oniom_eechg`** Specifies the type of atomic charges used in the ONIOM calculations with electronic embedding. Note that in the case of the `mulli`, `lowdin`, `iao`, `chelpg` , and `mk` options, the atomic charges are generated after the first ONIOM subcalculation. Note also that in the cases where `gopt`≠`off` and `freq`≠`off`, these charges will be used in all calculations.

Options:

- `off` No embedding charges. This is equivalent with mechanical embedding.
- `mulli` Mulliken atomic charges are used as embedding charges.
- `lowdin` Löwdin atomic charges are used as embedding charges.
- `iao` Intrinsic atomic orbital (IAO) atomic charges atomic charges are used as embedding charges.
- `chelpg` CHELPG atomic charges are used as embedding charges.
- `mk` Merz–Singh–Kollman atomic charges are used as embedding charges.
- `external` The charges are obtained by an external program if the same program is used in the first calculation. Currently, only the XTB program is supported.
- `amber` Perturbation-independent charges are obtained from the topology file of the AMBER program. See the manual of AMBER how to generate such files and the example below how to use them with MRCC. If `oniom_eechg=amber` and `qmmm=off`, then it is assumed that the serial numbers of the atoms are the same in the topology file and in `MINP`.
- `user` Those atomic charges will be used that are specified after the line of `oniom_eechg`. Note that each given atomic charge must be in a new line, the order of the charges must be the same as the order of the corresponding atoms in the specification of the geometry, and the charge of all atoms must be specified.

Default: `oniom_eechg=lowdin`, if `oniom`≠`off` and electronic embedding is utilized, `oniom_eechg=off` otherwise.

Example:

1. Two-layer ONIOM calculation is requested with electronic embedding for a system with 9 atoms. The upper layer consists of atoms 8 and 9, while the layer borders are handled automatically. The subcalculations will be executed in the following order: PBE/STO-3G (full system), PBE/STO-3G (model system, in the presence of the IAO atomic charges of atoms 1-7), and CCSD/aug-cc-pVDZ (model system, in the presence of the IAO atomic charges of atoms 1-7).

```
basis=aug-cc-pVDZ
calc=CCSD
oniom_eechg=iao
oniom=2-ee
8-9
```

```
PBE / STO-3G
0
```

2. The same settings are applied to a 2-atom system, except that user-defined charges will be used:

```
basis=aug-cc-pVDZ
calc=CCSD
charge=-1
oniom_eechg=user
1.00
-1.00
oniom=2-ee
2
PBE / STO-3G
0
```

3. A two-layer ONIOM calculation is requested with electronic embedding, PBE0:GFN2-xTB (6-31G*:minimal basis). After the full system calculation, a GFN2-xTB and a PBE0 calculation are executed using the point charges obtained from the xTB program.

```
basis=6-31G*
calc=PBE0
oniom_eechg=external
oniom=2-ee
2
GFN2-xTB / / 0 / 1
0
```

4. A two-layer ONIOM calculation is requested with electronic embedding, PBE0:GFN2-xTB (6-31G*:minimal basis). After the full system calculation, a GFN2-xTB and a PBE0 calculation are executed using the point charges obtained from the topology file `example.prmtop` of the AMBER program.

```
basis=6-31G*
calc=PBE0
oniom_eechg=amber
example.prmtop
oniom=2-ee
2
GFN2-xTB / / 0 / 1
0
```

`oniom_pcm` Specifies the PCM scheme for ONIOM calculations [314, 315].

Options:

`off` ONIOM subcalculations will not use PCM.

`c` PCM is utilized only in the first calculation, i.e., in the lowest-level calculation performed for the entire system.

`x` PCM will be used in all subcalculations. The cavity of the upper layer calculation will be the same as in the first calculation. The wall of the cavity is repolarized by the density of the subcalculation.

Notes:

1. If the lowest-level calculation is handed by an external program, the implicit solvent model of the external program can be utilized with `oniom_pcm=c`.
2. If the Mopac program is utilized in the first calculation and `pcm=`<*solvent*>, then the `eps` and `rsolv` keywords are set for Mopac. The corresponding static permittivity and probe radius values of the given *solvent* are taken from the PCMSolver library.

Default: `oniom_pcm=x` if `oniom`≠`off` and `pcm`≠`off` and mechanical embedding is used, `oniom_pcm=c` if `oniom`≠`off` and `pcm`≠`off` and electronic embedding is used and the lowest-level calculation is handled by an external program, `oniom_pcm=off` otherwise.

Examples:

1. Two-layer ONIOM calculation is requested using the IEFPCM solvent model utilizing water for the medium. The upper layer consists of atoms 8 and 9, while the layer borders are handled automatically. The subcalculations will be executed in the following order: PBE/STO-3G (full system, in the cavity of the full system), PBE/STO-3G (model system, in the cavity of the full system), and CCSD/aug-cc-pVDZ (model system, in the cavity of the full system). Note that the interface is repolarized in all subcalculations.
```
basis=aug-cc-pVDZ
calc=CCSD
pcm=water
oniom_pcm=x
oniom=2
8-9
PBE / STO-3G
0
```
2. The same settings are applied as in example 1, except that PCM is only utilized in the first calculation:
```
basis=aug-cc-pVDZ
calc=CCSD
pcm=water
oniom_pcm=c
oniom=2
8-9
PBE / STO-3G
0
```
3. The same settings are applied as in example 2 except that the implicit solvent model is only utilized in the first calculation by an external program:
```
basis=aug-cc-pVDZ
calc=CCSD
pcm=water
```

```
oniom_pcm=c
oniom=2
8-9
PM6-D3H4 / / 0 / 1
0
```

`oniom_qcorr` Specifies the point charge correction scheme for ONIOM calculations to avoid the overpolarization of the electron density by those classical point charges that are spatially close to the quantum mechanical system. This feature also guarantees integer net charges for the subsystems.

Options:

`0` The correct integer charge of the subsystems is guaranteed. The point charges of the upper-layer atoms are summed, the integer charge of the upper-layer is subtracted from the sum, and the residual charge is distributed equally among those atoms that were not counted in the summation.

`1` Same as option `0`, but moreover, the point charges of the lower-layer host atoms are added to the point charge summation, and then they are zeroed.

`special` Same as option `1`, but moreover, user-defined point charges of the lower-layer host atoms are added to the point charge summation, and then they are zeroed. The serial numbers of the additional atoms have to be specified in the subsequent lines after the keyword `oniom_qcorr` for each layer. See the examples below.

`off` No charge correction scheme is utilized. The initial input point charges will be used throughout the subcalculations, thus the sum of the point charges may differ from the integer charge of the environment subsystem.

Notes:

1. In the case of options `0`, `1`, and `special`, the integer charges are guaranteed for each sublayer, thus a unique charge correction is utilized for each layer.
2. Option `1` only differs from option `0` if the subsystems are covalently bound.

Default: `oniom_qcorr=off` if `oniom=off`, and `oniom_qcorr=1` otherwise.

Example:

1. A CCSD:PBE (aug-cc-pVDZ:STO-3G) calculation is performed for the system that consists of 6 atoms. The manual border handling scheme is employed: a hydrogen link atom is positioned to the layer border, which is between atoms 2 and 5. No charge correction is utilized, thus the input point charges will be employed for the model system calculations. Note that the sum of point charges of the environmental subsystem ($-1.1$) differs from the integer charge of the subsystem ($-1$), that is, the charge of the total system (0) minus the charge of the model system (+1)

```
basis=aug-cc-pVDZ
calc=CCSD
charge=1
oniom_qcorr=off
```

```
oniom_eechg=user
0.85
-0.60
0.85
0.20
-1.5
0.20
oniom=2-ee
1-3
PBE / STO-3G / 0 / 1
2 / 1:2-5 / 1:H / 1:1.09
```

2. The same settings are applied as in the previous example except that the integer charge of the subsystem is guaranteed with `oniom_qcorr=0`. In this case, the sum of point charges of atoms 1-3 is summed (+1.1), the charge of the model system (+1) is subtracted, and the difference (0.1) is distributed equally among the rest of the charges, which results in point charges +0.233, −1.467, and +0.233 for atoms 4, 5, and 6, respectively. Note that the point charge representation of the environmental subsystem matches the integer charge of the subsystem.
3. The same settings are applied as in the previous example except that `oniom_qcorr=1`. In addition to the previous example, the point charge of the lower-layer host atom, which is atom 5, is included in the summation. After the integer charge of the subsystem (+1) is subtracted from the resulting point charge sum (−0.4), the residual charge (−1.4) is distributed equally among the rest of the point charges, which results in −0.5 and −0.5 for atoms 4 and 6, respectively. Note that the point charge representation of the environmental subsystem matches the integer charge of the subsystem.
4. A CCSD:PBE0:PBE (aug-cc-pVDZ:6-31G*:STO-3G) calculation is carried out for a system which consists of 12 atoms, and manual border handling scheme is employed. The same charge correction procedure is utilized as in the previous example, except that additional point charges are used in the summation procedure and then they are zeroed. In the case of the second layer calculations, atoms 7 and 8 are added to the charge correction procedure. In the case of the last layer calculations, the point charge of atoms 5-9 will also be used in the summation and then zeroed.

```
basis=aug-cc-pVDZ
calc=CCSD
charge=1
oniom_qcorr=special
7,8
5-9
oniom_eechg=lowdin
oniom=2-ee
1-6
PBE / STO-3G / 0 / 1
2 / 1:6-9 / 1:H / 1:1.09
```

```
1-3
PBE0 / 6-31G / 0 /1
2 / 1:1-4 / 1:H / 1:1.09
```

`optalg` Specifies the optimization algorithm used for geometry and basis set optimizations. For basis set optimization, at the moment, the downhill simplex method of Nelder and Mead [316] is the only available option. A geometry optimization can be carried out by either the Broyden–Fletcher–Goldfarb–Shanno (BFGS) or the simplex algorithm.

Options:

`simplex` the simplex method of Nelder and Mead.

`BFGS` the BFGS algorithm.

Default: `optalg=simplex` for basis set optimization, `optalg=BFGS` otherwise.

Example: To run a geometry optimization with the simplex algorithm set `optalg=simplex`

`optmaxit` Maximum number of iteration steps allowed in a geometry or basis set optimization. If the simplex algorithm is used, i.e., `optalg=simplex`, the maximum number of function evaluations is also controlled by the parameter `optmaxit`: it is set to 15×`optmaxit`. If the optimization is terminated with a message "the maximum number of function evaluation is exceeded", then you can increase the value of `optmaxit` appropriately.

Options: <*any positive integer*>

Default: `optmaxit=50`

Example: to allow 60 iteration steps set `optmaxit=60`

`optetol` Convergence threshold for geometry or basis set optimization. If the simplex algorithm is used, i.e., `optalg=simplex`, the optimization is terminated when the energy difference (in $E_h$) becomes less then this value and the `optstol` criterion is also fulfilled. In addition to the criterion for the gradient (`optgtol`) and the step-size (`optstol`) the energy change between the cycles is also monitored. For a successful geometry optimization it is required that the `optgtol` criterion is satisfied and either the energy difference between the last two steps becomes less than this value (in $E_h$) or the `optstol` criterion is met.

Options: <*any positive real number*>

Default: `optetol=1e-6`

Example: for a convergence threshold of $5 \cdot 10^{-6}$ $E_h$ set `optetol=5e-6`

`optex` Serial number of the excited state (excluding the ground state) for which the geometry optimization or property calculation is performed. See also keywords `nsing`, `ntrip`, and `nstate`.

Options: <*any positive integer*> that is smaller than the total number of states set by keywords `nsing`, `ntrip`, and `nstate`.

Default: `optex=nstate`-1, that is, the geometry optimization will be performed for the highest excited state considered.

Example: if you set

```
calc=CIS
nsing=3
optex=1
```

for a closed-shell molecule, the ground-state and two excited-state roots will be determined in each geometry optimization step, but the gradients will be calculated for the lowest singlet excited state.

`optgtol` Convergence threshold for geometry optimization, upper limit (in $\mathrm{E}_h$/bohr) for the maximum gradient component. For a successful geometry optimization this criterion must be fulfilled.

Options: <*any positive real number*>

Default: `optgtol=1e-4`

Example: for a convergence threshold of $3 \cdot 10^{-4}$ $\mathrm{E}_h$/bohr set `optgtol=3e-4`

`optstol` Convergence threshold for geometry or basis set optimization. For the latter the optimization is terminated when the maximum change in the parameters becomes less then this value and the `optetol` criterion is also fulfilled, for the former this criterion is met when the maximum step-size from the previous step (in bohr) is lower than this value. The geometry optimization is terminated successfully if, in addition to the `optgtol` criterion obeyed, either this criterion is met or the energy difference between the last two steps becomes less than `optetol`.

Options: <*any positive real number*>

Default: `optstol=1e-3` for a basis set optimization, `optstol=1e-4` otherwise.

Example: to set a threshold of $10^{-5}$ bohr for a geometry optimization type `optstol=1e-5`

`orblocc` Specifies what type of orbital localization is performed for the core molecular orbitals.

Options: All the options introduced for keyword `orbloco` also work for `orblocc`, see the description of keyword `orbloco` for details.

Default: `orblocc=orbloco` if `localcc=2013`, or `core=corr` and `localcc`$\neq$`off`; `orblocc=off` otherwise

Example: to localize of core orbitals with the Pipek–Mezey algorithm specify `orblocc=on`

`orbloce` Specifies what type of orbital localization is performed for the molecular orbitals in the case of localization-based embedding calculations.

Options: All the options introduced for keyword `orbloco` also work for `orbloce`, see the description of keyword `orbloco` for details. An additional option is `orbloce=special`, which requires specifying `orblocc`, `orbloco`, and `orblocv`, respectively, in the three subsequent lines after `orbloce`.

Notes:

1. Except for `orbloce=special`, the same option is set for the core and valance orbitals.

2. The keyword also controls the orbital localization in the case of automated link atom handling of ONIOM calculations.
3. `orbloce=special` is the only option to allow the selection of virtual orbitals in the subsequent projector constructions, where all options of `orblocv` can be used, except `pao-subsys`. Note that it is allowed to utilize the full virtual subspace in the high-level SCF run but to use different virtual orbitals during the correlation calculations. See examples 10 and 11 for keyword `embed`.
4. If `orbloce=spade` is utilized, the atoms specified under the keyword `embed` are used to construct the projector.

Default: `orbloce=spade` if `embed`$\neq$`off` and `oniom=off`,
`orbloce=pm` if `oniom`$\neq$`off`,
`orbloce=off` otherwise

Examples:

1. to localize the orbitals with the Pipek–Mezey algorithm specify `orbloce=pm`
2. to use Pipek–Mezey localization for the occupied orbitals and SPADE localization for the virtual orbitals, type
   ```
   orbloce=special
   pm
   pm
   spade
   ```
   note that this setup allows the selection of active virtual orbitals, thus the projector of the embedding calculations will contain frozen virtual orbitals during the high-level SCF run.

`orblocguess` Initial guess for the orbital localization.

Options:

`cholesky` Guess orbitals are calculated by the Cholesky decomposition of the one-particle density matrix [317].

`restart` Guess orbitals for the localization are read from the `MOCOEF.LOC` file produced in a previous run where orbital localization was performed.

`read` Orbitals are read from the `MOCOEF.LOC` file and directly employed at later steps of the calculation without any change. The locality of the orbitals is not checked and the orbital localization is skipped entirely.

Note: The combination of `orblocguess=restart` (or `orblocguess=read`) with `scfiguess=off` can be particularly useful if the result files of the converged SCF iteration and orbital localization steps are available and only the local correlation step should be repeated with different settings. See also the description of `scfiguess=off` for this option.

Default: `orblocguess=cholesky`

Example: to speed up the orbital localization by using the localized orbitals of a previous calculation as guess specify `orblocguess=restart`.

`orbloco` Specifies what type of orbital localization is performed for occupied molecular orbitals.

Options:

`off` No orbital localization.

`boys` Boys localization is performed [318].

`pm` Pipek–Mezey localization is performed [319].

`IBO` intrinsic bond orbitals of Knizia are constructed [320].

`gboys`$<m>$ generalized Boys localization by Jansík *et al.* [321], where $m$ is a positive integer defining the power of orbital variance. Note that for $m = 1$, the Boys localization is retrieved.

`cholesky` localized orbitals are calculated by the Cholesky decomposition of the one-particle density matrix [317].

`spade` the SPADE fragment localization of Claudino and Mayhall [296].

Default: `orbloco=off` in the general case, `orbloco=boys` for local correlation calculations

Note: In the case of `orbloco=spade`, a list of atoms is read after the line `orbloco` in the following format: $< n_1 >$,$< n_2 >$,...,$< n_k >$-$< n_l >$,..., where $n_i$'s are the serial numbers of the atoms. Serial numbers separated by dash mean that the atomic orbitals of atoms $< n_k >$ through $< n_l >$ are used in the projector construction. Note that the numbering of the atoms must be identical to that used in the Z-matrix or Cartesian coordinate specification, but dummy atoms must be excluded.

Examples:

1. to carry out Pipek–Mezey localization for the occupied orbitals, type `orbloco=pm`
2. to carry out generalized Boys localization with $m = 2$ for the occupied orbitals, type `orbloco=gboys2`
3. Consider three water molecules, denoted by A, B, and C, whose atoms have identifiers of 1-3, 4-6, and 7-9, respectively. To obtain orbitals that are localized on molecule B, type

```
orbloco=spade
4,5-6
```

Note that simultaneously one obtains occupied orbitals which are localized on molecules A and C.

`orblocv` Specifies what type of orbital localization is performed for virtual molecular orbitals.

Options: All the options introduced for keyword `orbloco` excepting `IBO` also work for `orblocv`, see the description of keyword `orbloco` for details. In addition, two more options are available:

`pao` Projected atomic orbitals (PAO)

`pao-subsys` Subsystem-specific projected atomic orbitals

Notes:

1. `orblocv=pao` is only an available option for local correlation calculations.
2. `orblocv=pao-subsys` generates virtual orbitals that are localized on a specified list of atoms. In this case, a list of atoms is read after the line `orblocv` in the

following format: $< n_1 >$,$< n_2 >$,...,$< n_k >$-$< n_l >$,..., where $n_i$'s are the serial numbers of the atoms. Serial numbers separated by dash mean that the atomic orbitals of atom $< n_k >$ through atom $< n_l >$ are used to calculate the norm of the PAOs. Note that the numbering of the atoms must be identical to that used in the Z-matrix or Cartesian coordinate specification, but dummy atoms must be excluded. Note also that those PAOs whose norm is below `pao_subsys_tol` are dropped.

3. If `orblocv=pao-subsys` and `embed`≠`off`, then the required atom list is read under the keyword `embed`, but this can be overridden if a list of atoms under the keyword `orblocv` is specified. See examples 10 and 11 for keyword `embed`.
4. If `orblocv=pao-subsys` and `molden`≠`off`, then the constructed PAOs are written to the end of the `MOLDEN` file, while the coefficients of the rest of the virtual orbitals are filled with zeros. Note that the number of the latter orbitals equals with the number of dropped PAOs.

Default: `orblocv=off` in the general case, `orblocv=pao` for local correlation calculations

Examples:

**1.** to carry out Boys localization for the virtual orbitals type `orblocv=boys`

**2.** To use projected atomic orbitals that are localized on atoms 2, 4 and 6-10, type
`orblocv=pao-subsys`
2,4,6-10

**3.** To partition the virtual space of the water molecules in the 3rd example of `orbloco`, type
`orblocv=spade`
4,5-6

`osveps` Threshold for the occupation numbers of orbital specific virtual orbitals (OSVs) used at the evaluation of pair correlation energies in local MP2 and dRPA calculations. See the description of keyword `wpairtol` for more details.

Options:

`off` OSVs are not constructed and not dropped

<*any real number in the [0,1] interval*> Orbitals with occupation numbers smaller than this number will be dropped.

Default: `osveps=1e-3` for `localcc=2015`, `osveps=off` for `localcc`≥`2016`

Example: to set a threshold of $10^{-4}$ type `osveps=1e-4`

`ovirt` This keyword controls the cost reduction approaches based on natural orbital (NO) or optimized virtual orbitals (OVOs) techniques. For a ground-state correlation calculation, if this keyword is set, the virtual MOs will be transformed to MP2 NOs or OVOs [322]. Subsequently the virtual space will be truncated on the basis of the populations of the orbitals, which can be controlled by keywords `eps`, `lnoepsv`, and `ovosnorb`. See Ref. 21 for more details.

Options:

`off` The virtual MOs are not changed.

`MP2` MP2 NOs will be used.

`OVOS` Optimized virtual orbitals will be used.

`PPL` MP2 NOs are used and the PPL+ and (T+) corrections [55] are calculated.

Note: If `ccprog=mrcc` no correction for the dropped NOs is calculated.

Default: `ovirt=MP2` for reduced-cost correlated calculations (that is, if the `FNO-` prefix is added to the acronym of the selected method), `ovirt=off` otherwise

Example: to use MP2 NOs for a ground-state CC calculation give `ovirt=MP2`

`ovltol` Tolerance for the eigenvalues of the AO overlap matrix. Eigenvectors corresponding to the eigenvalues lower that `ovltol` will be removed to cure the linear dependence of the AO basis set.

Options:

*<any positive real number or zero>* This number will be used as the threshold for the eigenvalues of the overlap matrix.

Default: `ovltol=1e-7`

Example: to keep all the basis functions set `ovltol=0.0`

`ovosnorb` Specifies the retained percentage of virtual orbitals in an optimized virtual orbitals (OVOs) calculations. `ovosnorb` % of virtual orbitals will be retained.

Options: *<any number between 0 and 100>*

Default: `ovosnorb=80.0`

Example: to retain only 70 % of the virtuals give `ovosnorb=70.0`

`pao_subsys_tol` This keyword sets the threshold on the norm of the retained nonredundant projected atomic orbitals (PAOs) during the second screening step of PAO construction. After the PAOs are constructed and filtered based on the norm threshold parameter `pao_tol`, a truncation is applied to the retained PAOs to obtain nonredundant orbitals. In this latter step, the norm of the PAOs is calculated using the atomic orbitals of a list of specified atoms, and PAOs whose norm is lower than `pao_subsys_tol` are dropped. Note that the number of constructed nonredundant PAOs is more sensitive to the `pao_subsys_tol` parameter than to `pao_tol`. See also keyword `orblocv`.

Options:

*<any positive real number in the [0,1] interval >*

Default: `pao_subsys_tol=1e-3`

Example: To keep a fewer number of virtual PAOs, use `pao_subsys_tol=1e-2`

`pao_tol` This keyword sets the threshold on the norm of the retained projected atomic orbitals (PAOs) during the first screening step of the PAO construction process. The norm of the PAOs is calculated using the atomic orbitals of a list of specified atoms, and PAOs whose norm is lower than `pao_tol` are dropped. Note that the number of constructed nonredundant PAOs is more sensitive to the `pao_subsys_tol` parameter than to `pao_tol`. See also keyword `orblocv`.

Options:

*<any positive real number in the [0,1] interval >*

Default: `pao_tol=1e-3`

Example: To keep a smaller number of PAOs, use `pao_subsys_tol=1e-2`

`pcm` If this keyword is set, solvent effects will be considered employing the polarizable continuum model (PCM) [94] via an interface to the PCMSolver library [95–97]. See also keywords `pcm_*` below.

Options:

`off` No solvent effects will be considered

*<solvent>* Solvent effects will be considered with the corresponding solvent. The name of the solvent should be given as defined in PCMSolver, such as `Water`, `Methanol`, `Acetonitrile`, . . . , see the manual of the PCMSolver [97]. Spaces in the names must be replaced by underscores, e.g., `Propylene_Carbonate` instead of `Propylene Carbonate`.

Default: `pcm=off`

Examples:

1. To perform a calculation with water as solvent set `pcm=Water`
2. To perform a calculation with a manually set permittivity of 78.39, add the following lines:
   `pcm=explicit`
   `pcm_green_eps=78.39`

`pcm_*` These keywords control the PCM calculation and correspond to the keywords of the PCMSolver library. See the documentation of the PCMSolver library for a detailed description [95–97]. These keywords will be passed to PCMSolver without any sanity check. The keywords and the corresponding default values are as follows (Å and $Å^2$ are used for length and area, respectively).

`PCM_Cavity_Type=GePol`

`PCM_Cavity_Area=0.3`

`PCM_Cavity_Scaling=False`

`PCM_Cavity_RadiiSet=UFF`

`PCM_Cavity_NpzFile=`*<empty string>*

`PCM_Cavity_MinRadius=100.0`

`PCM_Medium_SolverType=IEFPCM`

`PCM_Medium_Correction=0.0`

`PCM_Medium_ProbeRadius=1.0`

`PCM_Green_Eps=1.0`

`PCM_Green_Type=UniformDielectric`

Note: In particular cases the average area of the surface partition for the cavity (`PCM_Cavity_Area`) is not appropriate, and you get an error message. In this case, you should change the value of `PCM_Cavity_Area`.

Example: to change the cavity surface area partition to 0.2 Å$^2$ set `PCM_Cavity_Area=0.2`

`popul` This keyword controls the wave function analysis. See also keyword `nto` for the generation of natural transition orbitals (NTOs) and the keyword `espcharge` to generate CHELPG and Merz–Singh–Kollman atomic charges.

Options:

`off` No wave function analysis is performed.

`Mulli` A population analysis is also performed, and Mulliken and Löwdin atomic charges as well as Mayer bond orders are computed [323, 324].

`IAO` In addition to the above parameters, intrinsic atomic orbitals (IAOs) are constructed and IAO partial charges are calculated [320].

`deco` In addition to the parameters listed for `Mulli`, the atomic decomposition of the energy is also computed. This is only implemented for HF, DF-HF, and standard iterative CC methods, such as CCSD, CCSDT, . . . . In the case of the latter, population analysis is not performed.

Default: `popul=Mulli` if `dens` $\neq$ `0`, `popul=off` otherwise

Example: to calculate IAO charges set `popul=IAO`

`pressure` The pressure in Pa at which the thermodynamic properties are evaluated (see also keyword `freq`).

Options: <*any positive integer*>

Default: `pressure=100000`

Example: for 1 atm set `pressure=101325`

`pssp` Turns on pseudo-spectral (COSX) exchange matrix build for SCF calculations. It uses DFT grids as integration grid (see keywords `agrid` and `agrid_pssp`). By default, a smaller grid (specified with `agrid_pssp_sm`) is used to get a self-consistent density which is then used to get a more accurate result on the larger grid (specified with `agrid_pssp`). The result is further enhanced with a first order correction [50]. The COSX algorithm can be used for both Hartree–Fock and hybrid DFT calculations.

Options:

`on` The pseudo-spectral algorithm is used for exchange matrix formation.

`off` The pseudo-spectral algorithm is turned off.

Default: `pssp=off`

Notes: One can perform a single grid COSX calculation by setting `dual_df=off`. For more information, see keyword `dual_df` and the examples below. The radial grid of the integration grid can be specified by `rgrid`, while the accuracy of the grid can be tuned by the `grtol` keyword. Keep in mind, however, that these keywords control the DFT grid as well.

Examples:

1. DFT calculation with the PBE0 functional and COSX:
   ```
   basis=cc-pVTZ
   calc=PBE0
   scfalg=direct
   pssp=on
   ```
2. DFT calculation with the PBE0 functional and COSX using a single grid:
   ```
   basis=cc-pVTZ
   calc=PBE0
   scfalg=direct
   pssp=on
   dual_df=off
   ```
3. DFT calculation with the PBE0 functional and COSX using custom grid options:
   ```
   basis=cc-pVTZ
   calc=PBE0
   scfalg=direct
   pssp=on
   agrid_pssp=LD0006-LD0350
   agrid_pssp_sm=LD0006-LD0110
   ```

`ptfreq` Frequency of the perturbation for frequency-dependent properties in atomic units (available only with CFOUR). See Refs. 11 and 13 for more details.

Options: <*any real number*>

Default: `ptfreq=0.0`

Example: to set a frequency of 0.1 a.u. give `ptfreq=0.1`

`ptthreads` Sets the number of outer OpenMP threads in program `ccsd` while calculating the (T) correction.

Options: <*any positive integer*>

Default: `ptthreads=2`

Example: to reduce the memory requirement of a CCSD(T) calculation by turning off nested OpenMP parallelization set `ptthreads=1`

`qmmm` This keyword tells MRCC that a QM/MM calculation is performed and the point charges included in the input file must be processed. This keyword is automatically added to the `MINP` file by the MM program conducting the QM/MM calculations, and you do not need to bother with it. Use this keyword in the only case if you want to add point charges manually.

Options:

`off` QM/MM calculation is not performed and no point charges are added.

`Amber` Currently this is the only option, the AMBER MD code will be used for the QM/MM calculation or point charges will be added.

Note: Point charges can be manually added to the end of the input file in the following format:

`pointcharges`
$<$*number of point charges*$>$
$< x_1 > < y_1 > < z_1 > < q_1 >$
$< x_2 > < y_2 > < z_2 > < q_2 >$
$\vdots$

where $x_i$, $y_i$, and $z_i$ are the Cartesian coordinates and $q_i$ is the charge for point charge $i$. The charge must be given in a.u., while for the coordinates the same unit must be used as for the specification of the molecular geometry.

Default: `qmmm=off`

Example: to add two point charges with coordinates (0, 0, 1) and (0, 1, 0) a.u. (provided that the geometry is also given in bohr) and charges of 0.5 and -0.5 a.u. `qmmm=Amber` should be set anywhere in the `MINP` file, and the following lines should be added to the end the file:

```
pointcharges
2
0.0 0.0 1.0 0.5
0.0 1.0 0.0 -0.5
```

`qro` Switches on the computation of quasi-restricted orbitals (QROs) determined from UHF/UKS orbitals [51, 325]. QROs are useful for the cases where the convergence of the ROHF/ROKS iterations is problematic but restricted open-shell orbitals are desired/needed, such as in the case of our open-shell local correlation methods. The starting point of determining the QROs is an UHF/UKS solution, which may be simpler to find than the ROHF/ROKS one. The QROs are constructed as the eigenvectors of the total density matrix of the unrestricted computation [51, 325]. The QROs obtained in this way with occupation numbers close to 2, 1, and 0 are selected to be doubly, singly, and not occupied in the QRO determinant, respectively, which becomes an eigenfunction of $\hat{S}^2$ by construction.

Options: `on` or `off`

Default: `qro=on` for `localcc`≥`2021` and UHF/UKS computation, and `qro=off` otherwise.

Note: The `scfguessdens=uhf` option can also be useful to utilize satisfactory UHF/UKS densities for the initial guess construction of ROHF/ROKS computations.

`qscf` Use this option to carry out quadratic RHF, UHF, ROHF, RKS, and UKS SCF calculations and also to select the algorithm used for MCSCF calculations. One can use either (quasi-)Newton or trust region iterations with optional line search.

Options:

`off` No quadratic SCF, conventional SCF algorithm will be

`on` Equivalent to `AugHessG` executed.

`Newton` Simple Newton iteration.

`NewtonL` Simple Newton iteration with line search.

`AugHess` Trust region method with augmented Hessian algorithm without line search. The trust radius is updated according to the scheme described in Ref. 100.

`AugHessM` Trust region method with augmented Hessian algorithm using line search. The trust radius update scheme is similar to the above one but uses different coefficients.

`AugHessL` Trust region method with augmented Hessian algorithm using line search. The trust radius is updated as described in Ref. 100, but this method takes into account the step length found by the line search.

`AugHessG` Trust region method with augmented Hessian algorithm using line search. The trust radius update scheme takes into account the change of the gradient in the consecutive iterations.

`BFGS` Quasi-Newton method utilizing the BFGS algorithm (see also keywords `bfgsmem` and `bfgstol`).

Note: The simple (quasi-)Newton iteration schemes are only efficient in the vicinity of a minimum and not recommended in the general case. We recommend the use of the AugHessG or AugHessL options in difficult cases.

Default: `qscf=AugHessG` for MCSCF calculations, `qscf=off` otherwise

Example: UKS calculation with the B3LYP functional using the AugHessG algorithm:

```
calc=B3LYP
scftype=UHF
qscf=AugHessG
```

`redcost_exc` This keyword controls the cost reduction approaches based on natural orbital (NO) and natural auxiliary function (NAF) techniques for excited states. For a (spin-scaled) CIS($D_\infty$), ADC(2), and CC2 calculation, if `redcost_exc`$\neq$`off`, the reduced-cost approach of Refs. 35 and 37 is invoked, and truncated state-averaged MP2/CIS(D) NOs as well as NAFs will be used. The `lnoepso`, `lnoepsv`, `naf_cor`, and `naftyp` keywords will be automatically set depending on the selected option for `redcost_exc`. For CIS, TD-HF, TDA, TD-DFT calculations the NAF approximation will be invoked if `redcost_exc=8`.

Options:

`off` The cost reduction techniques will not be used.

`on` Reduced-cost calculation will be executed with default settings (see also Table 8).

`cust` Reduced-cost calculation will be executed with customized truncation thresholds. The threshold for the complete MO space NAFs (CS-NAFs), CIS coefficients, orbital energies, and linear dependency must be specified in the subsequent line, respectively, as
*<threshold 1> <threshold 2> <threshold 3> <threshold 4>*
The default values of these thresholds are 0.1 a.u., 0.35, 0.15 a.u., $10^{-7}$, respectively. These are used with if any other option is chosen. See Ref. 37 for the detailed description of these parameters.

*<positive integers from 1 to 10>* See Table 8.

Default: `redcost_exc=off`

Examples:

1. Reduced-cost ADC(2) calculation for the lowest singlet excited state with the default settings proposed in Ref. 37:
   `calc=ADC(2)`
   `nsing=2`
   `redcost_exc=on`
2. Reduced-cost TD-DFT calculation with the PBE0 functional for the lowest 3 singlet excited states of a molecule:
   `calc=PBE0`
   `redcost_exc=8`
   `nstate=4`

`redcost_tddft` This keyword controls the use of local fitting domains for reduced-scaling density fitting CIS, TD-HF, and TD-DFT calculations. See Ref. 40 for more details.

Options:

`off` The scaling reduction techniques will not be used.

`on` Reduced-scaling calculation will be executed with a completeness criterion of 0.985 for the wave function truncation.

<*any real number in the [0,1] interval*> This number will be used as the completeness criterion.

Default: `redcost_tddft=off`

Example: to set a completeness criterion of 0.99 type `redcost_tddft=0.99`

`refdet` The reference determinant (Fermi-vacuum) for CI/CC calculations can be specified using this keyword. By default the reference determinant is identical to the HF determinant, but sometimes it is necessary to change this.

Options:

`none` The reference determinant is identical to the HF determinant.

`serialno` Using this option one can define the occupation of the correlated orbitals in the reference determinant specifying their serial numbers. This option requires three more lines. In the first line the serial numbers of the doubly-occupied orbitals must be given, while in the second and third lines those orbitals should be specified which are singly-occupied by an alpha or a beta electron, respectively. For the format of these lines see the description of the `serialno` option of the `active` keyword. For relativistic calculations the occupation of the spinors (i.e., not that of the Kramers-pairs) should be given. For technical reasons all electrons are treated as alpha electrons and the serial numbers of the occupied spinors must be given in the second line, the first and third lines must be left blank.

`vector` Using this option one can set the occupation numbers for each correlated orbital. In the subsequent line an integer vector should be supplied with as many elements as the number of correlated orbitals (correlated spinors for relativistic calculations, not Kramers-pairs!). The integers must be separated by spaces. In

Table 8: Options for keyword `redcost_exc`. CS-NAF – complete MO space NAFs are used, NO – frozen natural orbitals are used, Can. – the NO space is augmented with canonical virtual orbitals, RS-NAF – restricted NO space NAFs are used. See Ref. 37 for more details.

| Option | CS-NAF | NO | Can. | RS-NAF | Note |
|---|---|---|---|---|---|
| `off` | no | no | no | no | Default |
| `on` | yes/no | yes | yes | yes | Equivalent to `1` or `6` depending on keyword `localcc` |
| `cust` | yes | yes | yes | yes | Customized thresholds |
| `1` | yes | yes | yes | yes | The approach presented in Ref. 37, default if `localcc=off` and `redcost_exc=on` |
| `2` | no | yes | no | yes | The approach presented in Ref. 35 |
| `3` | no | no | no | yes | The approach presented in Ref. 35 without NOs |
| `4` | no | yes | no | no | The approach presented in Ref. 35 without NAFs |
| `5` | no | yes | yes | no | |
| `6` | no | yes | yes | yes | Default if `localcc=2016` and `redcost_exc=on` |
| `7` | yes | yes | no | yes | |
| `8` | yes | no | no | no | For reduced-cost CIS, TD-HF, TDA, TD-DFT, . . . |
| `9` | yes | yes | yes | no | |
| `10` | yes | yes | no | no | |

the case of non-relativistic calculations type 2 for doubly-occupied orbitals, 1 for open-shell orbitals with alpha electron, -1 for open-shell orbitals with beta electron, 0 otherwise. In the case of relativistic calculations type 1 for each occupied spinor, 0 otherwise.

Notes:

1. Frozen orbitals must not be considered here in any case.
2. If the MO integrals are taken over from another program, the numbering of orbitals may be different from that of the parent program. Here the order of MOs: doubly occupied, open shell, virtual; and in each of this blocks the MOs are reordered according to the orbital energies (natural orbital occupations in the case of active MCSCF orbitals).
3. If the MO integrals are taken over from another program, and this line is omitted, the program will fill the orbitals with electrons from the bottom automatically. In this manner we do not need this line for closed shells or a doublet reference determinant, but, e.g., for high-spin states the Fermi vacuum must be defined here.
4. For relativistic calculations (DIRAC interface) this line is always required. The spinors are symmetry-blocked according to the Fermion irreps of the corresponding double group. Complex conjugate irreps follow each other. Within each irrep the spinors are numbered according to orbital energies. Please note that this

line is automatically printed by the `dirac_mointegral_export` program, and you do not have to do it by hand. However, for technical reasons, always a closed-shell occupation is generated, and you may need to remove or add some electrons.

Default: `refdet=none`, that is, the reference determinant is identical to the HF determinant.

Examples:

1. We have 20 correlated orbitals, 10 electrons, and we are interested in a high-spin triplet state. Suppose that orbitals 1 to 4 are doubly-occupied while orbitals 5 and 6 are singly occupied by alpha electrons. Using the `serialno` option the input should include the following four lines (note the blank line at the end):
   ```
   refdet=serialno
   1-4
   5,6

   ```
2. The same using the `vector` option:
   ```
   refdet=vector
   2 2 2 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   ```
3. We perform a relativistic calculation for the Be atom with 20 correlated spinors. We have 6, 6, 4, and 4 spinors in the four Fermion irreps, $\mathrm{E}_{1/2g}$, $\mathrm{E}_{-1/2g}$, $\mathrm{E}_{1/2u}$, and $\mathrm{E}_{-1/2u}$ of the $C^*_{2h}$ double group, respectively, and two occupied spinors in both of the gerade irreps. Thus using the `vector` option the occupation vector should be given as:
   ```
   refdet=vector
   1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
   ```
4. The same using the `serialno` option (note the blank lines):
   ```
   refdet=serialno

   1,2,7,8

   ```

`rest` Use this keyword to restart canonical (i.e., not local) CI and CC calculations from previously calculated wave function parameters (cluster amplitudes, CI coefficients, $\lambda$ amplitudes, etc.) if `ccprog=mrcc`. For restarting local correlation calculations see keyword `lccrest`.

Options:

`1` The program restarts from the previously calculated parameters.

`2` The program executes automatically the lower-level calculations of the same type consecutively (e.g., CCSD, CCSDT, and CCSDTQ if CCSDTQ is requested) and restarts each calculation from the previous one (this is only available for energy calculations).

`3` Same as `rest=1`, however, only selected roots from the previous calculation will be used as initial guess. The serial number of the roots must be specified in the subsequent line as

$< n_1 >< n_2 >< n_3 > \ldots$
where $< n_i >$ is the serial number of the states. The number of states given here must be equal to `nstate` or `nsing` + `ntrip`. Please note that the ground state solution is not automatically selected, it should also be given here if needed. It is recommended to use root following (`diag=follow`) together with this option.

4 Same as `rest=2` but the initial vectors are selected as in the case of `rest=3`.

Notes: use the restart option, e.g.,

1. after system crash.
2. if the iteration procedure did not converge in the given number of steps.
3. for geometry optimization.
4. for potential curve calculations.
5. if you are interested in high-order CC/CI energies. Then it is worth restarting the calculations with higher excitations using the converged vectors of the same method including lower excitations, e.g., CCSDT using the converged CCSD amplitudes, CCSDTQ using the CCSDT amplitudes, and so on. With this trick 1-3 iteration steps can be saved usually, but more ones in the case of strong static correlation (i.e., large cluster amplitudes). Use exclusively `rest=2` for this purpose (that is, not `rest=1`)!
6. if you are interested in calculating the energies for all methods in a hierarchy (e.g., executing all CC methods up to CCSDTQP). Use exclusively `rest=2` for this purpose (that is, not `rest=1`)!
7. to generate brute-force initial guess for excited-state calculations (`rest=3` or `4`). That is, if you do not want to bother with the initial guess for excited states, but you know approximately the energy of the excited states, then execute a low-level method for many roots. Use LR-CCS (`calc=CCS`) and CIS (`calc=CIS`), respectively, for higher-order LR-CC and CI calculations. Select the desired roots on the basis of their energies, and use them as initial guess in high-level calculations. (For other options for the initial guess for excited-state calculations see keyword `ciguess`.)
8. Please note that the program always needs the file `fort.16` from the previous calculation for the restart and also `fort.17`, if more than one root is sought or for geometry optimization.

Default: `rest=0`

Examples:

1. to restart a CC calculation after power failure set `rest=1`
2. to restart a LR-CCSD calculation using the first, third, and fifth roots of a previous LR-CCS calculation the input should include the following two lines:
```
rest=3
1 3 5
```

`rgrid` Specifies the radial integration grid for DFT calculations. See also the description of keywords `grid`, `agrid`, and `grtol`.

Options:

`Log3` the Log3 quadrature of Mura and Knowles [305]

`GC` Gauss–Chebyshev quadrature. A modified version of the mapping function of Ref. 303 is employed: the function is scaled by the atomic scaling parameters of Becke [174].

`GCTA` Gauss–Chebyshev quadrature of Treutler and Ahlrichs (M4 quadrature of Ref. 175)

`EM` Euler–Maclaurin quadrature [304]

Note: For the grids of Treutler–Ahlrichs (`grid=TA*`), the number of grid points is fixed (see Ref. 175). Otherwise, the number of grid points is calculated by the max{20, 5*(3*`grtol`/2+$i$+10)} formula with $i$ as the number of the row in the periodic table where the atom is located [303]. To change the number of radial integration points, set the value of `grtol`. If an adaptive grid is used (see keyword `grid`), the number of radial grid points is determined automatically so that the error in the radial integrals will not be larger than $10^{1-\texttt{grtol}}$. In this case, the above value is just the maximum allowed number of points.

Default: `rgrid=Log3`

Examples:

1. to use the Euler–Maclaurin scheme set `rgrid=EM`
2. to use the default Log3 grid with more points set `grtol=12`

`rism` Specifies the 3D-RISM settings of an EC-RISM calculation.

Options:

`off` No EC-RISM calculation.

`on` EC-RISM with default settings (TIP3P water, PSE-3 closure, gridspace=0.3 Å, buffer=15 Å, CHELPG charge, see Ref. 56 for further details).

<*any options of the* RISM3D.SNGLPNT *program*> Other 3D-RISM settings can be used by simply passing the RISM3D.SNGLPNT commands to this keyword. For example, the above default setting written in this mode is `rism=--xvv solvent.xvv --grdspc 0.3,0.3,0.3 --tolerance 1E-5 --closure pse3 --buffer 15 --maxstep 10000`, where `solvent.xvv` is the solvent susceptibility file provided by the user from an earlier 1D-RISM calculation with TIP3P water and PSE-3 closure. The `--pdb`, `--prmtop`, `--rst`, `--volfmt`, `--chgdist`, `--pc+`, and `--gf` settings are automatically set by MRCC, and passing them to the keyword could cause problems. `--xvv` must be defined in these cases, while the `--closure`, `--grdspc`, and `--buffer` settings should be considered, because these can alter the results substantially. See the AMBERTOOLS manual for further details on the various RISM3D.SNGLPNT settings.

Default: `rism=off`

Notes:

1. If `rism`≠`off`, and no `espcharge` is specified, `espcharge=chelpg` is the default setting.

2. The solvent susceptibility file (`solvent.xvv`) can be created by modifying the automatically created RISM1D input of a default EC-RISM run (`solvent.inp`). In this case, an `mdl` file of the solvent is required, either from the AMBER library or from the user. See the AMBERTOOLS manual for further details on the solvent susceptibility file preparation with RISM1D.
3. The EC-RISM calculation can be restarted by providing an `ATCHARGE.GUESS` file, containing only the corresponding atomic charges in one column. In this case, for the first 3D-RISM calculation, the provided charges will be used, not the ones determined from the gas-phase calculation.
4. If the studied system has atoms without GAFF2 force field parameters, you must provide a file called `FRCMOD` which contains the missing Lennard–Jones parameters by listing each atom with their van der Waals radius ($r*$) and well-depth ($\epsilon$). Here is an example for the file structure (the values are not recommendations):
```
NONBON
Si 1.778 0.015
Fe 1.800 0.010
```

Examples:

1. to use EC-RISM with default settings give `rism=on`
2. to use EC-RISM with Kovalenko–Hirata closure (KH), ethanol solvent (`ethanol.xvv`), 10 Å buffer and 0.5 Å gridspace set `rism=-xvv ethanol.xvv --closure KH --buffer 10 --grdspc 0.5,0.5,0.5`, where the solvent susceptibility file of the ethanol (`ethanol.xvv`) is in the same folder as the `MINP` file.

`rohfcore` Specifies the type of the frozen ROHF or ROKS orbitals if `rohftype=semicanonical` and `core=frozen`. See also the description of keyword `rohftype`.

Options:

`standard` The orbitals will be frozen before the semi-canonicalization of the orbitals, that is, standard ROHF (ROKS) orbitals will be frozen.

`semicanonical` The orbitals will be frozen after the semi-canonicalization of the orbitals, that is, semicanonical ROHF (ROKS) orbitals will be frozen.

Notes:

1. For ROHF-based correlation calculations CFOUR and MOLPRO use semicanonical and standard frozen orbitals, respectively. If you want to compare the results computed with MRCC to those obtained with the above programs, use this keyword to select the same type of frozen ROHF orbitals.
2. In the case of local correlation calculations standard ROHF (ROKS) orbitals are frozen for theoretical reasons. If results of local and canonical correlation calculations are compared, standard ROHF (ROKS) orbitals should be frozen in the canonical calculations as well.

Default: `rohfcore=standard` for local correlation calculations, `rohfcore=semicanonical` otherwise.

Example: to use standard ROHF core orbitals for canonical correlation methods set `rohfcore=semicanonical` `rohftype` Specifies the type of the ROHF or ROKS orbitals. See also the description of keyword `scftype`.

Options:

- `standard` Standard ROHF (ROKS) orbitals obtained by diagonalizing the ROHF (ROKS) Fock-matrix.
- `semicanonical` Semicanonical ROHF (ROKS) orbitals obtained by separately diagonalizing the alpha and beta UHF (UKS) Fock-matrices constructed using the converged ROHF (ROKS) orbitals.

Notes:

1. `rohftype=semicanonical` is required for perturbative CC methods if ROHF orbitals are used, otherwise the expressions for the perturbative corrections are not correct. Iterative CC and CI methods are invariant to the choice of ROHF orbitals (if all electrons are correlated).
2. It is very important to give this keyword if Mrcc is used together with another code and ROHF orbitals are used since this keyword tells Mrcc what type of ROHF orbitals are taken over from the other code.

Default: for `ccprog=ccsd` `rohftype=semicanonical`; for `ccprog=mrcc` `rohftype=standard` for iterative CC and CI methods, `rohftype=semicanonical` for perturbative methods.

Example: to use semicanonical ROHF orbitals for iterative CC methods give `rohftype=semicanonical`

`scf_conv` Specifies the parameter set for SCF acceleration techniques, which includes Pulay's DIIS algorithm, the level shifting of orbital energies, and the density matrix damping schemes. See also Table 9 for the specific parameters.

Options:

- `normal` initiates the `normal` set of SCF acceleration parameters
- `fast` initiates the `fast` set of SCF acceleration parameters

Notes:

1. In the case of `scf_conv=normal`, damping is turned off for Hartree–Fock calculations.
2. The SCF acceleration parameters can be overwritten manually by specifying the above keywords regardless of the options of `scf_conv`.

Default: `scf_conv=normal`

Example: if the SCF convergence may be less problematic to reach, type `scf_conv=fast`

`scfalg` Specifies what type of SCF algorithm is to be used.

Options:

- `disk` Conventional SCF algorithm, two-electron integrals are stored on disk.
- `direct` Direct SCF algorithm, two-electron integrals are recalculated in each iteration step.

Table 9: Parameter table for SCF acceleration techniques

| Keyword | `fast` | `normal (DFT)` | `normal (HF)` |
|---|---|---|---|
| `scfdiis_start` | `2` | `2` | `2` |
| `scfdiis_end` | `scfmaxit` | `scfmaxit` | `scfmaxit` |
| `scfdiis_dtol` | `0.2` | `0` | `0` |
| `scfdiis_watch` | `off` | `off` | `off` |
| `scfext` | `10` | `10` | `10` |
| `scfdiis_step` | `1` | `1` | `1` |
| `scfdamp_mode` | `3` | `1` | `off` |
| `scfdamp` | `0.5` | `0.7` | `0` |
| `scfdamp_maxfact` | `0.9` | `0.7` | `0` |
| `scfdamp_minfact` | `0.1` | `0.7` | `0` |
| `scfdamp_dampstep` | `0.1` | `0` | `0` |
| `scfdamp_dtol` | `0.5` | `0` | `0` |
| `scfdamp_end` | `scfmaxit` | `3` | `scfmaxit` |
| `scflshift` | `0.1` | `0.2` | `0.2` |
| `scflshift_gaptol` | `0.1` | `0.2` | `0.2` |
| `scflshift_dtol` | `1d-2` | `0` | `0` |
| `scflshift_end` | `scfmaxit` | `8` | `8` |

`auto` Based on the size and geometry of the molecule the program will automatically select the more efficient one from the above options.

`locfit1` Density-fitting direct SCF algorithm using local fitting domains for the exchange contribution [31, 158]. See also keyword `excrad`.

`locfit2` Same as locfit1 but, in addition to the auxiliary functions, domains are also constructed for the AOs and MOs. This is efficient only for very large molecules.

Default: `scfalg=auto`

Notes: In the case of `locfit1` and `locfit2` calculations, the orbital localization method can be specified with the `scfloc` keyword.

Example: to run direct SCF, add `scfalg=direct`

`scfdamp` Specifies whether damping of the SCF density matrices is performed.

Options:

`off` No damping.

<*any real number in the [0,1] interval*> In each SCF iteration cycle the new and old SCF density matrices are mixed by factors (1-`scfdamp`) and `scfdamp`, respectively.

`on` Equivalent to `scfdamp=0.7`

Default: `scfdamp=off` for Hartree–Fock, and `scfdamp=0.7` for DFT calculations, and `scfdamp=0.7` when a damping scheme is specified using `scfdamp_mode`.

Notes: In the case of dynamic dampig schemes (`scfdamp_mode=2` or `scfdamp_mode=3`), the keyword `scfdamp` sets the initial damping factor.

Example: to use a damping factor of 0.8 type `scfdamp=0.8`

`scfdamp_dtol` Specifies the tolerance in the RMS of the difference density matrix where the SCF damping is turned off. See also the notes for `scfdamp_mode` if `scfdamp_end` is used with `scfdamp_dtol`.

Options: <*any positive real number*> or `0`

Default: `scfdamp_dtol=0`, that is, the SCF damping is independent of the RMS of the difference density matrix.

Example: to use a density-dependent damping approach, set `scfdamp_dtol=1d-3`, which means that the density damping is switched off when the RMS of difference density matrix is below `1d-3`.

`scfdamp_dampstep` Specifies the constant change in `scfdamp` if `scfdamp_mode=2` or `scfdamp_mode=3` is used.

Options: <*any positive real number*> or `0`

Default: `scfdamp_dampstep=0`, that is, `scfdamp` is not changed in any iteration. If `scfdamp_mode=2` or `scfdamp_mode=3`, then `scfdamp_dampstep=0.05`

Example: to use larger steps in the `scfdamp`, type `scfdamp_dampstep=0.1`

`scfdamp_end` Specifies the last iteration step in which the SCF damping is applied. See also the notes for `scfdamp_mode` if `scfdamp_end` is used with `scfdamp_dtol`.

Options: <*any positive integer*>

Default: `scfdamp_end=3` for DFT calculations, that is, the SCF damping is turned off after the third SCF iteration, `scfdamp_end=scfmaxit` otherwise.

Example: to turn off the SCF damping after iteration step 20, give `scfdamp_end=20`

`scfdamp_minfact` Specifies the smallest value that `scfdamp` can take if `scfdamp_mode=2` or `scfdamp_mode=3` is used.

Options: <*any real number in the [0,*`scfdamp`*] interval*>

Default: `scfdamp_minfact=scfdamp` in the case of `scfdamp_mode=1`, that is, an unchanged damping factor for the static damping approach. If a dynamic damping scheme is specified (`scfdamp_mode=2` or `scfdamp_mode=3`), then `scfdamp_minfact=0.1`, that is, `scfdamp` cannot take a smaller value than 0.1.

Example: to use a minimum factor for `scfdamp`, type `scfdamp_minfact=0.2`

`scfdamp_maxfact` Specifies the largest value that `scfdamp` can take if `scfdamp_mode=2` or `scfdamp_mode=3` is used.

Options: <*any real number in the [*`scfdamp`*,1.0] interval*>

Default: `scfdamp_maxfact=scfdamp` in the case of `scfdamp_mode=1`, that is, an unchanged damping factor for the static damping approach. If a dynamic damping scheme is specified (`scfdamp_mode=2` or `scfdamp_mode=3`), then `scfdamp_maxfact=0.95`, that is, `scfdamp` cannot take a larger value than 0.95.

Example: to restrict the `scfdamp` parameter to a smaller maximum value in the case of dynamic damping, type `scfdamp_maxfact=0.85`

`scfdamp_mode` Specifies which SCF damping scheme is performed.

Options:

- `off` No damping.
- `1` Static damping. The damping factor set by `scfdamp` is unchanged in any iterations.
- `2` Steadily decreasing damping scheme. The damping factor, which is initialized by `scfdamp`, is decreased by `scfdamp_dampstep` in every iteration if the RMS of the gradient decreases. The smallest possible damping factor is `scfdamp_minfact`.
- `3` Dynamically adjusted damping scheme. The damping factor, which is initialized by `scfdamp`, is increased by `scfdamp_dampstep` if the RMS of the gradient increases and it is decreased otherwise by the same magnitude. The damping factor cannot take a larger value than `scfdamp_maxfact` or a lower one than `scfdamp_minfact`.
- `on` Equivalent to `scfdamp_mode=3`

Notes:

1. The damping schemes are applied from the first iteration. The damping can be turned off by specifying an iteration step (`scfdamp_end`), or using a tolerance in the RMS of difference density matrix (`scfdamp_dtol`).
2. If `scfdamp_end` and `scfdamp_dtol` are used together, damping is activated if the iteration number is lower than `scfdamp_end` or the RMS of difference density is larger than `scfdamp_dtol`.
3. Note that when `scfdamp_mode=3` and `scfdamp_dtol` is set, the damping can be switched on again if a larger change in the density matrix occurs. When the damping is re-activated, `scfdamp` is set to `scfdamp_minfact`. The reactivation of the damping of the SCF density matrix is forbidden in the case of `scfdamp_mode=1` and `scfdamp_mode=2`.

Default: `scfdamp_mode=1` for DFT, and `scfdamp_mode=off` for HF calculations, respectively.

Example: to use a steadily decreasing damping factor, set `scfdamp_mode=2`

`scfdiis` Specifies if DIIS convergence acceleration is used in the SCF calculations.

Options: `on` or `off`

Default: `scfdiis=on`

Example: to turn off DIIS convergence accelerator add `scfdiis=off`

`scfdiis_delmax` Specifies the number of times when the DIIS error vector is allowed to be deleted (i.e., the DIIS is restarted) by the `scfdiis_watch` algorithm. See also the description of `scfdiis_watch`.

Options: <*any positive integer*>

Default: `scfdiis_delmax=2`, that is, `scfdiis_watch` turns off if the DIIS error vectors have already deleted twice.

Example: to increase the allowed number of DIIS restarts, type `scfdiis_delmax=5`

`scfdiis_dtol` Specifies the tolerance in the RMS of difference density matrix when the DIIS convergence acceleration is turned on. Note that `scfdiis_start` has higher priority.

Options: <*any positive real number*> or `0`

Default: `scfdiis_dtol=0`, that is, the DIIS procedure is independent from the RMS of difference density matrix.

Notes: Once the RMS of the difference density is lower than `scfdiis_dtol`, the DIIS procedure is applied and it is not switched off even if the SCF density matrix changes significantly.

Example: to turn on the DIIS convergence accelerator after the deviation of the density matrix is small, type `scfdiis_dtol=1d-3`

`scfdiis_end` Specifies the last iteration step in which the DIIS convergence acceleration is applied.

Options: <*any positive integer*>

Default: `scfdiis_end=scfmaxit`, that is, the DIIS procedure is not turned off.

Example: to turn off the DIIS convergence accelerator after iteration step 20 give `scfdiis_end=20`

`scfdiis_start` Specifies the first iteration step in which the DIIS convergence acceleration is applied.

Options: <*any positive integer*>

Default: `scfdiis_start=2`, that is, the DIIS procedure is active from the second iteration.

Example: to turn on the DIIS convergence accelerator in iteration step 5 give `scfdiis_start=5`

`scfdiis_step` Specifies the frequency of DIIS extrapolations. The extrapolation will be carried out in every `scfdiis_step`'th iteration cycle.

Options: <*any positive integer*>

Default: `scfdiis_step=1`, that is, the DIIS extrapolation is performed in each iteration step.

Example: to carry out DIIS extrapolation only in every second iteration step give `scfdiis_step=2`

`scfdiis_watch` Specifies if the DIIS error vectors are deleted when the DIIS error raises at least `scfdiis_wlimit` times in the previous `scfdiis_wrange` iterations. Note that the deletion of error vectors can happen at most `scfdiis_delmax` times.

Options:

`off` The DIIS error vectors will not be deleted.

`on` The DIIS error vectors will be deleted if the above conditions are met.

Default: `scfdiis_watch=off`

Example: to turn on the DIIS error watch algorithm type `scfdiis_watch=on`

`scfdiis_wlimit` Specifies the number of times when the DIIS error is allowed to rise in the previous `scfdiis_wrange` iterations before the vectors are deleted if `scfdiis_watch=on`.

Options: <*any positive integer*>

Default: `scfdiis_wlimit=scfext/2` if scfext is an even number and `scfdiis_watch=(scfext+1)/2` otherwise.

Example: to increase the number of iterations in which the DIIS error is allowed to raise, type `scfdiis_wlimit=20`

`scfdiis_wrange` Specifies the number of SCF iterations in which the DIIS error is checked by the "`scfdiis_watch`" algorithm.

Options: <*any positive integer*>

Default: `scfdiis_wrange=scfext`

Example: to increase the number of iterations in which the DIIS error is checked by the `scfdiis_watch` algorithm, type `scfdiis_wrange=15`

`scfdtol` Convergence threshold for the density matrix in SCF calculations. The RMS change in the density matrix will be smaller than $10^{-\texttt{scfdtol}}$.

Options: <*any integer*>

Default: `scfdtol=scftol+2` for frequency calculations, otherwise `scfdtol=scftol+1` for correlation calculations, `scfdtol=scftol` for SCF calculations

Example: for an accuracy of $10^{-8}$ one must give `scfdtol=8`

`scfext` Specifies the maximum number of Fock-matrices used for the DIIS extrapolation in SCF calculations.

Options: <*any positive integer*>

Default: `scfext=10`

Example: to increase the maximum number of DIIS vectors to 15 give `scfext=15`

`scfguessdens` Communicates the format of the initial guess density stored in the `SCFDENSITIES` file, which is used for `scfiguess=restart`. Upon the completion of the SCF job step in the cases of unrestricted SCF computations and restricted open-shell SCF computations followed by the construction of semi-canonical orbitals, the `SCFDENSITIES` file format is different from that of standard restricted open-shell computations. The `scfguessdens` keyword can be used to perform a restricted open-shell SCF computation using the density matrices of an unrestricted computation in the `SCFDENSITIES` file as an initial guess and vice versa.

Options:

`uhf` The program assumes for `scfiguess=restart` that the `SCFDENSITIES` file is a result of an unrestricted SCF job.

`rhf` The program assumes for `scfiguess=restart` that the `SCFDENSITIES` file is a result of a restricted SCF job.

`rohfscn` The program assumes for `scfiguess=restart` that the `SCFDENSITIES` file is a result of a restricted open-shell computation ended with the construction of semi-canonical orbitals.

`rohfstd` The program assumes for `scfiguess=restart` that the `SCFDENSITIES` file is a result of a standard restricted open-shell computation (without the construction of semi-canonical orbitals).

Default: `scfguessdens` is not set by default assuming that the `SCFDENSITIES` file structure is the same for the job that produced the initial guess density and the job to be restarted from that initial guess density.

`scfiguess` Initial guess for the SCF calculation.

Options:

`sad` Superpositions of atomic densities. For each atom a density-fitting UHF calculation is performed, and the initial one-particle density matrix is constructed from the averaged alpha and beta atomic densities.

`ao` Atomic density initial guess. The initial one-particle density matrix is constructed from diagonal atomic densities derived from the occupation of the atoms. It is efficient for Dunning's basis sets.

`core` Core Hamiltonian initial guess. The initial MOs are obtained by diagonalizing the one-electron integral matrix.

`mo` The SCF calculation will use the MO coefficients obtained in a previous calculation and stored in the `MOCOEF` file. The calculation can only be restarted from the MOs computed with the same basis set.

`restart` The SCF calculation will use the density matrices obtained in a previous calculation and stored in the `SCFDENSITIES` file. If the calculation is restarted from the densities obtained with another basis set, the `VARS` file is also required. See also note 1 below.

`off` No SCF calculation will be performed, but the Fock-matrix and the MO coefficients obtained in a previous calculation will be used in the correlation calculations. This requires the `FOCK`, `MOCOEF`, and `VARS` files from the previous calculation. See also note 1 below.

`min` A density fitting SCF calculation will be performed using the cc-pVTZ-min minimal basis set (see the description of keyword `basis`), and the resulting density will be used as initial guess. In the minimal-basis SCF calculation the AO basis set is used as the auxiliary basis, and loose convergence thresholds are employed, consequently, the energy is unreliable and should not be used for any purpose.

`small` A density fitting SCF calculation will be performed using a smaller basis set which must be specified by keyword `basis_sm` (see the description of keyword `basis_sm`), and the resulting density will be used as initial guess.

`gfn1` A GFN1-xTB calculation will be performed if the XTB program is available, and the resulting density is used as an initial guess for the valence electrons. The `sad` technology is utilized to get an initial guess for the core electrons.

`gfn2` It is the same as `gfn1`, except that a GFN2-xTB is used by the XTB program.

Notes:

1. If your SCF calculation is killed for some reason, e.g., by a power failure, but if you have all the files created by the program, you can simply continue the SCF calculation. To that end you should just execute program `scf` in the directory where MRCC was running. The SCF program will continue from the iteration step where it crashed. If you also want to run a post-SCF calculation, just restart the entire run with `scfiguess=off` after the SCF has converged.
2. Restarting from densities obtained with a bigger basis set is not allowed.
3. To restart SCF runs from the results of DFT embedding calculations with the Huzinaga-equation- or projector-based approaches use MO coefficients, i.e., `scfiguess=mo`, since only the subsystem densities are stored at the end of the embedding calculation but the MOs are available for the entire system.

Default: `scfiguess=sad`

Examples:

1. For a core Hamiltonian initial guess set `scfiguess=core`
2. For restarting the SCF calculation from the results of a calculation performed with the same basis set type `scfiguess=restart`. Note that you need the `SCFDENSITIES` file from the previous run.
3. You would like to generate a good initial guess for an aug-cc-pVTZ SCF calculation. First, run a calculation with the cc-pVTZ basis set (cc-pVTZ-min is also a good option), that is, your input file should contain the
   ```
   basis=cc-pVTZ
   ```
   line. Then, restart your aug-cc-pVTZ calculation from the cc-pVTZ density matrix. To that end the `MINP` file should include the following lines:
   ```
   basis=aug-cc-pVTZ
   scfiguess=restart
   ```
   Note that the `SCFDENSITIES` and the `VARS` files from the cc-pVTZ run must be copied to the directory where the aug-cc-pVTZ calculation is executed.
4. The calculations in the previous example can be run more simply, in one step using the `small` option and the `basis_sm` keyword as
   ```
   basis=aug-cc-pVTZ
   basis_sm=cc-pVTZ
   scfiguess=small
   ```

`scfloc` Specifies the localization type used for local density fitting SCF calculations. For more information on local SCF calculations, see the `scfalg` keyword.

Options:

`boys` Boys–Foster localization.

`pm` Pipek–Mezey localization with Löwdin charges.

`pmm` Pipek–Mezey localization with Mulliken charges.

`gboys`<*m*> generalized Boys localization by Jansík *et al.* [321], where $m$ is a positive integer defining the power of orbital variance. Note that for $m = 1$, the Boys localization is retrieved.

Default: `scfloc=pm`

Example: to use Boys localization in the SCF cycles, set `scfloc=boys`

`scflshift` Level shift parameter for the SCF calculation.

Options:

`off` No level shifting.

<*any real positive number* > The value of the level shift parameter in a.u.

`on` Equivalent to `scflshift=0.2`

Default: `scflshift=0.2`

Notes:

1. `scflshift` is turned off in the first diagonalization by default.
2. `scflshift_gaptol` and `scflshift_end` have equal priority in shutting down level shifting. This means that the HOMO-LUMO gap has to be smaller than `scflshift_gaptol` and the iteration number has to be lower than `scflshift_end` to perform level shifting. However, if the HOMO-LUMO gap is lower than `scflshift_gaptol` and the iteration number is lower than `scflshift_end`, the level shifting is still switched off if the RMS of difference density is lower than `scflshift_dtol`.

Example: To use a level shift value of 0.5 a.u. give `scflshift=0.5`

`scflshift_dtol` Specifies the tolerance in the RMS of the difference density matrix where the SCF level shifting is turned off.

Options: <*any positive real number*> or `0`

Default: `scflshift_dtol=0`, that is, the SCF level shifting is independent of the RMS of the difference density matrix.

Note: the `scflshift_dtol` criterion of level shifting is only checked if the HOMO-LUMO gap is smaller than `scflshift_gaptol` and the number of SCF iterations is lower than `scflshift_end`.

Example: to use a level shifting scheme that depends on the RMS of the difference density matrix, set `scflshift_dtol=1d-4`, that is, the level shifting is turned off after the RMS of difference density is lower than 0.0001 (regardless of the iteration step and the HOMO-LUMO gap).

`scflshift_end` Specifies the last iteration step in which the SCF level shifting is applied.

Options: <*any positive integer*>

Default: `scflshift_end=8`, that is, the SCF level shifting is turned off after iteration step 8.

Notes: `scflshift_gaptol` and `scflshift_end` have equal priority in shutting down level shifting. See also `scflshift_dtol` and the notes of `scflshift` for more details.

Example: to turn off the SCF level shifting after iteration step 20, give `scflshift_end=20`

`scflshift_gaptol` Specifies the tolerance (in a.u.) in the gap between the HOMO and the LUMO where the SCF level shifting is turned off.

Options: <*any positive real number*> or `0`

Default: `scflshift_gaptol=0.2`, that is, the SCF level shifting is turned off after the HOMO-LUMO gap is larger than the given value.

Notes:

1. `scflshift_gaptol` and `scflshift_end` have equal priority in shutting down level shifting. See also `scflshift_dtol` and the notes for `scflshift` for more details.
2. `scflshift_gaptol=0` means that the level shifting scheme is independent of the HOMO-LUMO gap.
3. `scflshift_gaptol` allows dynamic level shifting, i.e., when the HOMO-LUMO gap is larger than `scflshift_gaptol`, the level shifting will be switched off, while it is re-activated if the gap becomes lower than `scflshift_gaptol` (note that the re-activation can be inhibited with `scflshift_dtol` and `scflshift_end`).

Example: to increase the gap criteria to turn the level shifting on type `scfdamp_gaptol=0.3`

`scfmaxit` Maximum number of iteration steps in SCF calculations.

Options: <*any positive integer*>

Default: `scfmaxit=50`

Example: to increase the maximum number of SCF iterations to 200 give `scfmaxit=200`

`scftol` Convergence threshold for the energy in SCF calculations. The energy will be accurate to $10^{-\texttt{scftol}}$ $\mathrm{E}_h$.

Options: <*any integer*>

Default: `scftol`=max(8,`cctol`) for property calculations,
`scftol`=max(6,`cctol`) otherwise

Example: for an accuracy of $10^{-8}$ $\mathrm{E}_h$ one must give `scftol=8`

`scftype` Specifies the type of the Hartree–Fock/Kohn–Sham/MC SCF procedure, or the type of the molecular orbitals if the MO integrals are computed by other programs. See also the description of keyword `rohftype`.

Options: `RHF`, `ROHF`, `UHF`, `RKS`, `ROKS`, `UKS`, or `MCSCF`

Notes:

1. It is very important to give this keyword if MRCC is used together with another code and ROHF or MCSCF orbitals are used since this keyword tells MRCC that the orbitals are not canonical HF orbitals. Please also set keyword `rohftype` in this case.
2. If an SCF calculation is run, the type of the SCF wave function can also be controlled by keyword `calc`. See the description of `calc`.
3. The `RKS`, `UKS`, and `ROKS` options are synonyms for the `RHF`, `UHF`, and `ROHF` options, respectively. That is, for DFT calculations, the `RHF`, `UHF`, and `ROHF` options will instruct the code to run RKS, UKS, and ROKS calculations, respectively, and vice versa.

Default: `scftype=RHF` for closed-shell systems, `scftype=UHF` for open shells.

Example: to use ROHF or ROKS for open-shell systems type `scftype=ROHF`

`scspe` Scaling factor for the higher-order terms in the dRPA equations for the sedRPA and seSOSEX methods of Ref. 44.

Options:

<*any real number*> the higher-order terms in the dRPA equations will be scaled by this number.

Default: `scspe=0.8` for the sedRPA and seSOSEX methods, `scspe=1.0` otherwise.

Example: to set a scaling factor of 0.9 type `scspe=0.9`

`scsph` Scaling factor for the higher-order terms in the energy expressions for the dsdRPA and dsSOSEX methods of Ref. 44.

Options:

<*any real number*> the higher-order terms in the energy expressions will be scaled by this number.

Default: `scsph=0.85` for the dsdRPA and dsSOSEX methods, `scsph=1.0` otherwise.

Example: to set a scaling factor of 0.9 type `scsph=0.9`

`scsps` Scaling factor for the antiparallel-spin component of the correlation energy in spin-scaled correlation calculations [SCS-MP2, SCS-CC2, SCS-CIS(D), SCS-CIS($D_\infty$), SCS-ADC(2), SOS-MP2, SOS-CC2, SOS-CIS(D), SOS-CIS($D_\infty$), SOS-ADC(2)] [109, 120].

Options:

<*any real number*> the antiparallel-spin component of the correlation energy will be scaled by this number.

Notes:

1. In the case of ground-state double hybrid DFT calculations, `scsps` is overwritten by the corresponding parameter of the DFT functional. To change this parameter, use `dft=user` or `userd` and set the value of `MP2s`. If an excited-state double hybrid DFT calculation is executed, the scaling factor for the antiparallel-spin component is calculated as `MP2s*scsps` when solving the excited-state equations.

2. For SCS-CIS(D), in the case of the parameterization of Rhee and Head-Gordon [151], this scaling factor is used for the direct terms.

Default: `scsps=6/5` for the SCS methods, `scsps=1.3` for the SOS approaches

Example: to set a scaling factor of 1.5 type `scsps=1.5`

`scsps_t` Scaling factor for the indirect terms with antiparallel-spin in SCS-CIS(D) calculations (see Ref. 151).

Options:

*<any real number>* the indirect terms with antiparallel-spin will be scaled by this number.

Note: In the case of the SCS-CC2 consistent analogue of the SCS-CIS(D) method [119], the direct and indirect terms are scaled by the same factor.

Default: `scsps_t=scsps`

Example: to set a scaling factor of 1.5 type `scsps_t=1.5`

`scspt` Scaling factor for the parallel-spin component of the correlation energy in spin-scaled correlation calculations [SCS-MP2, SCS-CC2, SCS-CIS(D), SCS-CIS($D_\infty$), SCS-ADC(2), SOS-MP2, SOS-CC2, SOS-CIS(D), SOS-CIS($D_\infty$), SOS-ADC(2)] [109, 120].

Options:

*<any real number>* the parallel-spin component of the correlation energy will be scaled by this number.

Notes:

1. In the case of ground-state double hybrid DFT calculations, `scspt` is overwritten by the corresponding parameter of the DFT functional. To change this parameter, use `dft=user` or `userd` and set the value of `MP2t`. If an excited-state double hybrid DFT calculation is executed, the scaling factor for the parallel-spin component is calculated as `MP2t*scspt` when solving the excited-state equations.
2. For SCS-CIS(D) and SOS-CIS(D), in the case of the parameterization of Rhee and Head-Gordon [151], this scaling factor is used for the direct terms.

Default: `scspt=1/3` for the SCS methods, `scspt=0.0` for the SOS approaches

Example: to set a scaling factor of 0.5 type `scspt=0.5`

`scspt_t` Scaling factor for the indirect terms with parallel-spin in SOS-CIS(D) or SCS-CIS(D) calculations (see Ref. 151).

Options:

*<any real number>* the indirect terms with parallel-spin will be scaled by this number.

Note: In the case of the SCS-CC2 and SOS-CC2 consistent analogues of the SCS-CIS(D) and SOS-CIS(D) methods [119], the direct and indirect terms are scaled by the same factor.

Default: `scspt_t=scspt`

Example: to set a scaling factor of 0.5, type `scspt_t=0.5`

`spairtol` Threshold for the selection of strong pairs in local MP2, dRPA, and CC methods. For each orbital pair an estimate of the pair correlation energy is calculated (see the description of keyword `wpairtol`). An orbital pair will be considered as strong pair if the absolute value of the pair correlation energy estimate is greater than `spairtol`. In the subsequent calculations strong pairs will be treated at a higher level, while for the other pairs (weak and distant) the corresponding pair correlation energy estimates will be added to the correlation energy. See also Refs. 30 and 38 for more details.

Options:

`off` the local MP2 or dRPA pair correlation energy estimate is not calculated, an orbital pair will be considered as strong pair in this case if the absolute value of the available pair correlation energy estimate is greater than `wpairtol`. See also the description of keyword `wpairtol` and Ref. 31.

<*any positive real number*> Orbital pairs with pair correlation energy estimates greater than this number (in $E_h$) will be considered as strong pairs.

Default: `spairtol=1e-4` for `localcc=2015`, `spairtol=off` for `localcc`$\geq$`2016`

Example: to set a threshold of $10^{-5}$ $E_h$ type `spairtol=1e-5`

`sqmprog` Specifies the semi-empirical quantum chemistry program to be used for ONIOM calculations. See the description below for the supported methods and programs. Currently, it is not necessary to specify the keyword as MRCC automatically checks the program availability for the specified semi-empirical method, and the available techniques are implemented in only one of the supported programs.

Options:

`off` Semi-empirical quantum chemistry method is not requested.

`mopac2016` The MOPAC package of Stewart [91].

`xtb` The XTB package of Grimme [86].

Default: the default for the AM1, MNDO, MNDOD, PM3, PM6, PM6-D3, PM6-DH+, PM6-DH2, PM6-DH2X, PM6-D3H4, PM6-D3H4X, PM7, PM7TS, and RM1 methods is `mopac2016`. The default for GFN0-xTB, GFN1-xTB, and GFN2-xTB is `xtb`. Otherwise `sqmprog=off`.

Example: to use the MOPAC program for a PM5 calculation, type `sqmprog=mopac2016`.

`subminp` This keyword controls the input handling of ONIOM calculations.

Options:

`top` Keywords specified in the MINP file are only used in the last (high level) calculation. For other layers, the default options will be set.

`minp` Keywords specified in the MINP file are used in all calculations. The following keywords are not affected by this option as these are handled by the ONIOM algorithm automatically: `mem`, `qmmm`, `gopt`, `freq`, `calc`, `basis`, `charge`, `mult`, `verbosity`, `test`, `qmreg`, `oniom`, `geom`, `subminp`, `dens`, `mpitasks`, `pcm`, `oniom_pcm`, `pcm_*`, `oniom_eechg`, `embed`, `corembed`, `orbloce`, `orblocv`.

`temp` The use of template files is requested. Before each calculation, a `MINP.<` $X$ `>.tpl` file is searched, where $X$ is the serial number of the calculation. If any template file is found, its content is used as is, without consistency check, thus the template file should contain keywords excluding those that are listed above for the `minp` option. Note that the template files are only searched in the directory where `dmrcc` is executed.

`t+t` The `top` and `temp` options are used together.

`m+t` The `minp` and `temp` options are used together.

Notes

1. The first line of the `MINP.<` $X$ `>.tpl` file is catanated to the input keyword list of `MINP.<` $X$ `>` if the MOPAC program is used for the calculation.
2. The first line of the `MINP.<` $X$ `>.tpl` file is catanated to the command line if the xTB program is used for the calculation. The rest of `MINP.<` $X$ `>.tpl` is added to `MINP.<` $X$ `>.xtbinp`, which is the xcontrol file of the `xTB` program.

Default: `top`

Examples:

1. The following example is a CCSD:PBE (aug-cc-pVDZ:STO-3G) ONIOM calculation. The `MINP` file contains the following keywords:
```
basis=aug-cc-pVDZ
calc=CCSD
scfdtol=10
subminp=top
oniom=2
8-9
PBE / STO-3G / 0 / 1
0
```
   In this case, PBE/STO-3G (full system), PBE/STO-3G (model system), and CCSD/aug-cc-pVDZ (model system) calculations are requested, which are denoted as calculation #1, #2, and #3, respectively. Calculation #1 and #2 are executed with the default `scfdtol`, while calculation #3 will use `scfdtol=10`.
2. The setting are the same as in example 1 except that `subminp=minp`. In this case, all calculations will use `scfdtol=10`.
3. The settings are the same as in example 1 except that `subminp=temp`, and there are two files (`MINP.2.tpl` and `MINP.3.tpl`) in which there is only the following line:
```
scflshift=0.3
```
   In this case, calculation #1 will use the default options, while in calculations #2 and #3, `scflshift=0.3` will be set. Note that all calculations will use the default option for `scfdtol`.

`subsys_bopu` The keyword controls the execution of the Boughton–Pulay (BP) algorithm for subsystem-related methods. Note that the BP technique is required for the automatic border handling in the `oniom` schemes and for the `bopu` orbital selection technique of the embedding methods (see keywords `embed` and `corembed`). Otherwise, it functions only as an analytical tool when the system is divided into environment and active parts and can be omitted for efficiency considerations. See also keywords `bp_subsyso` and `bp_subsysv`.

Options: `on` or `off`

Default: `on`

Example: To omit the BP algorithm, type `subsys_bopu=off`.

`symm` Spatial symmetry (irreducible representation) of the state. See Sect. 13 for the implemented point groups, conventions for irreps, etc.

Options:

`0` No symmetry adaptation, that is, all calculations will use the $C_1$ point group

`off` Equivalent to `symm=0`

`1, 2, ..., 8` Serial number of the irrep (see Sect. 13).

<*irrep label*> Label for the irrep (see Sect. 13).

Note: Irreps can only be specified by their serial numbers if MRCC is used with another program. In that case please check the manual or output of the other program system for the numbering of irreps.

Default: by default the state symmetry is determined on the basis of the occupation of the HF determinant.

Examples:

1. for the second irrep of the point group type `symm=2`
2. for the $\mathrm{B}_{1u}$ irrep of the $D_{2h}$ point group type `symm=B1u`

`talg` Specifies the algorithm for the calculation of the (T) correction in the case of the CCSD(T) method.

Options:

`occ` The outmost loops run over the occupied indices of the triples amplitudes.

`virt` The outmost loops run over the virtual indices of the triples amplitudes.

`lapl` Laplace transformed (T) energy expression for `localcc`≥`2016` according to Ref. 36. See also the `laptol` keyword to set the accuracy of the numerical Laplace transform.

`topr` The T0′ semi-canonical approximation of the local (T) expression according to Ref. 36.

Default: `talg=occ` for conventional CCSD(T) calculations, `talg=lapl` for the local CCSD(T) scheme of `localcc`≥`2016` and `talg=virt` for local CCSD(T) with `localcc=2013` or `2015`.

Notes:

1. For algorithmic reasons in the case of previous local CCSD(T) schemes (`localcc=2013` or `2015`) `talg=virt` is the only option. For `localcc`≥`2016`, the default is `talg=lapl`, and `talg=virt` is used if `lcorthr=0` is set.

2. For conventional CCSD(T) calculations `talg=occ` is recommended since the algorithm is somewhat faster than the other one. In turn, if density-fitting is not employed, its memory requirement is higher. The program checks automatically if the available memory is sufficient for the first algorithm (i.e., `talg=occ`). If this is not the case, `talg` will be automatically set to `virt`.
3. If density-fitting is employed and `ccsdalg=dfdirect` is set, a highly memory-economic, completely in-core, integral-direct implementation of the `talg=occ` scheme is invoked [46]. This algorithm assembles the necessary four-center integrals with two and three virtual MO indices on the fly instead of storing the entire integral lists on disk or in memory. The on the fly integral assembly has negligible additional cost compared to the cost of the (T) correlation energy and has better OpenMP parallel scaling due to the elimination of all disk I/O.
4. In the case of `localcc`$\geq$`2016` the `talg=topr` algorithm is approximately three times faster for the (T) contribution than `talg=lapl` with its default settings but considerably less accurate. For quick exploratory calculations the `talg=lapl` algorithm is recommended in combination with `laptol=0.1`.

Example: to change the default for a conventional CCSD(T) calculation set `talg=virt`

`temp` The temperature in K at which the thermodynamic properties are evaluated (see also keyword `freq`).

Options: <*any positive real number*>

Default: `temp=298.15`

Example: for 300 K set `temp=300.0`

`test` A keyword for testing Mrcc. If an energy value is specified using this keyword, it will be compared to the energy calculated last time [e.g., the CCSD(T) energy and not the CCSD or HF energy if `calc=CCSD(T)`] in the Mrcc run. An error message will be displayed and the program exits with an error code if the test energy and the calculated energy differ. This keyword is mainly used by the developers of the program to create test jobs to check the correctness of the computed energies. (See Sect. 8 for the further details.)

Options:

`off` No testing.

<*any real number*> The energy to be tested.

Default: `test=off`

Example: to set a test energy of -40.38235315 E$_h$ type `test=-40.38235315`

`tlmo` Threshold for local MO (LMO) completeness for local excited-state calculations. See Ref. 43 for a detailed description of this threshold ($T_{\rm LMO}$ in the paper).

Options:

<*any real number in the [0,1] interval*> This number will be used as the completeness criterion.

Default: `tlmo=0.999`

Example: to set a threshold of 0.99 type `tlmo=0.99`

`tpao` Threshold for projected AO (PAO) completeness for local excited-state calculations. See Ref. 43 for a detailed description of this threshold ($T_{\mathrm{PAO}}$ in the paper).

Options:

*<any real number in the [0,1] interval>* This number will be used as the completeness criterion.

Default: `tpao=0.94`

Example: to set a threshold of 0.9 type `tpao=0.9`

`tprint` Controls the printing of converged cluster amplitudes/CI coefficients if `ccprog=mrcc` or `ccprog=ccsd`.

Options:

`off` No printing.

*<any real number>* Cluster amplitudes/CI coefficients whose absolute value is greater than this number will be printed.

Note: The value of the cluster amplitude/CI coefficient and the corresponding spin-orbital labels (serial number of the orbital + `a` or `b` for alpha or beta spin orbitals, respectively) will be printed. The numbering of the orbitals corresponds to increasing orbital energy order. Note that orbital energies are printed at the end of the SCF run if `verbosity`$\geq$`3`. You can also identify the orbitals using MOLDEN (see Sect. 14.1).

Default: `tprint=off`

Example: to set a threshold of 0.01 give `tprint=0.01`

`uncontract` Uncontract contracted basis sets.

Options: `on` or `off`

Default: `uncontract=off`

Example: to uncontract the basis set add `uncontract=on`

`unit` Specifies the units used for molecular geometries.

Options:

`angs` Ångströms will be used

`bohr` Atomic units will be used

Default: `unit=angs`

Example: to use bohrs rather than ångströms the user should set `unit=bohr`

`usedisk` Controls the memory and disk usage of the domain construction part of the LNO-CC algorithms.

Options: `0`, `1`, `2`. The minimal memory requirement of the domain construction part of the LNO-CC methods decreases, while the size of the arrays stored temporarily on disk increases with increasing value of the option. `usedisk=0` prohibits the use of disk I/O in this part and hence requires somewhat more memory. For more details see Ref. 42.

Default: `usedisk=2`

Note: `usedisk=0` is only compatible with the in-core DF-CCSD algorithm invoked by `ccsdalg=dfdirect`.

Example: to invoke the in-core domain construction algorithm, for instance, if a local hard disk is not available for the executing node, set `usedisk=0`.

`verbosity` Controls the verbosity of the output.

Options: `0`, `1`, `2`, `3`. The verbosity of the output increases gradually with increasing value of the option. Error messages are not suppressed at any level.

Default: `verbosity=2`

Example: to increase the amount of information printed out, give `verbosity=3`

`wpairtol` Threshold for the selection of weak pairs in local MP2, RPA, and CC methods. For each orbital pair the estimate of the pair correlation energy is calculated with a multipole approximation [31, 312, 326]. An orbital pair will be considered as distant pair if the absolute value of the multipole-based pair correlation energy estimate is smaller than `wpairtol`. For the distant pairs the corresponding multipole-based pair correlation energy estimates will be added to the correlation energy, and distant pairs will be neglected in the subsequent calculations.

In the case of `localcc=2015`, for the remaining pairs a more accurate pair correlation energy estimate will be calculated using orbital specific virtuals (OSVs) controlled by keyword `osveps`, and these pairs will be further classified as weak and strong pairs controlled by keyword `spairtol`, see the description of keyword `spairtol`. The extended domain of an occupied orbital will include those orbitals for which the latter accurate pair correlation energy estimate is greater than `spairtol`. See also Ref. 30 for more details.

In the case of `localcc`$\geq$`2016`, `spairtol=off` is set as default, and the extended domain of an occupied orbital will include those orbitals for which the multipole-based pair correlation energy is greater than `wpairtol`. See also Refs. 31 and 38 for more details.

Note that for closed-shell local CC methods if `spairtol`$\neq$`off` is specified as a non-default option in the case of `localcc`$\geq$`2016`, accurate MP2 pair energies are computed in the extended domains for the remaining non-distant pairs. Then the non-distant pairs are further divided into weak and strong categories according to the value of `spairtol`, as discussed above. In this case the MP2 pair energies of the weak pairs are added to the correlation energy, and new, somewhat smaller extended domains are constructed to proceed with the higher-level computation as above using solely the strong pair list. See also Ref. 38 for more details.

Options:

<*any positive real number*> Orbital pairs with multipole-based pair correlation energy estimates smaller than this number (in $\mathrm{E}_h$) will be considered as distant pairs.

Default: `wpairtol=1e-5` for local MP2 and CC if `localcc`≥`2018`, `wpairtol=`min(`1e-6`, `0.01*spairtol`) for `localcc=2015`

Note: For defaults with other than the above settings, see the description of `lcorthr`

Example: to set a threshold of $5 \cdot 10^{-6}$ $\mathrm{E}_h$ type `wpairtol=5e-6`

# 13 Symmetry

The Mrcc program can handle Abelian point group symmetry. The handling of symmetry can be controlled by keywords `cmpgrp` (see page 60) and `symm` (see page 163). In the following we give the character tables used by the program. The symmetry of electronic states can be specified by keyword `symm` using either the serial number of the irrep or its symbol. The serial number of an irrep is given by its position in the below tables as appropriate. To specify the state symmetry by the symbol of the irrep replace the superscripts in the irrep symbol by lowercase letters, e.g., give `B2g` for $\mathrm{B}_{2g}$. For the $\mathrm{A}'$ and $\mathrm{A}''$ irreps of $C_s$ group use `A’` and `A"`, respectively (apostrophe and quotation mark).

Character table for the $C_1$ point group

| | $E$ | |
|---|---|---|
| A | 1 | $x$, $y$, $z$, $R_x$, $R_y$, $R_z$,<br>$x^2$, $y^2$, $z^2$, $xy$, $xz$, $yz$ |

Character table for the $C_i$ point group

| | $E$ | $i$ | |
|---|---|---|---|
| $\mathrm{A}_{1g}$ | 1 | 1 | $R_x$, $R_y$, $R_z$, $x^2$, $y^2$, $z^2$, $xy$, $xz$, $yz$ |
| $\mathrm{A}_{1u}$ | 1 | $-1$ | $x$, $y$, $z$ |

Character table for the $C_s$ point group

| | $E$ | $\sigma_h$ | |
|---|---|---|---|
| $\mathrm{A}'$ | 1 | 1 | $x$, $y$, $R_z$, $x^2$, $y^2$, $z^2$, $xy$ |
| $\mathrm{A}''$ | 1 | $-1$ | $z$, $R_x$, $R_y$, $yz$, $xz$ |

Character table for the $C_2$ point group

| | $E$ | $C_2$ | |
|---|---|---|---|
| A | 1 | 1 | $z$, $R_z$, $x^2$, $y^2$, $z^2$, $xy$ |
| B | 1 | $-1$ | $x$, $y$, $R_x$, $R_y$, $yz$, $xz$ |

Character table for the $C_{2v}$ point group

| | $E$ | $C_2$ | $\sigma_h$ | $\sigma_v$ | |
|---|---|---|---|---|---|
| $\mathrm{A}_1$ | 1 | 1 | 1 | 1 | $z$, $x^2$, $y^2$, $z^2$ |
| $\mathrm{B}_1$ | 1 | $-1$ | 1 | $-1$ | $y$, $R_x$, $yz$ |
| $\mathrm{B}_2$ | 1 | $-1$ | $-1$ | 1 | $x$, $R_y$, $xz$ |
| $\mathrm{A}_2$ | 1 | 1 | $-1$ | $-1$ | $R_z$, $xy$ |

Character table for the $C_{2h}$ point group

| | $E$ | $C_2(z)$ | $i$ | $\sigma_h$ | |
|---|---|---|---|---|---|
| $\mathrm{A}_g$ | 1 | 1 | 1 | 1 | $R_z$, $x^2$, $y^2$, $z^2$, $xy$ |
| $\mathrm{B}_g$ | 1 | −1 | 1 | −1 | $R_x$, $R_y$, $xz$, $yz$ |
| $\mathrm{A}_u$ | 1 | 1 | −1 | −1 | $z$ |
| $\mathrm{B}_u$ | 1 | −1 | −1 | 1 | $x$, $y$ |

Character table for the $D_2$ point group

| | $E$ | $C_2(z)$ | $C_2(y)$ | $C_2(x)$ | |
|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 1 | $x^2$, $y^2$, $z^2$ |
| $\mathrm{B}_1$ | 1 | 1 | −1 | −1 | $z$, $R_z$, $xy$ |
| $\mathrm{B}_2$ | 1 | −1 | 1 | −1 | $y$, $R_y$, $xz$ |
| $\mathrm{B}_3$ | 1 | −1 | −1 | 1 | $x$, $R_x$, $yz$ |

Character table for the $D_{2h}$ point group

| | $E$ | $C_2(z)$ | $C_2(y)$ | $C_2(x)$ | $i$ | $\sigma_{xy}$ | $\sigma_{xz}$ | $\sigma_{yz}$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\mathrm{A}_g$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | $x^2$, $y^2$, $z^2$ |
| $\mathrm{B}_{1g}$ | 1 | 1 | −1 | −1 | 1 | 1 | −1 | −1 | $R_z$, $xy$ |
| $\mathrm{B}_{2g}$ | 1 | −1 | 1 | −1 | 1 | −1 | 1 | −1 | $R_y$, $xz$ |
| $\mathrm{B}_{3g}$ | 1 | −1 | −1 | 1 | 1 | −1 | −1 | 1 | $R_x$, $yz$ |
| $\mathrm{A}_u$ | 1 | 1 | 1 | 1 | −1 | −1 | −1 | −1 | |
| $\mathrm{B}_{1u}$ | 1 | 1 | −1 | −1 | −1 | −1 | 1 | 1 | $z$ |
| $\mathrm{B}_{2u}$ | 1 | −1 | 1 | −1 | −1 | 1 | −1 | 1 | $y$ |
| $\mathrm{B}_{3u}$ | 1 | −1 | −1 | 1 | −1 | 1 | 1 | −1 | $x$ |

# 14 Interface to molecular visualization software

## 14.1 Molden

For the visualization of molecular structures, molecular orbitals, electron densities, geometry optimization steps, normal modes, and IR spectra an interface has been developed to the Molden program [327]. Cartesian coordinates, basis function information, MO coefficients, etc. are saved to file `MOLDEN` using Molden format. After the termination of Mrcc, Molden should be started by typing `molden MOLDEN`. The Molden interface can be controlled by the `molden` keyword (see page 115 for the description of the keyword).

If the MOs are localized, the `MOLDEN` file containing the canonical HF orbitals is saved under the name `MOLDEN.CAN`, and the canonical MOs are replaced by the localized ones in the `MOLDEN` file. You can use both files to visualize the MOs that you are interested in.

Please note that Molden can also be used for the generation of input molecular structures in Z-matrix or xyz format (see the description of the `geom` keyword on page 98).

## 14.2 xyz-file

Cartesian coordinates are also written to file `COORD.xyz` in xyz (XMol) format, which can be processed by many molecular visualization programs. This interface can also be controlled by the `molden` keyword (see page 115 for the description of the keyword).

# 15 Acknowledgments

The authors of the Mrcc program are very grateful to

- Professor Jürgen Gauss (Universität Mainz) for continuous support and discussions, development and maintenance of the Cfour interface, and for the permission to use the basis set format of Cfour.
- Dr. Michael Harding (TU Karlsruhe) for continuous support and for his help in the parallelization of the code.
- Professor Hans-Joachim Werner (Universität Stuttgart), Professor Peter J. Knowles (Cardiff University), and Dr. Andy May (Cardiff University) for the development and maintenance of the Molpro interface, and many useful suggestions.
- Professor Péter G. Szalay (Eötvös University, Budapest) for his continuous support and for the development of the Columbus interface.
- Professor Lucas Visscher (VU University Amsterdam) for the development and maintenance of the Dirac interface, and for the permission to distribute the interface code.
- Dr. Andreas Götz (University of California San Diego) for the development and maintenance of the Amber interface,
- Professor Péter R. Surján (Eötvös University, Budapest) for his support at the early stages of the code development.
- János Brátán (TU Budapest) for continuous technical support.

Financial support to the development of the Mrcc suite has been provided by the

- Hungarian Scientific Research Fund (OTKA), grant agreement Nos. T023052, T035094, T047182, T49718, D048583, NF72194, K108752, and PD108451.
- National Research, Development, and Innovation Office (NKFIH), grant agreement Nos. KKP126451, FK142489, and PD142372.
- European Research Council (ERC) under the European Community's Seventh Framework Programme (FP7/2007-2013), ERC grant agreement No. 200639; and the 2021-2027 Horizon Europe Framework Programme, ERC grant agreement No. 101076972.
- Lendület Program of the Hungarian Academy of Sciences, grant agreement Nos. LEN2013-13 and LP2022-9/2022.
- Eötvös Loránd Research Network and Hungarian Research Network
- Hungarian Science and Technology Foundation, grant agreement Nos. IND-5/2001 and IND 04/2006.
- Bolyai Research Scholarship of the Hungarian Academy of Sciences, grant agreement Nos. BO/00593/07, BO/00199/14, BO/00558/18, and BO/00529/23/7.

- New Hungary Development Plan, project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002.
- Higher Education Excellence Program of the Ministry of Human Capacities.
- New National Excellence Program of the Ministry of Human Capacities, IDs: ÚNKP-17-4-BME-55, ÚNKP-18-4-BME-257, ÚNKP-20-4-II-BME-293, ÚNKP-21-4-II-BME-322, ÚNKP-22-4-II-BME-157, ÚNKP-23-4-II-BME-275, EKÖP-24-4-I-BME-266, EKÖP-24-4-II-BME-321, and EKÖP-24-4-II-BME-83.
- Hungarian HPC Infrastructure at NIIF Institute, Hungary.

# References

[1] Mihály Kállay, Péter R. Nagy, Dávid Mester, Zoltán Rolik, Gyula Samu, József Csontos, József Csóka, P. Bernát Szabó, László Gyevi-Nagy, Bence Hégely, István Ladjánszki, Lóránt Szegedy, Bence Ladóczki, Klára Petrov, Máté Farkas, Pál D. Mezei, and Ádám Ganyecz: The MRCC program system: Accurate quantum chemistry from water to proteins, *J. Chem. Phys.* **152**, 074107 (2020).

[2] Dávid Mester, Péter R. Nagy, József Csóka, László Gyevi-Nagy, P. Bernát Szabó, Réka A. Horváth, Klára Petrov, Bence Hégely, Bence Ladóczki, Gyula Samu, Balázs D. Lőrincz, and Mihály Kállay: Overview of developments in the MRCC program system, *J. Phys. Chem. A* **129**, 2086 (2025).

[3] Mihály Kállay and Péter R. Surján: Higher excitations in coupled-cluster theory, *J. Chem. Phys.* **115**, 2945 (2001).

[4] Mihály Kállay, Péter G. Szalay, and Péter R. Surján: A general state-selective coupled-cluster algorithm, *J. Chem. Phys.* **117**, 980 (2002).

[5] Mihály Kállay, Jürgen Gauss, and Péter G. Szalay: Analytic first derivatives for general coupled-cluster and configuration interaction models, *J. Chem. Phys.* **119**, 2991 (2003).

[6] Mihály Kállay and Jürgen Gauss: Analytic second derivatives for general coupled-cluster and configuration interaction models, *J. Chem. Phys.* **120**, 6841 (2004).

[7] Mihály Kállay and Jürgen Gauss: Calculation of excited-state properties using general coupled-cluster and configuration-interaction models, *J. Chem. Phys.* **121**, 9257 (2004).

[8] Yannick J. Bomble, John F. Stanton, Mihály Kállay, and Jürgen Gauss: Coupled cluster methods including non-iterative approximate quadruple excitation corrections, *J. Chem. Phys.* **123**, 054101 (2005).

[9] Mihály Kállay and Jürgen Gauss: Approximate treatment of higher excitations in coupled-cluster theory, *J. Chem. Phys.* **123**, 214105 (2005).

[10] Jürgen Gauss, Attila Tajti, Mihály Kállay, John F. Stanton, and Péter G. Szalay: Analytic calculation of the diagonal Born–Oppenheimer correction within configuration-interaction and coupled-cluster theory, *J. Chem. Phys.* **125**, 144111 (2006).

[11] Mihály Kállay and Jürgen Gauss: Calculation of frequency-dependent polarizabilities using general coupled-cluster models, *J. Mol. Struct.:THEOCHEM* **768**, 71 (2006).

[12] Jürgen Gauss, Kenneth Ruud, and Mihály Kállay: Gauge-origin independent calculation of magnetizabilities and rotational $g$ tensors at the coupled-cluster level, *J. Chem. Phys.* **127**, 074101 (2007).

[13] Darragh P. O'Neill, Mihály Kállay, and Jürgen Gauss: Calculation of frequency-dependent hyperpolarizabilities using general coupled-cluster models, *J. Chem. Phys.* **127**, 134109 (2007).

[14] Darragh P. O'Neill, Mihály Kállay, and Jürgen Gauss: Analytic evaluation of Raman intensities in coupled-cluster theory, *Mol. Phys.* **105**, 2447 (2007).

[15] Mihály Kállay and Jürgen Gauss: Approximate treatment of higher excitations in coupled-cluster theory. II. Extension to general single-determinant reference functions and improved approaches for the canonical Hartree–Fock case, *J. Chem. Phys.* **129**, 144101 (2008).

[16] Jürgen Gauss, Mihály Kállay, and Frank Neese: Calculation of electronic g-tensors using coupled-cluster theory, *J. Phys. Chem. A* **113**, 11541 (2009).

[17] Sanghamitra Das, Debashis Mukherjee, and Mihály Kállay: Full implementation and benchmark studies of Mukherjee's state-specific multi-reference coupled-cluster ansatz, *J. Chem. Phys.* **132**, 074103 (2010).

[18] Huliyar S. Nataraj, Mihály Kállay, and Lucas Visscher: General implementation of the relativistic coupled-cluster method, *J. Chem. Phys.* **133**, 234109 (2010).

[19] Sanghamitra Das, Mihály Kállay, and Debashis Mukherjee: Inclusion of selected higher excitations involving active orbitals in the state-specific multi-reference coupled-cluster theory, *J. Chem. Phys.* **133**, 234110 (2010).

[20] Mihály Kállay, Huliyar S. Nataraj, Bijaya K. Sahoo, Bhanu P. Das, and Lucas Visscher: Relativistic general-order coupled-cluster method for high-precision calculations: Application to the $\mathrm{Al}^+$ atomic clock, *Phys. Rev. A* **83**, 030503(R) (2011).

[21] Zoltán Rolik and Mihály Kállay: Cost-reduction of high-order coupled-cluster methods via active-space and orbital transformation techniques, *J. Chem. Phys.* **134**, 124111 (2011).

[22] Zoltán Rolik and Mihály Kállay: A general-order local coupled-cluster method based on the cluster-in-molecule approach, *J. Chem. Phys.* **135**, 104111 (2011).

[23] Sanghamitra Das, Mihály Kállay, and Debashis Mukherjee: Superior performance of Mukherjee's state-specific multi-reference coupled-cluster theory at the singles and doubles truncation scheme with localized active orbitals, *Chem. Phys.* **392**, 83 (2012).

[24] Zoltán Rolik, Lóránt Szegedy, István Ladjánszki, Bence Ladóczki, and Mihály Kállay: An efficient linear-scaling CCSD(T) method based on local natural orbitals, *J. Chem. Phys.* **139**, 094105 (2013).

[25] Zoltán Rolik and Mihály Kállay: A quasiparticle-based multireference coupled-cluster method, *J. Chem. Phys.* **141**, 134112 (2014).

[26] Mihály Kállay: A systematic way for the cost reduction of density fitting methods, *J. Chem. Phys.* **141**, 244113 (2014).

[27] Dávid Mester, József Csontos, and Mihály Kállay: Unconventional bond functions for quantum chemical calculations, *Theor. Chem. Acc.* **134**, 74 (2015).

[28] Pál D. Mezei, Gábor I. Csonka, Adrienn Ruzsinszky, and Mihály Kállay: Construction and application of a new dual-hybrid random phase approximation, *J. Chem. Theory Comput.* **11**, 4615 (2015).

[29] Bence Hégely, Ferenc Bogár, György G. Ferenczy, and Mihály Kállay: A QM/MM program using frozen localized orbitals and the Huzinaga equation, *Theor. Chem. Acc.* **134**, 132 (2015).

[30] Mihály Kállay: Linear-scaling implementation of the direct random-phase approximation, *J. Chem. Phys.* **142**, 204105 (2015).

[31] Péter R. Nagy, Gyula Samu, and Mihály Kállay: An integral-direct linear-scaling second-order Møller–Plesset approach, *J. Chem. Theory Comput.* **12**, 4897 (2016).

[32] Pál D. Mezei, Gábor I. Csonka, Adrienn Ruzsinszky, and Mihály Kállay: Construction of a spin-component scaled dual-hybrid random phase approximation, *J. Chem. Theory Comput.* **13**, 796 (2017).

[33] Bence Hégely, Péter R. Nagy, György G. Ferenczy, and Mihály Kállay: Exact density functional and wave function embedding schemes based on orbital localization, *J. Chem. Phys.* **145**, 064107 (2016).

[34] Gyula Samu and Mihály Kállay: Efficient evaluation of three-center Coulomb integrals, *J. Chem. Phys.* **146**, 204101 (2017).

[35] Dávid Mester, Péter R. Nagy, and Mihály Kállay: Reduced-cost linear-response CC2 method based on natural orbitals and natural auxiliary functions, *J. Chem. Phys.* **146**, 194102 (2017).

[36] Péter R. Nagy and Mihály Kállay: Optimization of the linear-scaling local natural orbital CCSD(T) method: Redundancy-free triples correction using Laplace transform, *J. Chem. Phys.* **146**, 214106 (2017).

[37] Dávid Mester, Péter R. Nagy, and Mihály Kállay: Reduced-cost second-order algebraic-diagrammatic construction method for excitation energies and transition moments, *J. Chem. Phys.* **148**, 094111 (2018).

[38] Péter R. Nagy, Gyula Samu, and Mihály Kállay: Optimization of the linear-scaling local natural orbital CCSD(T) method: Improved algorithm and benchmark applications, *J. Chem. Theory Comput.* **14**, 4193 (2018).

[39] Bence Hégely, Péter R. Nagy, and Mihály Kállay: Dual basis set approach for density functional and wave function embedding schemes, *J. Chem. Theory Comput.* **14**, 4600 (2018).

[40] Dávid Mester and Mihály Kállay: Reduced-scaling approach for configuration interaction singles and time-dependent density functional theory calculations using hybrid functionals, *J. Chem. Theory Comput.* **15**, 1690 (2019).

[41] Dávid Mester and Mihály Kállay: Combined density functional and algebraic-diagrammatic construction approach for accurate excitation energies and transition moments, *J. Chem. Theory Comput.* **15**, 4440 (2019).

[42] Péter R. Nagy and Mihály Kállay: Approaching the basis set limit of CCSD(T) energies for large molecules with local natural orbital coupled-cluster methods, *J. Chem. Theory Comput.* **15**, 5275 (2019).

[43] Dávid Mester, Péter R. Nagy, and Mihály Kállay: Reduced-scaling correlation methods for the excited states of large molecules: Implementation and benchmarks for the second-order algebraic-diagrammatic construction approach, *J. Chem. Theory Comput.* **15**, 6111 (2019).

[44] Pál D. Mezei, Adrienn Ruzsinszky, and Mihály Kállay: Reducing the many-electron self-interaction error in the second-order screened exchange method, *J. Chem. Theory Comput.* **15**, 6607 (2019).

[45] Pál D. Mezei and Mihály Kállay: Construction of a range-separated dual-hybrid direct random phase approximation, *J. Chem. Theory Comput.* **15**, 6678 (2019).

[46] László Gyevi-Nagy, Mihály Kállay, and Péter R. Nagy: Integral-direct and parallel implementation of the CCSD(T) method: Algorithmic developments and large-scale applications, *J. Chem. Theory Comput.* **16**, 366 (2020).

[47] József Csóka and Mihály Kállay: Speeding up density fitting Hartree–Fock calculations with multipole approximations, *Mol. Phys.* **118**, e1769213 (2020).

[48] László Gyevi-Nagy, Mihály Kállay, and Péter R. Nagy: Accurate reduced-cost CCSD(T) energies: Parallel implementation, benchmarks, and large-scale applications, *J. Chem. Theory Comput.* **17**, 860 (2021).

[49] Dávid Mester and Mihály Kállay: A simple range-separated double-hybrid density functional theory for excited states, *J. Chem. Theory Comput.* **17**, 927 (2021).

[50] József Csóka and Mihály Kállay: Speeding up Hartree–Fock and Kohn–Sham calculations with first-order corrections, *J. Chem. Phys.* **154**, 164114 (2021).

[51] P. Bernát Szabó, József Csóka, Mihály Kállay, and Péter R. Nagy: Linear scaling open-shell MP2 approach: algorithm, benchmarks, and large-scale applications, *J. Chem. Theory Comput.* **17**, 2886 (2021).

[52] Dávid Mester and Mihály Kállay: Spin-scaled range-separated double-hybrid density functional theory for excited states, *J. Chem. Theory Comput.* **17**, 4211 (2021).

[53] Mihály Kállay, Réka A. Horváth, László Gyevi-Nagy, and Péter R. Nagy: Size-consistent explicitly correlated triple excitation correction, *J. Chem. Phys.* **155**, 034107 (2021).

[54] Bence Hégely and Mihály Kállay: Multilevel approach to the initial guess for self-consistent field calculations, *Int. J. Quantum Chem.* **122**, e26782 (2021).

[55] Péter R. Nagy, László Gyevi-Nagy, and Mihály Kállay: Basis set truncation corrections for improved frozen natural orbital CCSD(T) energies, *Mol. Phys.* **119**, e1963495 (2021).

[56] Ádám Ganyecz and Mihály Kállay: Implementation and optimization of the embedded cluster reference interaction site model, *J. Phys. Chem. A* **126**, 2417 (2022).

[57] Dávid Mester and Mihály Kállay: Accurate spectral properties within double-hybrid density functional theory: A spin-scaled range-separated second-order algebraic-diagrammatic construction-based approach, *J. Chem. Theory Comput.* **18**, 865 (2022).

[58] Dávid Mester and Mihály Kállay: Charge-transfer excitations within density functional theory: How accurate are the most recommended approaches?, *J. Chem. Theory Comput.* **18**, 1646 (2022).

[59] Péter R. Nagy, László Gyevi-Nagy, Balázs D. Lőrincz, and Mihály Kállay: Pursuing the basis set limit of CCSD(T) non-covalent interaction energies for medium-sized complexes: case study on the S66 compilation, *Mol. Phys.* **121**, e2109526 (2023).

[60] Mihály Kállay, Réka A. Horváth, László Gyevi-Nagy, and Péter R. Nagy: Basis set limit CCSD(T) energies for extended molecules via a reduced-cost explicitly correlated approach, *J. Chem. Theory Comput.* **19**, 174 (2023).

[61] Bence Hégely, Ádám B. Szirmai, Dávid Mester, Attila Tajti, Péter G. Szalay, and Mihály Kállay: Performance of multilevel methods for excited states, *J. Phys. Chem. A* **126**, 6548 (2022).

[62] Dávid Mester and Mihály Kállay: Double-hybrid density functional theory for core excitations: Theory and benchmark calculations, *J. Chem. Theory Comput.* **19**, 1310 (2023).

[63] Dávid Mester and Mihály Kállay: Reduced-cost second-order algebraic-diagrammatic construction method for core excitations, *J. Chem. Theory Comput.* **19**, 2850 (2023).

[64] József Csóka and Mihály Kállay: Analytic gradients for local density fitting Hartree–Fock and Kohn–Sham methods, *J. Chem. Phys.* **158**, 024110 (2023).

[65] Dávid Mester and Mihály Kállay: Vertical ionization potentials and electron affinities at the double-hybrid density functional level, *J. Chem. Theory Comput.* **19**, 3982 (2023).

[66] P. Bernát Szabó, József Csóka, Mihály Kállay, and Péter R. Nagy: Linear-scaling local natural orbital CCSD(T) approach for open-shell systems: algorithm, benchmarks, and large-scale applications, *J. Chem. Theory Comput.* **19**, 8166 (2023).

[67] Dávid Mester and Mihály Kállay: Basis set limit of CCSD(T) energies: Explicit correlation versus density-based basis-set correction, *J. Chem. Theory Comput.* **19**, 8210 (2023).

[68] Réka A. Horváth and Mihály Kállay: Basis set limit MP2 energies for extended molecules via a reduced-cost explicitly correlated approach, *Mol. Phys.* **122**, e2304103 (2024).

[69] József Csóka, Bence Hégely, Péter R. Nagy, and Mihály Kállay: Development of analytic gradients for the Huzinaga quantum embedding method and its applications to large-scale hybrid and double hybrid DFT forces, *J. Chem. Phys.* **160**, 124113 (2024).

[70] Klára Petrov, József Csóka, and Mihály Kállay: Analytic gradients for density fitting MP2 using natural auxiliary functions, *J. Phys. Chem. A* **128**, 6566 (2024).

[71] Dávid Mester, Péter R. Nagy, and Mihály Kállay: Basis-set limit CCSD(T) energies for large molecules with local natural orbitals and reduced-scaling basis-set corrections, *J. Chem. Theory Comput.* **20**, 7453 (2024).

[72] Dávid Mester and Mihály Kállay: Higher-order coupled-cluster calculations with basis-set corrections, *Chem. Phys. Lett.* **861**, 141780 (2025).

[73] Bence Ladóczki, László Gyevi-Nagy, Péter R. Nagy, and Mihály Kállay: Enabling accurate and large-scale explicitly correlated CCSD(T) computations via a reduced-cost and parallel implementation, *J. Chem. Theory Comput.* **21**, 2432 (2025).

[74] Jan Řezáč: Cuby: An integrative framework for computational chemistry, *J. Comput. Chem.* **37**, 1230 (2016).

[75] Thomas Weymuth, Jan P. Unsleber, Paul L. Türtscher, Miguel Steiner, Jan-Grimo Sobez, Charlotte H. Müller, Maximilian Mörchen, Veronika Klasovita, Stephanie A. Grimmel, Marco Eckhoff, Katja-Sophia Csizi, Francesco Bosia, Moritz Bensberg, and Markus Reiher: SCINE-software for chemical interaction networks, *J. Chem. Phys.* **160**, 222501 (2024).

[76] P. Salvador, E. Ramos-Cordoba, M. Montilla, L. Pujal, and M. Gimferrer: APOST-3D: Chemical concepts from wavefunction analysis, *J. Chem. Phys.* **160**, 172502 (2024).

[77] https://ash.readthedocs.io/en/latest/index.html, Accessed Jan 1, 2025.

[78] https://cuby4.molecular.cz/, Accessed Jan 1, 2025.

[79] https://github.com/qcscine, Accessed Jan 1, 2025.

[80] https://github.com/mgimferrer/apost3d.git, Accessed Jan 1, 2025.

[81] A. Götz, M. A. Clack, and R. C. Walker: An extensible interface for QM/MM molecular dynamics simulations with AMBER, *J. Comput. Chem.* **35**, 95 (2014).

[82] Frederick R. Manby, Martina Stella, Jason D. Goodpaster, and Thomas F. Miller III: A simple, exact density-functional-theory embedding scheme, *J. Chem. Theory Comput.* **8**, 2564 (2012).

[83] F. Maseras and K. Morokuma: IMOMM – A new integrated ab-initio plus molecular mechanics geometry optimization scheme of equilibrium structures and transition-states, *J. Comput. Chem.* **16**, 1170 (1995).

[84] Thomas Kloss, Jochen Heil, and Stefan M. Kast: Quantum chemistry in solution by combining 3D integral equation theory with a cluster embedding approach, *J. Phys. Chem. B* **112**, 4337 (2008).

[85] Stefan M. Kast, Jochen Heil, Stefan Güssregen, and K. Friedemann Schmidt: Prediction of tautomer ratios by embedded-cluster integral equation theory, *J. Comput. Aided Mol. Des.* **24**, 343 (2010).

[86] C. Bannwarth, E. Caldeweyher, S. Ehlert, A. Hansen, P. Pracht, J. Seibert, S. Spicher, and S. Grimme: Extended tight-binding quantum chemistry methods, *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **11**, e1493 (2020).

[87] S. Grimme, C. Bannwarth, and 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 (2017).

[88] C. Bannwarth, S. Ehlert, and 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 (2019).

[89] S. Spicher and S. Grimme: Robust atomistic modeling of materials, organometallic, and biochemical systems, *Angew. Chem. Int. Ed.* **59**, 15665 (2020).

[90] S. Ehlert, M. Stahn, S. Spicher, and S. Grimme: Robust and efficient implicit solvation model for fast semiempirical methods, *J. Chem. Theory Comput.* **17**, 4250 (2021).

[91] Mopac2016, James J. P. Stewart, Stewart Computational Chemistry, web: http://OpenMOPAC.net.

[92] Dávid Mester and Mihály Kállay: Near-basis-set-limit double-hybrid DFT energies with exceptionally low computational costs, *J. Phys. Chem. Lett.* **16**, 2136–2143 (2025).

[93] S. Humbel, S. Sieber, and K. Morokuma: The IMOMO method: Integration of different levels of molecular orbital approximations for geometry optimization of large systems: Test for *n*-butane conformation and $S_N2$ reaction: $RCl+Cl^-$, *J. Chem. Phys.* **105**, 1959 (1996).

[94] Jacopo Tomasi, Benedetta Mennucci, and Roberto Cammi: Quantum mechanical continuum solvation models, *Chem. Rev.* **105**, 2999 (2005).

[95] Roberto Di Remigio, Krzysztof Mozgawa, Hui Cao, Ville Weijo, and Luca Frediani: A polarizable continuum model for molecules at spherical diffuse interfaces, *J. Chem. Phys.* **144**, 124103 (2016).

[96] Roberto Di Remigio, Arnfinn Hykkerud Steindal, Krzysztof Mozgawa, Ville Weijo, Hui Cao, and Luca Frediani: PCMSolver: An open-source library for solvation modeling, *Int. J. Quantum Chem.* **119**, e25685 (2019).

[97] `PCMSolver`, an open-source library for the polarizable continuum model electrostatic problem, written by R. Di Remigio, L. Frediani and contributors (see http://pcmsolver.readthedocs.io/), Accessed Jan 1, 2025.

[98] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*, McGraw-Hill: New York: 1989.

[99] C. C. J. Roothaan: Self-consistent field theory for open shells of electronic systems, *Rev. Mod. Phys.* **32**, 179 (1960).

[100] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic Structure Theory*, Wiley: Chichester: 2000.

[101] József Csóka and Mihály Kállay, *to be published*, 2020.

[102] W. Kohn and L. J. Sham: Self-consistent equations including exchange and correlation effects, *Phys. Rev.* **140**, A1133 (1965).

[103] Michael Filatov and Sason Shaik: Spin-restricted density functional approach to the open-shell problem, *Chem. Phys. Lett.* **288**, 689 (1998).

[104] Andreas Savin and Heinz-Jürgen Flad: Density functionals for the Yukawa electron-electron interaction, *Int. J. Quantum Chem.* **56**, 327 (1995).

[105] Stefan Grimme: Semiempirical hybrid density functional with perturbative second-order correlation, *J. Chem. Phys.* **124**, 034108 (2006).

[106] János G. Ángyán, Iann C. Gerber, Andreas Savin, and Julien Toulouse: van der Waals forces in density functional theory: Perturbational long-range electron-interaction corrections, *Phys. Rev. A* **72**, 012510 (2005).

[107] Cairedine Kalai and Julien Toulouse: A general range-separated double-hybrid density-functional theory, *J. Chem. Phys.* **148**, 164105 (2018).

[108] C. Møller and M. S. Plesset: Note on an approximation treatment for many-electron systems, *Phys. Rev.* **46**, 618 (1934).

[109] Stefan Grimme: Improved second-order Møller–Plesset perturbation theory by separate scaling of parallel- and antiparallel-spin pair correlation energies, *J. Chem. Phys.* **118**, 9095 (2003).

[110] Yousung Jung, Rohini C. Lochan, Anthony D. Dutoi, and Martin Head-Gordon: Scaled opposite-spin second order Møller–Plesset correlation energy: An economical electronic structure method, *J. Chem. Phys.* **121**, 9793 (2004).

[111] Stanislav Kedžuch, Matúš Milko, and Jozef Noga: Alternative formulation of the matrix elements in MP2-R12 theory, *Int. J. Quantum Chem.* **105**, 929 (2005).

[112] Rafał A. Bachorz, Florian A. Bischoff, Andreas Glöß, Christof Hättig, Sebastian Höfener, Wim Klopper, and David P. Tew: The MP2-F12 method in the TURBOMOLE program package, *J. Comput. Chem.* **32**, 2492 (2011).

[113] Filipp Furche: Molecular tests of the random phase approximation to the exchange-correlation energy functional, *Phys. Rev. B* **64**, 195120 (2001).

[114] Andreas Grüneis, Martijn Marsman, Judith Harl, Laurids Schimka, and Georg Kresse: Making the random phase approximation to electronic correlation accurate, *J. Chem. Phys.* **131**, 154115 (2009).

[115] Xinguo Ren, Patrick Rinke, Gustavo E. Scuseria, and Matthias Scheffler: Renormalized second-order perturbation theory for the electron correlation energy: Concept, implementation, and benchmarks, *Phys. Rev. B* **88**, 035120 (2013).

[116] A. Szabo and N. S. Ostlund: The correlation energy in the random phase approximation: Intermolecular forces between closed-shell systems, *J. Chem. Phys.* **67**, 4351 (1977).

[117] Andreas Heßelmann: Random-phase-approximation correlation method including exchange interactions, *Phys. Rev. A* **85**, 012517 (2012).

[118] O. Christiansen, H. Koch, and P. Jørgensen: The second-order approximate coupled cluster singles and doubles model CC2, *Chem. Phys. Lett.* **243**, 409 (1995).

[119] Arnim Hellweg, Sarah A. Grün, and Christof Hättig: Benchmarking the performance of spin-component scaled CC2 in ground and electronically excited states, *Phys. Chem. Chem. Phys.* **10**, 4119 (2008).

[120] Nina O. C. Winter and Christof Hättig: Scaled opposite-spin CC2 for ground and excited states with fourth order scaling computational costs, *J. Chem. Phys.* **134**, 184101 (2011).

[121] Rodney J Bartlett and David M Silver: Many-body perturbation theory applied to electron pair correlation energies. I. Closed-shell first-row diatomic hydrides, *J. Chem. Phys.* **62**, 3258 (1975).

[122] J. Čížek: On the correlation problem in atomic and molecular systems. Calculation of wavefunction components in Ursell-type expansion using quantum-field theoretical methods, *J. Chem. Phys.* **45**, 4256 (1966).

[123] G. D. Purvis III and R. J. Bartlett: A full coupled-cluster singles and doubles model: The inclusion of disconnected triples, *J. Chem. Phys.* **76**, 1910 (1982).

[124] J. Noga and R. J. Bartlett: The full CCSDT model for molecular electronic structure, *J. Chem. Phys.* **86**, 7041 (1987).

[125] S. A. Kucharski and R. J. Bartlett: The coupled-cluster single, double, triple, and quadruple excitation method, *J. Chem. Phys.* **97**, 4282 (1992).

[126] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon: A fifth-order perturbation comparison of electron correlation theories, *Chem. Phys. Lett.* **157**, 479 (1989).

[127] M. Urban, J. Noga, S. J. Cole, and R. J. Bartlett: Towards a full CCSDT model for electron correlation, *J. Chem. Phys.* **83**, 4041 (1985).

[128] S. A. Kucharski and R. J. Bartlett: An efficient way to include connected quadruple contributions into the coupled cluster method, *J. Chem. Phys.* **108**, 9221 (1998).

[129] S. A. Kucharski and R. J. Bartlett: Noniterative energy corrections through fifth-order to the coupled cluster singles and doubles method, *J. Chem. Phys.* **108**, 5243 (1998).

[130] T. D. Crawford and J. F. Stanton: Investigation of an asymmetric triple-excitation correction for coupled-cluster energies, *Int. J. Quantum Chem.* **70**, 601 (1998).

[131] Y. S. Lee, S. A. Kucharski, and R. J. Bartlett: A coupled cluster approach with triple excitations, *J. Chem. Phys.* **81**, 5906 (1984).

[132] H. Koch, O. Christiansen, P. Jørgensen, A. M. Sánchez de Merás, and T. Helgaker: The CC3 model: An iterative coupled cluster approach including connected triples, *J. Chem. Phys.* **106**, 1808 (1997).

[133] Christof Hättig, David P. Tew, and Andreas Köhn: Accurate and efficient approximations to explicitly correlated coupled-cluster singles and doubles, CCSD-F12, *J. Chem. Phys.* **132**, 231102 (2010).

[134] G. Knizia, T. B. Adler, and H.-J. Werner: Simplified CCSD(T)-F12 methods: Theory and benchmarks, *J. Chem. Phys.* **130**, 054104 (2009).

[135] Emmanuel Giner, Barthélémy Pradines, Anthony Ferté, Roland Assaraf, Andreas Savin, and Julien Toulouse: Curing basis-set convergence of wave-function theory using density-functional theory: A systematically improvable approach, *J. Chem. Phys.* **149**, 194301 (2018).

[136] Pierre-François Loos, Barthélémy Pradines, Anthony Scemama, Julien Toulouse, and Emmanuel Giner: A density-based basis-set correction for wave function theory, *J. Phys. Chem. Lett.* **10**, 2931 (2019).

[137] N. Oliphant and L. Adamowicz: Multireference coupled-cluster method using a single-reference formalism, *J. Chem. Phys.* **94**, 1229 (1991).

[138] P. Piecuch, N. Oliphant, and L. Adamowicz: A state-selective multireference coupled-cluster theory employing the single-reference formalism, *J. Chem. Phys.* **99**, 1875 (1993).

[139] R. J. Buenker and S. D. Peyerimhoff: Individualized configuration selection in CI calculations with subsequent energy extrapolation, *Theor. Chem. Acc.* **35**, 33 (1974).

[140] Stephen R. Langhoff and Ernest R. Davidson: Configuration interaction calculations on the nitrogen molecule, *Int. J. Quantum Chem.* **8**, 61 (1974).

[141] J. B. Foresman, M. Head-Gordon, J. A. Pople, and M. J. Frisch: Toward a systematic molecular orbital theory for excited states, *J. Phys. Chem.* **96**, 135 (1992).

[142] A. D. McLachlan and A. M. Ball: Time-dependent Hartree–Fock theory for molecules, *Rev. Mod. Phys.* **36**, 844 (1964).

[143] M. E. Casida, Recent advances in density functional methods, in *Computational Chemistry: Reviews of Current Trends*, edited by D. P. Chong: volume 1: World Scientific: Singapore: 1999.

[144] So Hirata and Martin Head-Gordon: Time-dependent density functional theory within the Tamm–Dancoff approximation, *Chem. Phys. Lett.* **314**, 291 (1999).

[145] Takeshi Yanai, David P Tew, and Nicholas C Handy: A new hybrid exchange-correlation functional using the Coulomb-attenuating method (CAM-B3LYP), *Chem. Phys. Lett.* **393**, 51 (2004).

[146] Yoshihiro Tawada, Takao Tsuneda, Susumu Yanagisawa, Takeshi Yanai, and Kimihiko Hirao: A long-range-corrected time-dependent density functional theory, *J. Chem. Phys.* **120**, 8425 (2004).

[147] Stefan Grimme and Frank Neese: Double-hybrid density functional theory for excited electronic states of molecules, *J. Chem. Phys.* **127**, 154116 (2007).

[148] Marcos Casanova-Páez, Michael B. Dardis, and Lars Goerigk: $\omega$B2PLYP and $\omega$B2GPPLYP: The first two double-hybrid density functionals with long-range correction optimized for excitation energies, *J. Chem. Theory Comput.* **15**, 4735 (2019).

[149] M. Head-Gordon, R. J. Rico, M. Oumi, and T. J. Lee: A doubles correction to electronic excited states from configuration interaction in the space of single substitutions, *Chem. Phys. Lett.* **219**, 21 (1994).

[150] Stefan Grimme and Ekaterina I. Izgorodina: Calculation of 0–0 excitation energies of organic molecules by CIS(D) quantum chemical methods, *Chem. Phys.* **305**, 223 (2004).

[151] Young Min Rhee and Martin Head-Gordon: Scaled second-order perturbation corrections to configuration interaction singles: Efficient and reliable excitation energy methods, *J. Phys. Chem. A* **111**, 5314 (2007).

[152] Martin Head-Gordon, Manabu Oumi, and David Maurice: Quasidegenerate second-order perturbation corrections to single-excitation configuration interaction, *Mol. Phys.* **96**, 593 (1999).

[153] Jochen Schirmer: Beyond the random-phase approximation: A new approximation scheme for the polarization propagator, *Phys. Rev. A* **26**, 2395 (1982).

[154] P. Piecuch, S. A. Kucharski, and R. J. Bartlett: Coupled-cluster methods with internal and semi-internal triply and quadruply excited clusters: CCSDt and CCSDtq approaches, *J. Chem. Phys.* **110**, 6103 (1999).

[155] WanZhen Liang and Martin Head-Gordon: Approaching the basis set limit in density functional theory calculations using dual basis sets without diagonalization, *J. Phys. Chem. A* **108**, 3206 (2004).

[156] R. B. Murphy, Y. Cao, M. D. Beachy, M. N. Ringnalda, and R. A. Friesner: Efficient pseudospectral methods for density functional calculations, *J. Chem. Phys.* **112**, 10131 (2000).

[157] F. Neese, F. Wennmohs, A. Hansen, and 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 (2009).

[158] Robert Polly, Hans-Joachim Werner, Frederick R. Manby, and Peter J. Knowles: Fast Hartree–Fock theory using local fitting approximations, *Mol. Phys.* **102**, 2311 (2004).

[159] Samuel Manzer, Paul R. Horn, Narbe Mardirossian, and Martin Head-Gordon: Fast, accurate evaluation of exact exchange: The occ-RI-K algorithm, *J. Chem. Phys.* **143**, 024113 (2015).

[160] S. Li, J. Ma, and Y. Jiang: Linear scaling local correlation approach for solving the coupled cluster equations of large systems, *J. Comput. Chem.* **23**, 237 (2002).

[161] H. Stoll: Correlation energy of diamond, *Phys. Rev. B* **46**, 6700 (1992).

[162] P. Pulay and S. Saebø: Orbital-invariant formulation and second-order gradient evaluation in Møller–Plesset perturbation theory, *Theor. Chem. Acc.* **69**, 357 (1986).

[163] Yasmine S. Al-Hamdani, Péter R. Nagy, Dennis Barton, Mihály Kállay, Jan Gerit Brandenburg, and Alexandre Tkatchenko: Interactions between large molecules pose a puzzle for reference quantum mechanical methods, *Nat. Commun.* **12**, 3927 (2021).

[164] Tamás Földes, Ádám Madarász, Ágnes Révész, Zoltán Dobi, Szilárd Varga, Andrea Hamza, Péter R. Nagy, Petri M. Pihko, and Imre Pápai: Stereocontrol in diphenylprolinol silyl ether catalyzed michael additions: Steric shielding or Curtin–Hammett scenario?, *J. Am. Chem. Soc.* **139**, 17052 (2017).

[165] Sanim Rahman, Vered Wineman-Fisher, Péter R. Nagy, Yasmine Al-Hamdani, Alexandre Tkatchenko, and Sameer Varma: Methyl-induced polarization destabilizes the noncovalent interactions of N-methylated lysines, *Chem. Eur. J.* **27**, 11005 (2021).

[166] Péter R. Nagy: State-of-the-art local correlation methods enable accurate and affordable gold standard quantum chemistry up to a few hundred atoms, *Chem. Sci.* **15**, 14556 (2024).

[167] Basis Set Exchange, https://www.basissetexchange.org/.

[168] David J. Feller: The role of databases in support of computational chemistry calculations, *J. Comput. Chem.* **17**, 1571 (1996).

[169] K. L. Schuchardt, B. T. Didier, T. Elsethagen, L. Sun, V. Gurumoorthi, J. Chase, J. Li, and T. L. Windus: Basis set exchange: A community database for computational sciences, *J. Chem. Inf. Model.* **47**, 1045 (2007).

[170] Benjamin P. Pritchard, Doaa Altarawy, Brett Didier, Tara D. Gibson, and Theresa L. Windus: A new basis set exchange: An open, up-to-date resource for the molecular sciences community, *J. Chem. Inf. Model.* **59**, 4814 (2019).

[171] Miguel A. L. Marques, Micael J. T. Oliveira, and Tobias Burnus: LIBXC: A library of exchange and correlation functionals for density functional theory, *Comput. Phys. Commun.* **183**, 2272 (2012).

[172] Susi Lehtola, Conrad Steigemann, Micael J. T. Oliveira, and Miguel A. L. Marques: Recent developments in LIBXC – A comprehensive library of functionals for density functional theory, *SoftwareX* **7**, 1 (2018).

[173] https://libxc.gitlab.io/, Accessed Jan 1, 2025.

[174] Axel D. Becke: A multicenter numerical integration scheme for polyatomic molecules, *J. Chem. Phys.* **88**, 2547 (1988).

[175] Oliver Treutler and Reinhart Ahlrichs: Efficient molecular numerical integration schemes, *J. Chem. Phys.* **102**, 346 (1995).

[176] Thom H. Dunning Jr.: Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen, *J. Chem. Phys.* **90**, 1007 (1989).

[177] Rick A. Kendall, Thom H. Dunning Jr., and Robert J. Harrison: Electron affinities of the first-row atoms revisited. Systematic basis sets and wave functions, *J. Chem. Phys.* **96**, 6796 (1992).

[178] David E. Woon and Thom H. Dunning Jr.: Gaussian basis sets for use in correlated molecular calculations. III. The atoms aluminum through argon, *J. Chem. Phys.* **98**, 1358 (1993).

[179] David E. Woon and Thom H. Dunning Jr.: Gaussian basis sets for use in correlated molecular calculations. V. Core-valence basis sets for boron through neon, *J. Chem. Phys.* **103**, 4572 (1995).

[180] Kirk A. Peterson and Thom H. Dunning Jr.: Accurate correlation consistent basis sets for molecular core-valence correlation effects: The second row atoms Al-Ar, and the first row atoms B-Ne revisited, *J. Chem. Phys.* **117**, 10548 (2002).

[181] Thom H. Dunning Jr., Kirk A. Peterson, and Angela K. Wilson: Gaussian basis sets for use in correlated molecular calculations. X. The atoms aluminum through argon revisited, *J. Chem. Phys.* **114**, 9244 (2001).

[182] P. C. Hariharan and J. A. Pople: The influence of polarization functions on molecular orbital hydrogenation energies, *Theor. Chem. Acc.* **28**, 213 (1973).

[183] R. Krishnan, J. S. Binkley, R. Seeger, and J. A. Pople: Self-consistent molecular orbital methods. XX. A basis set for correlated wave functions, *J. Chem. Phys.* **72**, 650 (1980).

[184] W. J. Hehre, R. Ditchfield, and J. A. Pople: Self-consistent molecular orbital methods. XII. Further extensions of Gaussian-type basis sets for use in molecular orbital studies of organic molecules, *J. Chem. Phys.* **56**, 2257 (1972).

[185] J. D. Dill and J. A. Pople: Self-consistent molecular orbital methods. XV. Extended Gaussian-type basis sets for lithium, beryllium, and boron, *J. Chem. Phys.* **62**, 2921 (1975).

[186] M. M. Francl, W. J. Petro, W. J. Hehre, J. S. Binkley, M. S. Gordon, D. J. DeFrees, and J. A. Pople: Self-consistent molecular orbital methods. XXIII. A polarization-type basis set for second-row elements, *J. Chem. Phys.* **77**, 3654 (1982).

[187] J. S. Binkley, J. A. Pople, and W. J. Hehre: Self-consistent molecular orbital methods. 21. Small split-valence basis sets for first-row elements, *J. Am. Chem. Soc.* **102**, 939 (1980).

[188] M. S. Gordon, J. S. Binkley, J. A. Pople, W. J. Pietro, and W. J. Hehre: Self-consistent molecular-orbital methods. 22. Small split-valence basis sets for second-row elements, *J. Am. Chem. Soc.* **104**, 2797 (1983).

[189] A. D. McLean and G. S. Chandler: Contracted Gaussian basis sets for molecular calculations. I. Second row atoms, Z=11-18, *J. Chem. Phys.* **72**, 5639 (1980).

[190] T. Clark, J. Chandrasekhar, G.W. Spitznagel, and P. v. R. Schleyer: Efficient diffuse function-augmented basis sets for anion calculations. III. The 3-21+G basis set for first-row elements, Li-F, *J. Comput. Chem.* **4**, 294 (1983).

[191] Florian Weigend and Reinhart 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**, 3297 (2005).

[192] Dmitrij Rappoport and Filipp Furche: Property-optimized Gaussian basis sets for molecular response calculations, *J. Chem. Phys.* **133**, 134105 (2010).

[193] Kirk A. Peterson, Thomas B. Adler, and Hans-Joachim Werner: Systematically convergent basis sets for explicitly correlated wavefunctions: The atoms H, He, B-Ne, and Al-Ar, *J. Chem. Phys.* **128**, 084102 (2008).

[194] Thom H. Dunning Jr. and P. Jeffrey Hay, Gaussian basis sets for molecular calculations, in *Methods of Electronic Structure Theory*, edited by Henry F. Schaefer III: volume 2: Plenum: New York: 1977.

[195] Florian Weigend, Marco Häser, Holger Patzelt, and Reinhart Ahlrichs: RI-MP2: optimized auxiliary basis sets and demonstration of efficiency, *Chem. Phys. Lett.* **294**, 143 (1998).

[196] Florian Weigend, Andreas Köhn, and Christof Hättig: Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations, *J. Chem. Phys.* **116**, 3175 (2002).

[197] Arnim Hellweg and Dmitrij Rappoport: 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, *Phys. Chem. Chem. Phys.* **17**, 1010 (2015).

[198] Florian Weigend: Hartree–Fock exchange fitting basis sets for H to Rn, *J. Comput. Chem.* **29**, 167 (2008).

[199] Florian Weigend: Accurate Coulomb-fitting basis sets for H to Rn, *Phys. Chem. Chem. Phys.* **8**, 1057 (2006).

[200] Kazim E. Yousaf and Kirk A. Peterson: Optimized auxiliary basis sets for explicitly correlated methods, *J. Chem. Phys.* **129**, 184108 (2008).

[201] Kazim E. Yousaf and Kirk A. Peterson: Optimized complementary auxiliary basis sets for explicitly correlated methods: aug-cc-pVnZ orbital basis sets, *Chem. Phys. Lett.* **476**, 303 (2009).

[202] J. Grant Hill, Shivnath Mazumder, and Kirk A. Peterson: Correlation consistent basis sets for molecular core-valence effects with explicitly correlated wave functions: The atoms B–Ne and Al–Ar, *J. Chem. Phys.* **132**, 054108 (2010).

[203] J. Grant Hill and Kirk A. Peterson: Explicitly correlated coupled cluster calculations for molecules containing group 11 (Cu, Ag, Au) and 12 (Zn, Cd, Hg) elements: Optimized complementary auxiliary basis sets for valence and core-valence basis sets, *J. Chem. Theory Comput.* **8**, 518 (2012).

[204] P. Jeffrey Hay and Willard R. Wadt: Ab initio effective core potentials for molecular calculations. Potentials for the transition metal atoms Sc to Hg, *J. Chem. Phys.* **82**, 270 (1985).

[205] Willard R. Wadt and P. Jeffrey Hay: Ab initio effective core potentials for molecular calculations. Potentials for main group elements Na to Bi, *J. Chem. Phys.* **82**, 284 (1985).

[206] P. Jeffrey Hay and Willard R. Wadt: Ab initio effective core potentials for molecular calculations. Potentials for K to Au including the outermost core orbitals, *J. Chem. Phys.* **82**, 299 (1985).

[207] Kirk A. Peterson: Systematically convergent basis sets with relativistic pseudopotentials. I. Correlation consistent basis sets for the post-d group 13-15 elements, *J. Chem. Phys.* **119**, 11099 (2003).

[208] Kirk A. Peterson, Detlev Figgen, Erich Goll, Hermann Stoll, and Michael Dolg: Systematically convergent basis sets with relativistic pseudopotentials. II. Small-core pseudopotentials and correlation consistent basis sets for the post-d group 16-18 elements, *J. Chem. Phys.* **119**, 11113 (2003).

[209] Kirk A Peterson and Cristina Puzzarini: Systematically convergent basis sets for transition metals. II. Pseudopotential-based correlation consistent basis sets for the group 11 (Cu, Ag, Au) and 12 (Zn, Cd, Hg) elements, *Theor. Chem. Acc.* **114**, 283 (2005).

[210] Kirk A. Peterson, Detlev Figgen, Michael Dolg, and Hermann Stoll: Energy-consistent relativistic pseudopotentials and correlation consistent basis sets for the 4d elements Y-Pd, *J. Chem. Phys.* **126**, 124101 (2007).

[211] Detlev Figgen, Kirk A. Peterson, Michael Dolg, and Hermann Stoll: Energy-consistent pseudopotentials and correlation consistent basis sets for the 5d elements Hf-Pt, *J. Chem. Phys.* **130**, 164108 (2009).

[212] Kirk A. Peterson and Kazim E. Yousaf: Molecular core-valence correlation effects involving the post-d elements Ga-Rn: Benchmarks and new pseudopotential-based correlation consistent basis sets, *J. Chem. Phys.* **133**, 174116 (2010).

[213] Christof Hättig: Optimization of auxiliary basis sets for RI-MP2 and RI-CC2 calculations: Core-valence and quintuple-$\zeta$ basis sets for H to Ar and QZVPP basis sets for Li to Kr, *Phys. Chem. Chem. Phys.* **7**, 59 (2005).

[214] J. Grant Hill: Auxiliary basis sets for density-fitting second-order Møller–Plesset perturbation theory: Weighted core-valence correlation consistent basis sets for the 4d elements Y-Pd, *J. Comput. Chem.* **34**, 2168 (2013).

[215] Stella Kritikou and J. Grant Hill: Auxiliary basis sets for density fitting in explicitly correlated calculations: The atoms H–Ar, *J. Chem. Theory Comput.* **11**, 5269 (2015).

[216] A correlation consistent basis sets repository, http://www.grant-hill.group.shef.ac.uk/ccrepo/, Accessed Jan 1, 2025.

[217] James W. Boughton and Peter Pulay: Comparison of the Boys and Pipek–Mezey localizations in the local correlation approach and automatic virtual basis selection, *J. Comput. Chem.* **14**, 736 (1993).

[218] Julien Toulouse, Wuming Zhu, Andreas Savin, Georg Jansen, and János G. Ángyán: Closed-shell ring coupled cluster doubles theory with range separation applied on weak intermolecular interactions, *J. Chem. Phys.* **135**, 084119 (2011).

[219] Christof Hättig and Florian Weigend: CC2 excitation energy calculations on large molecules using the resolution of the identity approximation, *J. Chem. Phys.* **113**, 5154 (2000).

[220] Nisha Mehta and Jan M. L. Martin: Explicitly correlated double-hybrid DFT: A comprehensive analysis of the basis set convergence on the GMTKN55 database, *J. Chem. Theory Comput.* **18**, 5978 (2022).

[221] Sonia Coriani and Henrik Koch: Communication: X-ray absorption spectra and core-ionization potentials within a core-valence separated coupled cluster framework, *J. Chem. Phys.* **143**, 181103 (2015).

[222] L. S. Cederbaum, W. Domcke, and J. Schirmer: Many-body theory of core holes, *Phys. Rev. A* **22**, 206 (1980).

[223] Akio Takatsuka, Seiichiro Ten-no, and Wolfgang Hackbusch: Minimax approximation for the decomposition of energy denominators in Laplace-transformed Møller–Plesset perturbation theories, *J. Chem. Phys.* **129**, 044112 (2008).

[224] Alejandro J. Garza, Ireneusz W. Bulik, Thomas M. Henderson, and Gustavo E. Scuseria: Range separated hybrids of pair coupled cluster doubles and density functionals, *Phys. Chem. Chem. Phys.* **17**, 22412 (2015).

[225] Functionals were obtained from the Density Functional Repository as developed and distributed by the Quantum Chemistry Group, CCLRC Daresbury Laboratory, Daresbury, Cheshire, WA4 4AD United Kingdom. Contact Huub van Dam (h.j.j.vandam@dl.ac.uk) or Paul Sherwood for further information.

[226] R. Strange, F. R. Manby, and P. J. Knowles: Automatic code generation in density functional theory, *Comput. Phys. Commun.* **136**, 310 (2001).

[227] Stefan Grimme, Jens Antony, Stephan Ehrlich, and Helge 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).

[228] Stefan Grimme, Stephan Ehrlich, and Lars Goerigk: Effect of the damping function in dispersion corrected density functional theory, *J. Comput. Chem.* **32**, 1456 (2011).

[229] P. A. M. Dirac: Quantum mechanics of many-electron systems, *Proc. R. Soc. London A* **123**, 714 (1929).

[230] J. C. Slater: A simplification of the Hartree–Fock method, *Phys. Rev.* **81**, 385 (1951).

[231] S. H. Vosko, L. Wilk, and M. Nusair: Accurate spin-dependent electron liquid correlation energies for local spin density calculations: A critical analysis, *Can. J. Phys.* **58**, 1200 (1980).

[232] J. P. Perdew and A. Zunger: Self-interaction correction to density-functional approximations for many-electron systems, *Phys. Rev. B* **23**, 5048 (1981).

[233] J. P. Perdew and Y. Wang: Accurate and simple analytic representation of the electron-gas correlation energy, *Phys. Rev. B* **45**, 13244 (1992).

[234] Axel D. Becke: Density-functional exchange-energy approximation with correct asymptotic-behavior, *Phys. Rev. A* **38**, 3098 (1988).

[235] John P. Perdew, Kieron Burke, and Matthias Ernzerhof: Generalized gradient approximation made simple, *Phys. Rev. Lett.* **77**, 3865 (1996).

[236] M. Ernzerhof and J. P. Perdew: Generalized gradient approximation to the angle- and system-averaged exchange hole, *J. Chem. Phys.* **109**, 3313 (1998).

[237] J. P. Perdew, J. A. Chevary, S. H. Vosko, K. A. Jackson, M. R. Pederson, D. J. Singh, and C. Fiolhais: Atoms, molecules, solids and surfaces: Applications of the generalized gradient approximation for exchange and correlation, *Phys. Rev. B* **46**, 6671 (1992).

[238] C. Adamo and V. Barone: Exchange functionals with improved long-range behavior and adiabatic connection methods without adjustable parameters: The mPW and mPW1PW models, *J. Chem. Phys.* **108**, 664 (1998).

[239] Javier Carmona-Espíndola, José L. Gázquez, Alberto Vela, and S. B. Trickey: Generalized gradient approximation exchange energy functional with near-best semilocal performance, *J. Chem. Theory Comput.* **15**, 303 (2019).

[240] C. Lee, W. Yang, and R. G. Parr: Development of the Colle–Salvetti correlation-energy formula into a functional of the electron density, *Phys. Rev. B* **37**, 785 (1988).

[241] John P. Perdew: Density-functional approximation for the correlation energy of the inhomogeneous electron gas, *Phys. Rev. B* **33**, 8822 (1986).

[242] A. Daniel Boese, Nikos L. Doltsinis, Nicholas C. Handy, and Michiel Sprik: New generalized gradient approximation functionals, *J. Chem. Phys.* **112**, 1670 (2000).

[243] A. Daniel Boese and Nicholas C. Handy: A new parametriztion of exchange-correlation generalized gradient approximation functionals, *J. Chem. Phys.* **114**, 5497 (2001).

[244] Xin Xu and William A. Goddard III: The X3LYP extended density functional for accurate descriptions of nonbond interactions, spin states, and thermochemical properties, *Proc. Natl. Acad. Sci. U.S.A.* **101**, 2673 (2004).

[245] E. E. Dahlke and D. G. Truhlar: Improved density functionals for water, *J. Phys. Chem. B* **109**, 15677 (2005).

[246] Axel D. Becke: A new mixing of Hartree–Fock and local density-functional theories, *J. Chem. Phys.* **98**, 1372 (1993).

[247] Axel D. Becke: Density-functional thermochemistry. III. The role of exact exchange, *J. Chem. Phys.* **98**, 5648 (1993).

[248] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch: Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields, *J. Phys. Chem.* **98**, 11623 (1994).

[249] C. Adamo and V. Barone: Toward reliable adiabatic connection models free from adjustable parameters, *Chem. Phys. Lett.* **274**, 242 (1997).

[250] A. J. Cohen and N. C. Handy: Dynamic correlation, *Mol. Phys.* **99**, 607 (2001).

[251] Axel D. Becke: Density-functional thermochemistry. V. Systematic optimization of exchange-correlation functionals, *J. Chem. Phys.* **107**, 8554 (1997).

[252] John P. Perdew, Matthias Ernzerhof, and Kieron Burke: Rationale for mixing exact exchange with density functional approximations, *J. Chem. Phys.* **105**, 9982 (1996).

[253] J. Tao, J. P. Perdew, V. N. Staroverov, and G. E. Scuseria: Climbing the density functional ladder: Nonempirical meta-generalized gradient approximation designed for molecules and solids, *Phys. Rev. Lett.* **91**, 146401 (2003).

[254] J. P. Perdew, A. Ruzsinszky, G. I. Csonka, L. A. Constantin, and J. Sun: Workhorse semilocal density functional for condensed matter physics and quantum chemistry, *Phys. Rev. Lett.* **103**, 026403 (2009).

[255] Jianwei Sun, Adrienn Ruzsinszky, and John P. Perdew: Strongly constrained and appropriately normed semilocal density functional, *Phys. Rev. Lett.* **115**, 036402 (2015).

[256] A. D. Becke: Density-functional thermochemistry. IV. A new dynamical correlation functional and implications for exact-exchange mixing, *J. Chem. Phys.* **104**, 1040 (1996).

[257] Pál D. Mezei, Gábor I. Csonka, and Mihály Kállay: Simple modifications of the SCAN meta-generalized gradient approximation functional, *J. Chem. Theory Comput.* **14**, 2469 (2018).

[258] Y. Zhao and 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 (2006).

[259] Y. Zhao and D. G. Truhlar: A new local density functional for main-group thermochemistry, transition metal bonding, thermochemical kinetics, and noncovalent interactions, *J. Chem. Phys.* **125**, 194101 (2006).

[260] Narbe Mardirossian and Martin Head-Gordon: Mapping the genome of meta-generalized gradient approximation density functionals: The search for B97M-V, *J. Chem. Phys.* **142**, 074111 (2015).

[261] Yan Zhao and Donald G. Truhlar: Density functional for spectroscopy: No long-range self-interaction error, good performance for Rydberg and charge-transfer states, and better performance on average than B3LYP for ground states, *J. Phys. Chem. A* **110**, 13126 (2006).

[262] Y. Zhao and D. G. Truhlar: Exploring the limit of accuracy of the global hybrid meta density functional for main-group thermochemistry, kinetics, and noncovalent interactions, *J. Chem. Theory Comput.* **4**, 1849 (2008).

[263] V. N. Staroverov, G. E. Scuseria, J. Tao, and J. P. Perdew: Comparative assessment of a new nonempirical density functional: Molecules and hydrogen-bonded complexes, *J. Chem. Phys.* **119**, 12129 (2003).

[264] G. I. Csonka, J. P. Perdew, and A. Ruzsinszky: Global hybrid functionals: A look at the engine under the hood, *J. Chem. Theory Comput.* **6**, 3688 (2010).

[265] Y. Zhao and D. G. Truhlar: Hybrid meta density functional theory methods for thermochemistry, thermochemical kinetics, and noncovalent interactions: The MPW1B95 and MPWB1K models and comparative assessments for hydrogen bonding and van der Waals interactions, *J. Phys. Chem. A* **108**, 6908 (2004).

[266] Y. Zhao and D. G. Truhlar: Design of density functionals that are broadly accurate for thermochemistry, thermochemical kinetics, and nonbonded interactions, *J. Phys. Chem. A* **109**, 5656 (2005).

[267] Haoyu S. Yu, Xiao He, Shaohong L. Li, and Donald G. Truhlar: MN15: A Kohn–Sham global-hybrid exchange-correlation density functional with broad accuracy for multi-reference and single-reference systems and noncovalent interactions, *Chem. Sci.* **7**, 5032 (2016).

[268] Kerwin Hui and Jeng-Da Chai: SCAN-based hybrid and double-hybrid density functionals from models without fitted parameters, *J. Chem. Phys.* **144**, 044114 (2016).

[269] Erich Goll, Hans-Joachim Werner, and Hermann Stoll: A short-range gradient-corrected density functional in long-range coupled-cluster calculations for rare gas dimers, *Phys. Chem. Chem. Phys.* **7**, 3917 (2005).

[270] Jochen Heyd, Gustavo E. Scuseria, and Matthias Ernzerhof: Hybrid functionals based on a screened Coulomb potential, *J. Chem. Phys.* **118**, 8207 (2003).

[271] Aliaksandr V. Krukau, Oleg A. Vydrov, Artur F. Izmaylov, and Gustavo E. Scuseria: Influence of the exchange screening parameter on the performance of screened hybrid functionals, *J. Chem. Phys.* **125**, 224106 (2006).

[272] Oleg A. Vydrov and Gustavo E. Scuseria: Assessment of a long-range corrected hybrid functional, *J. Chem. Phys.* **125**, 234109 (2006).

[273] Jeng-Da Chai and Martin Head-Gordon: Systematic optimization of long-range corrected hybrid density functionals, *J. Chem. Phys.* **128**, 084106 (2008).

[274] Narbe Mardirossian and Martin Head-Gordon: $\omega$B97X-V: A 10-parameter, range-separated hybrid, generalized gradient approximation density functional with nonlocal correlation, designed by a survival-of-the-fittest strategy, *Phys. Chem. Chem. Phys.* **16**, 9904 (2014).

[275] Roberto Peverati and Donald G. Truhlar: Improving the accuracy of hybrid meta-GGA density functionals by range separation, *J. Phys. Chem. Lett.* **2**, 2810 (2011).

[276] Roberto Peverati and Donald G. Truhlar: Screened-exchange density functionals with broad accuracy for chemistry and solid-state physics, *Phys. Chem. Chem. Phys.* **14**, 16187 (2012).

[277] Narbe Mardirossian and Martin Head-Gordon: $\omega$B97M-V: A combinatorially optimized, range-separated hybrid, meta-GGA density functional with VV10 nonlocal correlation, *J. Chem. Phys.* **144**, 214110 (2016).

[278] Amir Karton, Alex Tarnopolsky, Jean-Francois Lamère, George C. Schatz, and Jan M. L. 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**, 12868 (2008).

[279] Sebastian Kozuch and Jan M. L. Martin: DSD-PBEP86: in search of the best double-hybrid DFT with spin-component scaled MP2 and dispersion corrections, *Phys. Chem. Chem. Phys.* **13**, 20104 (2011).

[280] Sebastian Kozuch and Jan M. L. Martin: Spin-component-scaled double hybrids: An extensive search for the best fifth-rung functionals blending DFT and perturbation theory, *J. Comput. Chem.* **34**, 2327 (2013).

[281] Ying Zhang, Xin Xu, and William A. Goddard III: Doubly hybrid density functional for accurate descriptions of nonbond interactions, thermochemistry, and thermochemical kinetics, *Proc. Natl. Acad. Sci. U.S.A.* **106**, 4963 (2009).

[282] Lars Goerigk and Stefan Grimme: Efficient and accurate double-hybrid-meta-GGA density functionals – Evaluation with the extended GMTKN30 database for general main group thermochemistry, kinetics, and noncovalent interactions, *J. Chem. Theory Comput.* **7**, 291 (2011).

[283] Brina Brauer, Manoj K. Kesharwani, Sebastian Kozuch, and Jan M. L. Martin: The S66x8 benchmark for noncovalent interactions revisited: explicitly correlated ab initio methods and density functional theory, *Phys. Chem. Chem. Phys.* **18**, 20905 (2016).

[284] Oleg A. Vydrov and Troy Van Voorhis: Nonlocal van der Waals density functional: The simpler the better, *J. Chem. Phys.* **133**, 244103 (2010).

[285] Golokesh Santra, Emmanouil Semidalas, and Jan M. L. Martin: Surprisingly good performance of XYG3 family functionals using a scaled KS-MP3 correlation, *J. Phys. Chem. Lett.* **12**, 9368 (2021).

[286] Golokesh Santra, Emmanouil Semidalas, and Jan M. L. Martin: Exploring avenues beyond revised DSD functionals: II. Random-phase approximation and scaled MP3 corrections, *J. Phys. Chem. A* **125**, 4628 (2021).

[287] Jeppe Olsen, Poul Jørgensen, and Jack Simons: Passing the one-billion limit in full configuration-interaction (FCI) calculations, *Chem. Phys. Lett.* **169**, 463 (1990).

[288] Mihály Kállay and Péter R. Surján: Computing coupled-cluster wave functions with arbitrary excitations, *J. Chem. Phys.* **113**, 1359 (2000).

[289] D. Andrae, U. Häußermann, M. Dolg, H. Stoll, and H. Preuß: Energy-adjusted *ab initio* pseudopotentials for the second and third row transition elements, *Theor. Chem. Acc.* **77**, 123 (1990).

[290] M. Kaupp, P. v. R. Schleyer, H. Stoll, and H. Preuss: Pseudopotential approaches to Ca, Sr, and Ba hydrides. Why are some alkaline earth MX2 compounds bent?, *J. Chem. Phys.* **94**, 1360 (1991).

[291] Thierry Leininger, Andreas Nicklass, Wolfgang Küchle, Hermann Stoll, Michael Dolg, and Andreas Bergner: The accuracy of the pseudopotential approximation: non-frozen-core effects for spectroscopic constants of alkali fluorides XF (X = K, Rb, Cs), *Chem. Phys. Lett.* **255**, 274 (1996).

[292] Bernhard Metz, Marcus Schweizer, Hermann Stoll, Michael Dolg, and Wenjian Liu: A small-core multiconfiguration Dirac–Hartree–Fock-adjusted pseudopotential for Tl – application to TlX (X = F, Cl, Br, I), *Theor. Chem. Acc.* **104**, 22 (2000).

[293] Bernhard Metz, Hermann Stoll, and Michael Dolg: Small-core multiconfiguration-Dirac–Hartree–Fock-adjusted pseudopotentials for post-d main group elements: Application to PbH and PbO, *J. Chem. Phys.* **113**, 2563 (2000).

[294] K. A. Peterson, B. C. Shepler, D. Figgen, and H. Stoll: On the spectroscopic and thermochemical properties of ClO, BrO, IO, and their anions, *J. Phys. Chem. A* **110**, 13877 (2006).

[295] Detlev Figgen, Guntram Rauhut, Michael Dolg, and Hermann Stoll: Energy-consistent pseudopotentials for group 11 and 12 atoms: adjustment to multi-configuration Dirac–Hartree–Fock data, *Chem. Phys.* **311**, 227 (2005).

[296] Daniel Claudino and Nicholas J. Mayhall: Automatic partition of orbital spaces based on singular value decomposition in the context of embedding theories, *J. Chem. Theory Comput.* **15**, 1053 (2019).

[297] Ádám B. Szirmai, Bence Hégely, Attila Tajti, Mihály Kállay, and Péter G. Szalay: Projected atomic orbitals as optimal virtual space for excited state projection-based embedding calculations, *J. Chem. Theory Comput.* **20**, 3420 (2024).

[298] Curt M. Breneman and Kenneth B. Wiberg: Determining atom-centered monopoles from molecular electrostatic potentials. The need for high sampling density in formamide conformational analysis, *J. Comput. Chem.* **11**, 361 (1990).

[299] U. Chandra Singh and Peter A. Kollman: An approach to computing electrostatic charges for molecules, *J. Comput. Chem.* **5**, 129 (1984).

[300] David P. Tew and Wim Klopper: New correlation factors for explicitly correlated electronic wave functions, *J. Chem. Phys.* **123**, 074101 (2005).

[301] Reinhart Ahlrichs: Efficient evaluation of three-center two-electron integrals over Gaussian functions, *Phys. Chem. Chem. Phys.* **6**, 5119 (2004).

[302] S. Reine, E. Tellgren, and T. Helgaker: A unified scheme for the calculation of differentiated and undifferentiated molecular integrals over solid-harmonic Gaussians, *Phys. Chem. Chem. Phys.* **9**, 4771 (2007).

[303] Matthias Krack and Andreas M. Köster: An adaptive numerical integrator for molecular integrals, *J. Chem. Phys.* **108**, 3226 (1998).

[304] Christopher W. Murray, Nicholas C. Handy, and Gregory J. Laming: Quadrature schemes for integrals of density functional theory, *Mol. Phys.* **78**, 997 (1993).

[305] M. E. Mura and P. J. Knowles: Improved radial grids for quadrature in molecular density functional calculations, *J. Chem. Phys.* **104**, 9848 (1996).

[306] Ciro A. Guido, Pietro Cortona, Benedetta Mennucci, and Carlo Adamo: On the metric of charge transfer molecular excitations: A simple chemical descriptor, *J. Chem. Theory Comput.* **9**, 3118 (2013).

[307] Jetze Sikkema, Lucas Visscher, Trond Saue, and Miroslav Iliaš: The molecular mean-field approach for correlated relativistic calculations, *J. Chem. Phys.* **131**, 124116 (2009).

[308] S. Obara and A. Saika: Efficient recursive computation of molecular integrals over Cartesian Gaussian functions, *J. Chem. Phys.* **84**, 3963 (1986).

[309] R. Lindh, U. Ryu, and B. Liu: The reduced multiplication scheme of the Rys quadrature and new recurrence relations for auxiliary function based two-electron integral evaluation, *J. Chem. Phys.* **95**, 5889 (1991).

[310] Harry F. King and Michel Dupuis: Numerical integration using Rys polynomials, *J. Comput. Phys.* **21**, 144 (1976).

[311] N. Flocke: On the use of shifted Jacobi polynomials in accurate evaluation of roots and weights of Rys polynomials, *J. Chem. Phys.* **131**, 064107 (2009).

[312] Christoph Riplinger and Frank Neese: An efficient and near linear scaling pair natural orbital based local coupled cluster method, *J. Chem. Phys.* **138**, 034106 (2013).

[313] Peter Pinski, Christoph Riplinger, Edward F. Valeev, and F. Neese: Sparse maps—A systematic infrastructure for reduced-scaling electronic structure methods. I. An efficient and simple linear scaling local MP2 method that uses an intermediate basis of pair natural orbitals, *J. Chem. Phys.* **143**, 034108 (2015).

[314] S. J. Mo, T. Vreven, B. Mennucci, K. Morokuma, and J. Tomasi: Theoretical study of the $S_N2$ reaction of $Cl^-(H_2O)+CH_3Cl$ using our own N-layered integrated molecular orbital and molecular mechanics polarizable continuum model method (ONIOM, PCM), *Theor. Chem. Acc.* **111**, 154 (2004).

[315] T. Vreven, B. Mennucci, C. O. da Silva, K. Morokuma, and J. Tomasi: The ONIOM-PCM method: Combining the hybrid molecular orbital method and the polarizable continuum model for solvation. Application to the geometry and properties of a merocyanine in solution, *J. Chem. Phys.* **115**, 62 (2001).

[316] J. A. Nelder and R. Mead: A simplex method for function minimization, *Comput. J.* **7**, 308 (1965).

[317] Francesco Aquilante, Thomas Bondo Pedersen, Alfredo M. Sánchez de Merás, and Henrik Koch: Fast noniterative orbital localization for large molecules, *J. Chem. Phys.* **125**, 174101 (2006).

[318] J. M. Foster and S. F. Boys: Canonical configurational interaction procedure, *Rev. Mod. Phys.* **32**, 300 (1960).

[319] J. Pipek and P. Mezey: A fast intrinsic localization procedure applicable for ab initio and semiempirical linear combination of atomic orbital wave functions, *J. Chem. Phys.* **90**, 4916 (1989).

[320] Gerald Knizia: Intrinsic atomic orbitals: An unbiased bridge between quantum theory and chemical concepts, *J. Chem. Theory Comput.* **9**, 4834 (2013).

[321] B. Jansík, S. Høst, K. Kristensen, and P. Jørgensen: Local orbitals by minimizing powers of the orbital variance, *J. Chem. Phys.* **134**, 194104 (2011).

[322] Pavel Neogrády, Michal Pitoňák, and Miroslav Urban: Optimized virtual orbitals for correlated calculations: An alternative approach, *Mol. Phys.* **103**, 2141 (2005).

[323] R. S. Mulliken: Electronic population analysis on LCAO-MO molecular wave functions. I, *J. Chem. Phys.* **23**, 1833 (1955).

[324] I. Mayer: Charge, bond order and valence in the ab initio SCF theory, *Chem. Phys. Lett.* **97**, 270 (1983).

[325] Frank Neese: Importance of direct spin–spin coupling and spin-flip excitations for the zero-field splittings of transition metal complexes: A case study, *J. Am. Chem. Soc.* **128**, 10213 (2006).

[326] Georg Hetzer, Peter Pulay, and Hans-Joachim Werner: Multipole approximation of distant pair energies in local MP2 calculations, *Chem. Phys. Lett.* **290**, 143 (1998).

[327] Gijs Schaftenaar and Jan H. Noordik: Molden: a pre- and post-processing program for molecular and electronic structures, *J. Comput.-Aided Mol. Design* **14**, 123 (2000).

# List of keywords

active, 40
agrid, 41
agrid_pssp, 42
agrid_pssp_sm, 42

basis, 42
basis_sm, 46
basopt, 47
bfbasis, 47
bfgsmem, 47
bfgstol, 48
boysalg, 48
bp_subsyso, 49
bp_subsysv, 49
bpcompo, 48
bpcompv, 48
bpedo, 48
bpedv, 49
bppdo, 50
bppdv, 50

cabscorr, 50
calc, 50
ccmaxit, 56
ccprog, 57
ccsdalg, 57
ccsdmkl, 58
ccsdrest, 58
ccsdthreads, 58
cctol, 58
charge, 58
cialg, 59
ciguess, 59
cmpgrp, 60
comprest, 60
core, 61
corembed, 61
csapprox, 63
cvs, 63

dboc, 63
deltaf12, 64
dendec, 64
dens, 65
denscorr, 65
dfalg, 66
dfbasis_cab, 67
dfbasis_cor, 67
dfbasis_scf, 69
dfbasis_scf_sm, 69
dfintran, 69
dft, 69
dhexc, 81
diag, 81
docc, 82
domrad, 82
drpaalg, 83
dual, 83
dual_df, 84
dyson, 85

ecp, 85
edisp, 87
edisp_embed, 88
embed, 89
epert, 94
eps, 95
espcharge, 95
etemp, 89
excrad, 95
excrad_f12, 96
excrad_fin, 97

fitting, 97
fmm, 97
fmmord, 97
freq, 98

gamma, 98
gauss, 98
geom, 98
ghost, 100
gopt, 101
grdens, 102
grid, 102
grid_sm, 103
grtol, 103
gtol, 104
guido_ct, 104

hamilton, 104

iface, 104
incore, 105
intalg, 105
ip_ea, 106
itol, 106

laptol, 106
lccoporder, 106
lccrest, 107
lcorthr, 108
lmp2dens, 109
lnoepso, 110
lnoepsv, 110
localcc, 111
localcorrsymm, 112

mact, 113
maxact, 114
maxdim, 114
maxex, 114
maxmicroit, 114
mcscfiguess, 115
mem, 115
mmprog, 115
molden, 115
mpitasks, 116
mulmet, 116
mult, 116

nab, 116
nacto, 117
nactv, 117
naf_cor, 118
naf_f12, 118
naf_scf, 119
nafalg, 117
nafdens, 118
naftyp, 119
nchol, 119
ndeps, 119
nsing, 120
nstate, 120
nto, 120

ntrip, 120
numgrad, 121

occ, 121
occri, 122
oniom, 122
oniom_eechg, 127
oniom_pcm, 128
oniom_qcorr, 130
optalg, 132
optetol, 132
optex, 132
optgtol, 133
optmaxit, 132
optstol, 133
orblocc, 133
orbloce, 133
orblocguess, 134
orbloco, 135
orblocv, 135
osveps, 136
ovirt, 136
ovltol, 137
ovosnorb, 137

pcm, 138
pcm_*, 138
popul, 139
pressure, 139
pssp, 139
ptfreq, 140
ptthreads, 140

qmmm, 140
qro, 141
qscf, 141

redcost_exc, 142
redcost_tddft, 143
refdet, 143
rest, 145
rgrid, 146
rism, 147
rohfcore, 148
rohftype, 149

scf_conv, 149
scfalg, 149
scfdamp, 150
scfdamp_dampstep, 151
scfdamp_dtol, 151
scfdamp_end, 151
scfdamp_maxfact, 151
scfdamp_minfact, 151
scfdamp_mode, 152
scfdiis, 152
scfdiis_delmax, 152
scfdiis_dtol, 153
scfdiis_end, 153
scfdiis_start, 153
scfdiis_step, 153
scfdiis_watch, 153
scfdiis_wlimit, 154
scfdiis_wrange, 154
scfdtol, 154
scfext, 154
scfguessdens, 154
scfiguess, 155
scfloc, 156
scflshift, 157
scflshift_dtol, 157
scflshift_end, 157
scflshift_gaptol, 158
scfmaxit, 158
scftol, 158
scftype, 158
scspe, 159
scsph, 159
scsps, 159
scsps_t, 160
scspt, 160
scspt_t, 160
spairtol, 161
sqmprog, 161
subminp, 161
subsys_bopu, 162
symm, 163

talg, 163
temp, 164
test, 164
tlmo, 164
tpao, 165
tprint, 165

uncontract, 165
unit, 165
usedisk, 165

verbosity, 166

wpairtol, 166