MOLPRO

Users Manual
Version 2010.1

H.-J. Werner

Institut für Theoretische Chemie
Universität Stuttgart
Pfaffenwaldring 55
D-70569 Stuttgart
Federal Republic of Germany

P. J. Knowles

School of Chemistry
Cardiff University
Main Building, Park Place, Cardiff CF10 3AT
United Kingdom

SHA1 fd122acbd35ef3619fab5330ac0556b6c1fa4be2

Introduction to MOLPRO

MOLPRO is a complete system of *ab initio* programs for molecular electronic structure calculations, designed and maintained by H.-J. Werner and P. J. Knowles, and containing contributions from a number of other authors. As distinct from other commonly used quantum chemistry packages, the emphasis is on highly accurate computations, with extensive treatment of the electron correlation problem through the multiconfiguration-reference CI, coupled cluster and associated methods. The recently developed explicitly correlated coupled-cluster methods yield CCSD(T) results with near basis set limit accuracy already with double$-\zeta$ or triple$-\zeta$ basis sets, thus reducing the computational effort for calculations of this quality by two orders of magnitude. Using local electron correlation methods, which significantly reduce the increase of the computational cost with molecular size, accurate *ab initio* calculations can be performed for much larger molecules than with most other programs. These methods have recently been augmented by explicitly correlated terms, which strongly reduce both the basis set truncation errors and the errors of the local approximations.

The heart of the program consists of the multiconfiguration SCF, multireference CI, and coupled-cluster routines, and these are accompanied by a full set of supporting features. The package comprises

- Integral generation for generally contracted symmetry adapted gaussian basis functions ($spdfghi$). There are two programs with identical functionality: the preferred code is SEWARD (R. Lindh) which is the best on most machines; ARGOS (R. M. Pitzer) is available as an alternative, and in some cases is optimum for small memory scalar machines. Also two different gradient integral codes, namely CADPAC (R. Amos) and ALASKA (R. Lindh) are available. Only the latter allows the use of generally contracted symmetry adapted gaussian basis functions.
- Effective Core Potentials (contributions from H. Stoll).
- Many one-electron properties.
- Some two-electron properties, e.g. L_x^2, L_y^2, L_z^2, L_xL_y etc..
- Closed-shell and open-shell (spin restricted and unrestricted) self consistent field.
- Density-functional theory in the Kohn-Sham framework with various gradient corrected exchange and correlation potentials.
- Multiconfiguration self consistent field. This is the quadratically convergent MCSCF procedure described in J. Chem. Phys. 82 (1985) 5053. The program can optimize a weighted energy average of several states, and is capable of treating both completely general configuration expansions and also long CASSCF expansions as described in Chem. Phys. Letters 115 (1985) 259.
- Multireference CI. As well as the usual single reference function approaches (MP2, SDCI, CEPA), this module implements the internally contracted multireference CI method as described in J. Chem. Phys. 89 (1988) 5803 and Chem. Phys. Lett. 145 (1988) 514. Non variational variants (e.g. MR-ACPF), as described in Theor. Chim. Acta 78 (1990) 175, are also available. Electronically excited states can be computed as described in Theor. Chim. Acta, **84** 95 (1992).
- Multireference second-order and third-order perturbation theory (MRPT2, CASPT2, MRPT3) as described in Mol. Phys. **89**, 645 (1996) and J. Chem. Phys. **112**, 5546 (2000).

- Møller-Plesset perturbation theory (MPPT), Coupled-Cluster (CCSD), Quadratic configuration interaction (QCISD), and Brueckner Coupled-Cluster (BCCD) for closed shell systems, as described in Chem. Phys. Lett. 190 (1992) 1. Perturbative corrections for triple excitations can also be calculated (Chem. Phys. Letters **227** (1994) 321).
- Open-shell coupled cluster theories as described in J. Chem. Phys. 99 (1993) 5219, Chem. Phys. Letters **227** (1994) 321.
- An interface to the MRCC program of M. Kallay, allowing coupled-cluster calculations with arbitrary excitation level.
- Full Configuration Interaction. This is the determinant based benchmarking program described in Comp. Phys. Commun. 54 (1989) 75.
- Analytical energy gradients for SCF, DFT, state-averaged MCSCF/CASSCF, MRPT2/CASPT2, MP2 and QCISD(T) methods.
- Analytical non-adiabatic coupling matrix elements for MCSCF.
- Valence-Bond analysis of CASSCF wavefunction, and energy-optimized valence bond wavefunctions as described in Int. J. Quant. Chem. **65**, 439 (1997).
- One-electron transition properties for MCSCF, MRCI, and EOM-CCSD wavefunctions, CASSCF and MRCI transition properties also between wavefunctions with different orbitals, as described in Mol. Phys. **105**, 1239, (2007).
- Spin-orbit coupling, as described in Mol. Phys., **98**, 1823 (2000). More recently, a new spin-orbit integral program for generally contracted basis sets has been implemented.
- Douglas-Kroll-Hess Hamiltonian up to arbitrary order.
- Density-functional theory symmetry-adapted intermolecular perturbation theory (with density fitting), DFT-SAPT , as described in J. Chem. Phys. **122**, 014103 (2005).
- Some two-electron transition properties for MCSCF wavefunctions (e.g., L_x^2 etc.).
- Mulliken population analysis and Natural Population Analysis (NPA)
- Orbital localization.
- Natural bond orbitals (NBOs).
- Distributed Multipole Analysis (A. J. Stone).
- Automatic geometry optimization as described in J. Comp. Chem. **18**, (1997), 1473. Constrained optimization is also possible.
- Automatic calculation of vibrational frequencies, intensities, and thermodynamic properties.
- Reaction path following, as described in Theor. Chem. Acc. **100**, (1998), 21.
- Efficient facilities to treat large lattices of point charges for QM/MM calculations, including lattice gradients.
- Various utilities allowing other more general optimizations, looping and branching (e.g., for automatic generation of complete potential energy surfaces), general housekeeping operations.

- Geometry output in XYZ, MOLDEN and Gaussian formats; molecular orbital and frequency output in MOLDEN format.

- Integral-direct implementation of all Hartree-Fock, DFT and pair-correlated methods (MP, CCSD, MRCI etc.), as described in Mol. Phys., **96**, (1999), 719. At present, perturbative triple excitation methods are not implemented.

- Local second-order Møller-Plesset perturbation theory (LMP2) and local coupled cluster methods, as described in in J. Chem. Phys. **104**, 6286 (1996), Chem. Phys. Lett. **290**, 143 (1998), J. Chem. Phys. **111**, 5691 (1999), J. Chem. Phys. **113**, 9443 (2000), J. Chem. Phys. **113**, 9986 (2000), Chem. Phys. Letters **318**, 370 (2000), J. Chem. Phys. **114**, 661 (2001), Phys. Chem. Chem. Phys. **4**, 3941 (2002), J. Chem. Phys. **116**, 8772 (2002).

- Local density fitting methods, as described in J. Chem. Phys. **118**, 8149 (2003), Phys. Chem. Chem. Phys. **5**, 3349 (2003), Mol. Phys. **102**, 2311 (2004).

- Analytical energy gradients for LMP2, DF-LMP2, and LQCISD as described in J. Chem. Phys. **108**, 5185, (1998), Phy. Chem. Chem. Phys. **3**, 4853 (2001), J. Chem. Phys. **121**, 737 (2004).

- Explicitly correlated MP2-F12 and CCSD(T)-F12 methods, as described in J. Chem. Phys. **119**, 4607 (2003), J. Chem. Phys. **121**, 4479 (2004), J. Chem. Phys. **124**, 054114 (2006), J. Chem. Phys. **124**, 094103 (2006), J. Chem. Phys. **127**, 221106 (2007), J. Chem. Phys. **130**, 054104 (2009).

- Explicitly correlated local methods, as described in J. Chem. Phys. **129**, 101103 (2009), J. Chem. Phys. **130**, 054106 (2009), J. Chem. Phys. **130**, 241101 (2009).

- Parallel execution on distributed memory machines, as described in J. Comp. Chem. **19**, (1998), 1215. At present, SCF, DFT, MRCI, MP2, LMP2, CCSD(T), LCCSD(T) energies and SCF, DFT gradients are parallelized. Most density fitted codes such as DF-HF, DF-KS, DF-LMP2, DF-LMP2 gradients, DF-LCCSD(T), DF-MP2-F12, DF-DFT-SAPT, and GIAO-DF-HF NMR shieldings are also parallelized.

- Automatic *embarrassingly parallel* computation of numerical gradients and Hessians (`mppx Version`).

The program is written mostly in standard Fortran–90. Those parts which are machine dependent are maintained through the use of a supplied preprocessor, which allows easy interconversion between versions for different machines. Each release of the program is ported and tested on a number of systems. A fuller description of the hardware and operating systems of these machines can be found at `http://www.molpro.net/supported`. A large library of commonly used orbital basis sets is available, which can be extended as required. There is a comprehensive users' manual, which includes installation instructions. The manual is available in PDF and also in HTML for mounting on a Worldwide Web server.

More recent methods and enhancements include:

1. Explicitly correlated MP2-F12 (closed-shell) and RMP2-F12 (open-shell) methods with many many different ansätze, as described in *H.-J. Werner, T. B. Adler, and F. R. Manby*, J. Chem. Phys. **126**, 164102 (2007) and *G. Knizia and H.-J. Werner*, J. Chem. Phys. **128**, 154103 (2008).

2. Explicitly correlated CCSD(T)-F12 methods as described in *T. B. Adler, G. Knizia, and H.-J. Werner*, J. Chem. Phys. **127**, 221106 (2007) (closed-shell) and *G. Knizia, T. B. Adler, and H.-J. Werner*, J. Chem. Phys. **130**, 054104 (2009) (open-shell).

3. Explicitly correlated LMP2-F12 and LCCSD-F12 methods as described in *H.-J. Werner*, J. Chem. Phys. **129**, 101103 (2009), *T. B. Adler, H.-J. Werner*, and F. R. Manby, J. Chem. Phys. **130**, 054106 (2009), and *T. B. Adler and H.-J. Werner*, J. Chem. Phys. **130**, 241101 (2009).

4. Natural localized orbitals, NPA and NBO analysis and improved methods for domain selection in local correlation calculations as described in *R. A. Mata and H.-J. Werner*, Mol. Phys. **105**, 2753 (2007); see also *R. A. Mata and H.-J. Werner*, J. Chem. Phys. **125**, 184110 (2006).

5. Correlation regions in local correlation calculations as described in *R. A. Mata, H.-J. Werner and M. Schütz*, J. Chem. Phys. **128**, 144106 (2008).

6. Automated calculation of anharmonic vibrational frequencies and zero-point energies using VCI methods as described in *T. Hrenar, H.-J. Werner, and G. Rauhut*, J. Chem. Phys. 126, 134108 (2007) and references therein.

7. Dynamical state weighting as described in *M. P. Deskevich and D. J. Nesbitt, and H.-J. Werner*, J. Chem. Phys. **120**, 7281 (2004).

8. Coupling of DFT and coupled cluster methods as described in *E. Goll, T. Leininger, F. R. Manby, A. Mitrushchenkov, H.-J. Werner, and H. Stoll*, Phys. Chem. Chem. Phys. 10, 3353 (2008) and references therein.

9. NMR chemical shifts using London atomic orbitals for density-fitted Hartree-Fock, as described in *S. Loibl, F.R. Manby, M. Schütz*, Mol. Phys. *108*, 1362 (2010).

10. Local response methods (LCC2) for computing excitation energies and transition properties in large molecule as described in *D. Kats, T. Korona, M. Schütz*, J. Chem. Phys. *125*, 104106 (2006), *D. Kats, T. Korona, M. Schütz*, J. Chem. Phys. *127*, 064107 (2007), and *D. Kats, M. Schütz*, J. Chem. Phys. *131*, 124117 (2009).

11. Automatic basis set extrapolation.

12. Enhanced connections to other programs, including graphical display of output and 3-dimensional structures.

13. Support for Mac OS X

Future enhancements presently under development include

- Explicitly correlated local coupled cluster methods (closed and open-shell)
- Explicitly correlated CASPT2-F12 and MRCI-F12 methods.
- Analytical energy gradients for CCSD(T) and CCSD(T)-F12.
- Analytic second derivatives for DFT.
- New, more efficient MRCI methods for larger molecules.
- NMR chemical shifts using London atomic orbitals for local MP2.

These features will be included in the base version at later stages. The above list is for information only, and no representation is made that any of the above will be available within any particular time.

MOLPRO on the WWW

The latest information on MOLPRO, including program updates, can be found on the worldwide web at location `http://www.molpro.net/`.

References

All publications resulting from use of this program must acknowledge the following.

MOLPRO, version 2010.1, a package of *ab initio* programs, H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, P. Celani, T. Korona, R. Lindh, A. Mitrushenkov, G. Rauhut, K. R. Shamasundar, T. B. Adler, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, E. Goll, C. Hampel, A. Hesselmann, G. Hetzer, T. Hrenar, G. Jansen, C. Köppl, Y. Liu, A. W. Lloyd, R. A. Mata, A. J. May, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklass, D. P. O'Neill, P. Palmieri, K. Pflüger, R. Pitzer, M. Reiher, T. Shiozaki, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson, M. Wang, and A. Wolf, , see http://www.molpro.net.

Some journals insist on a shorter list of authors; in such a case, the following should be used instead.

MOLPRO, version 2010.1, a package of *ab initio* programs, H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, and others , see http://www.molpro.net.

Depending on which programs are used, the following references should be cited.

Integral evaluation (`SEWARD`)
R. Lindh, U. Ryu, and B. Liu, J. Chem. Phys. **95**, 5889 (1991).

Integral-direct Implementation
M. Schütz, R. Lindh, and H.-J. Werner, Mol. Phys. **96**, 719 (1999).

`MCSCF/CASSCF`**:**
H.-J. Werner and P. J. Knowles, J. Chem. Phys. **82**, 5053 (1985);
P. J. Knowles and H.-J. Werner, Chem. Phys. Lett. **115**, 259 (1985).

See also:

H.-J. Werner and W. Meyer, J. Chem. Phys. **73**, 2342 (1980);
H.-J. Werner and W. Meyer, J. Chem. Phys. **74**, 5794 (1981);
H.-J. Werner, Adv. Chem. Phys. **LXIX**, 1 (1987).

Internally contracted MRCI:
H.-J. Werner and P.J. Knowles, J. Chem. Phys. **89**, 5803 (1988);
P.J. Knowles and H.-J. Werner, Chem. Phys. Lett. **145**, 514 (1988).

See also:

H.-J. Werner and E.A. Reinsch, J. Chem. Phys. **76**, 3144 (1982);
H.-J. Werner, Adv. Chem. Phys. **LXIX**, 1 (1987).

Excited states with internally contracted MRCI:
P. J. Knowles and H.-J. Werner, Theor. Chim. Acta **84**, 95 (1992).

Internally contracted MR-ACPF, QDVPT, etc:
H.-J. Werner and P. J. Knowles, Theor. Chim Acta **78**, 175 (1990).

The original reference to uncontracted MR-ACPF, QDVPT, MR-AQCC are:
R. J. Gdanitz and R. Ahlrichs, Chem. Phys. Lett. **143**, 413 (1988);
R. J. Cave and E. R. Davidson, J. Chem. Phys. **89**, 6798 (1988);
P. G. Szalay and R. J. Bartlett, Chem. Phys. Lett. **214**, 481 (1993).

Multireference perturbation theory (CASPT2/CASPT3):
H.-J. Werner, Mol. Phys. **89**, 645 (1996);
P. Celani and H.-J. Werner, J. Chem. Phys. **112**, 5546 (2000).

Coupling of multi-reference configuration interaction and multi-reference perturbation theory, P. Celani, H. Stoll, H.-J. Werner and P. J. Knowles, Mol. Phys. **102**, 2369 (2004).

Analytical energy gradients and geometry optimization
Gradient integral evaluation (`ALASKA`): R. Lindh, Theor. Chim. Acta **85**, 423 (1993);
MCSCF gradients: T. Busch, A. Degli Esposti, and H.-J. Werner, J. Chem. Phys. **94**, 6708 (1991);
MP2 and LMP2 gradients: A. El Azhary, G. Rauhut, P. Pulay, and H.-J. Werner, J. Chem. Phys. **108**, 5185 (1998);
DF-LMP2 gradients: M. Schütz, H.-J. Werner, R. Lindh and F. R. Manby, J. Chem. Phys. **121**, 737 (2004).
QCISD and LQCISD gradients: G. Rauhut and H.-J. Werner, Phys. Chem. Chem. Phys. **3**, 4853 (2001);
CASPT2 gradients: P. Celani and H.-J. Werner, J. Chem. Phys. **119**, 5044 (2003).
Geometry optimization: F. Eckert, P. Pulay and H.-J. Werner, J. Comp. Chemistry **18**, 1473 (1997);
Reaction path following: F. Eckert and H.-J. Werner, Theor. Chem. Acc. **100**, 21, 1998.

Harmonic frequencies
G. Rauhut, A. El Azhary, F. Eckert, U. Schumann, and H.-J. Werner, Spectrochimica Acta **55**, 651 (1999).
T. Hrenar, G. Rauhut, H.-J. Werner, J. Phys. Chem. A **110**, 2060 (2006).

Anharmonic frequencies (SURF, VSCF, VCI):
G. Rauhut, J. Chem. Phys. **121**, 9313 (2004);
T. Hrenar, H.-J. Werner, G. Rauhut, J. Chem. Phys. **126**, 134108 (2007);
G. Rauhut, T. Hrenar, Chem. Phys. **346**, 160 (2008);
M. Neff, G. Rauhut, J. Chem. Phys. **131**, 124129 (2009).

Møller-Plesset Perturbation theory (MP2, MP3, MP4):
Closed-shell Møller-Plesset Perturbation theory up to fourth order [MP4(SDTQ)] is part of the coupled cluster code, see CCSD.

Open-shell Møller-Plesset Perturbation theory (RMP2):
R. D. Amos, J. S. Andrews, N. C. Handy, and P. J. Knowles, Chem. Phys. Lett. **185**, 256 (1991).

Coupled-Cluster treatments (QCI, CCSD, BCCD):
C. Hampel, K. Peterson, and H.-J. Werner, Chem. Phys. Lett. **190**, 1 (1992) and references therein. The program to compute the perturbative triples corrections has been developed by M. J. O. Deegan and P. J. Knowles, Chem. Phys. Lett. **227**, 321 (1994).

Equation-of-Motion Coupled Cluster Singles and Doubles (EOM-CCSD):
T. Korona and H.-J. Werner, J. Chem. Phys. **118**, 3006 (2003).

Open-shell coupled-cluster (RCCSD, UCCSD):
P. J. Knowles, C. Hampel and H.-J. Werner, J. Chem. Phys. **99**, 5219 (1993); Erratum: J. Chem. Phys. **112**, 3106 (2000).

Local MP2 (LMP2):
G. Hetzer, P. Pulay, and H.-J. Werner, Chem. Phys. Lett. **290**, 143 (1998)
M. Schütz, G. Hetzer, and H.-J. Werner, J. Chem. Phys. **111**, 5691 (1999)
G. Hetzer, M. Schütz, H. Stoll, and H.-J. Werner, J. Chem. Phys. **113**, 9443 (2000)
See also references on energy gradients and density fitting.

Local Coupled Cluster methods (LCCSD, LQCISD, LMP4):
C. Hampel and H.-J. Werner, J. Chem. Phys. **104** 6286 (1996)
M. Schütz and H.-J. Werner, J. Chem. Phys. **114**, 661 (2001)
M. Schütz, Phys. Chem. Chem.Phys. 4, 3941 (2002)
See also references on energy gradients and density fitting.

Local triple excitations:
M. Schütz and H.-J. Werner, Chem. Phys. Lett. **318**, 370 (2000);
M. Schütz, J. Chem. Phys. **113**, 9986 (2000).
M. Schütz, J. Chem. Phys. 116, 8772 (2002).

Density fitting methods:

DFT, Poisson fitting:	F. R. Manby, P. J. Knowles, and A. W. Lloyd, J. Chem. Phys. **115**, 9144 (2001).
DF-MP2, DF-LMP2:	H.-J. Werner, F. R. Manby, and P. J. Knowles, J. Chem. Phys. **118**, 8149 (2003).
DF-LCCSD:	M. Schütz and F. R. Manby, Phys. Chem. Chem. Phys. **5**, 3349 (2003)
DF-HF:	R. Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, Mol. Phys. **102**, 2311 (2004).
DF-LMP2 gradients:	M. Schütz, H.-J. Werner, R. Lindh and F. R. Manby, J. Chem. Phys. **121**, 737 (2004).
DF-LCCSD(T):	H.-J. Werner and M. Schütz, in prepation.

Explicitly correlated methods with density fitting:

DF-MP2-R12:	F. R. Manby, J. Chem. Phys. **119**, 4807 (2003).
DF-MP2-F12/2A:	A. J. May and F. R. Manby, J. Chem. Phys. **121**, 4479 (2004).
DF-MP2-F12:	H.-J. Werner, T. B. Adler, and F. R. Manby, J. Chem. Phys. **126**, 164102 (2007).
DF-RMP2-F12:	G. Knizia and H.-J. Werner, J. Chem. Phys. **128**, 154103 (2008).
DF-LMP2-F12:	H.-J. Werner and F. R. Manby, J. Chem. Phys., **124**, 054114 (2006); F. R. Manby H.-J. Werner, T. B. Adler, and A. J. May, J. Chem. Phys. **124**, 094103 (2006); H.-J. Werner, T. B. Adler, and F. R. Manby, J. Chem. Phys. **126**, 164102 (2007); T. B. Adler, H.-J. Werner, and F. R. Manby, J. Chem. Phys. **130**, 054106 (2009).
CCSD(T)-F12:	T. B. Adler, G. Knizia, and H.-J. Werner, J. Chem. Phys. **127**, 221106 (2007).
UCCSD(T)-F12:	G. Knizia, T. B. Adler, and H.-J. Werner, J. Chem. Phys. **130**, 054104 (2009).

Rangehybrid methods:
T. Leininger, H. Stoll, H.-J. Werner, and A. Savin, Chem. Phys. Lett. **275**, 151 (1997).
J.G. Ángyán, I.C. Gerber, A. Savin, and J. Toulouse, Phys. Rev. A **72**, 012510 (2005).
E. Goll, H.-J. Werner, and H. Stoll, Phys. Chem. Chem. Phys. **7**, 3917 (2005).

Full CI (FCI):
P. J. Knowles and N. C. Handy, Chem. Phys. Letters **111**, 315 (1984);
P. J. Knowles and N. C. Handy, Comp. Phys. Commun. **54**, 75 (1989).

Distributed Multipole Analysis (DMA):
A. J. Stone, Chem. Phys. Letters **83**, 233 (1981).

Valence bond:
D. L. Cooper, T. Thorsteinsson, and J. Gerratt, Int. J. Quant. Chem. **65**, 439 (1997);
D. L. Cooper, T. Thorsteinsson, and J. Gerratt, Adv. Quant. Chem. **32**, 51-67 (1998).
See also "An overview of the CASVB approach to modern valence bond calculations",
T. Thorsteinsson and D. L. Cooper, in *Quantum Systems in Chemistry and Physics. Volume 1: Basic problems and models systems*, eds. A. Hernndez-Laguna, J. Maruani, R. McWeeny, and S. Wilson (Kluwer, Dordrecht, 2000); pp 303-26.

Relativistic corrections using the Douglas-Kroll Hamiltonian:
M. Reiher, A. Wolf, JCP **121**, 2037–2047 (2004);
M. Reiher, A. Wolf, JCP **121**, 10945–10956 (2004);
A. Wolf, M. Reiher, B. A. Hess, JCP **117**, 9215–9226 (2002).

Spin-orbit coupling:
A. Berning, M. Schweizer, H.-J. Werner, P. J. Knowles, and P. Palmieri, Mol. Phys., **98**, 1823 (2000).

Diabatization procedures:
H.-J. Werner and W. Meyer, J. Chem. Phys. **74**, 5802 (1981);
H.-J. Werner, B. Follmeg, and M. H. Alexander, J. Chem. Phys. **89**, 3139 (1988);
D. Simah, B. Hartke, and H.-J. Werner, J. Chem. Phys. **111**, 4523 (1999).

DF-DFT-SAPT:
A. Heßelmann, G. Jansen and M. Schütz, J. Chem. Phys. **122**, 014103 (2005).

NMR shielding tensors:
S. Loibl, F.R. Manby, M. Schütz, Mol. Phys. **108**, 1362 (2010).

Contents

1 HOW TO READ THIS MANUAL

This manual is organized as follows: The next chapter gives an overview of the general structure of MOLPRO. It is essential for the new user to read this chapter, in order to understand the conventions used to define the symmetry, records and files, orbital spaces and so on. The later chapters, which describe the input of the individual program modules in detail, assume that you are familiar with these concepts. The appendices describe details of running the program, and the installation procedure.

Throughout this manual, words in `Typewriter Font` denote keywords recognized by MOLPRO. In the input, these have to be typed as shown, but may be in upper or lower case. Numbers or options which must be supplied by the user are in *italic*. In some cases, various different forms of an input record are possible. This is indicated as [*options*], and the possible options are described individually in subsequent subsections.

2 RUNNING MOLPRO

On Unix systems, MOLPRO is accessed using the *molpro* unix command. The syntax is

`molpro` [*options*] [*datafile*]

MOLPRO's execution is controlled by user-prepared data; if *datafile* is not given on the command line, the data is read from standard input, and program results go to standard output. Otherwise, data is taken from *datafile,* and the output is written to a file whose name is generated from *datafile* by removing any trailing suffix, and appending `.out`. If the output file already exists, then the old file is appended to the same name with suffix `.out_1`, and then deleted. This provides a mechanism for saving old output files from overwriting. Note that the above behaviour can be modified with the `-o` or `-s` options.

2.1 Options

Most options are not required, since sensible system defaults are usually set. Options as detailed below may be given, in order of decreasing priority, on the command line, in the environment variable MOLPRO_OPTIONS, or in the files `./molpro.rc`, `$HOME/.molprorc`, and `tuning.rc` in the library files directory.

`-d` *dir1:dir2:...* where *dir1:dir2:...* is a list of directories which may be used for creating scratch files. Each of the directories should be writable by those who will use the program, and the directory specification may contain embedded environment variables in shell form, for example `$TMPDIR` or `/tmp/$USER`; these will be expanded at run time. If multiple scratch file systems are available, it is advantageous to present a list of directories of which there is one in each file system. Some parts of MOLPRO present extreme I/O demands, and it is therefore important to be careful in optimizing the provision and specification of scratch directories.

Note that in the building of `bin/molpro`, the environment variables `$TMPDIR`, `$TMPDIR2`, `$TMPDIR3`,... are used to construct the list of scratch directories for the `-d` option. Thus, these environment variables should at make time be filled with the names of directories on each available scratch file system (cf. section A.2.3).

`-o` | `--output` *outfile* specifies a different output file.

`-x` | `--executable` *executable* specifies an alternative MOLPRO executable file.

`-d` | `--directory` *directory1*:*directory2*... specifies a list of directories in which the program will place scratch files. For detailed discussion of optimal specification, see the installation guide.

`--backup` *nfile* enables the saving of previous output files, up to a maximum of *nfile*. If *nfile* is omitted, it defaults to infinity. The names of the backup files are constructed by appending _ and a sequence number to the output file name, and both regular and xml-format files are processed, together with any log file.

`-a` | `--append-backup` Previous output files are concatenated by appending, instead of being kept separate.

`--directory-backup` Backup files are stored in a single separate subdirectory, named *datafile*.`d`, with subdirectories `01`, `02`,.... `--directory-backup` and `--append-backup` are mutually exclusive, and switching one of them on will force the other to be switched off.

`--backup-directory` *dir* In the case of `--directory-backup`, use dir as the location of backup files instead of the default.

`-s` | `--nobackup` disables the mechanism whereby an existing output file is saved.

`-v` | `--verbose` causes the procedure to echo debugging information; `--noverbose` selects quiet operation (default).

`-k` *key* where *key* is the licence key. This is normally not necessary since the key should be installed globally when installing MOLPRO.

`-m` specifies the working memory to be assigned to the program, in 8-byte words. The memory may also be given in units of 1000 words by appending the letter `k` to the value, or in units of 1000000 with the key `m`, or 10^9 with `g`. `K, M, G` stand for 2^{10}, 2^{20} and 2^{30}.

`-I` | `--main-file-repository` *directory* specifies the directory where the permanent copy of any integral file (file 1) resides. This may be a pathname which is absolute or relative to the current directory (e.g., '`.`' would specify the current directory). Normally, the `-I` directory should be equal to the `-d` working directory to avoid copying of large integral files, since after completion of the job the file will be copied to the directory given after `-I`. On some main frames, the scratch directory is erased automatically after a job has terminated, and in such cases a different `-I` directory, e.g., `$HOME/int`, can be specified (environment variables will be expanded at run time). In view of the large integral file sizes, this should be used with care, however. Note that in parallel runs with more than 1 processor the integral file will never be copied, and cannot be restarted.

`-W` | `--wavefunction-file-repository` *directory* is similar to `--main-file-repository` except that it refers to the directory for the wavefunction files (2,3 and 4). This determines the destination of permanent wavefunction (dump) files used for storing information like orbitals or CI-vectors etc. These files are essential for restarting a job. As explained for the integral files above, permanent wavefunction files will be copied to *directory* after completion of the job. The default for *directory* is `$HOME/wfu`.

`-X |--xml-output` specifies that the output file will be a well-formed XML file suitable for automatic post-processing. Important data such as input, geometries, and results are tagged, and the bulk of the normal descriptive output is wrapped as XML comments. `--no-xml-output` switches off this behaviour and forces a plain-text output file to be produced.

`-L | --library` *directory* specifies the directory where the basis set library files (`LIBMOL*`) are found.

`-1 | --file-1-directory` *directory:directory:...* specifies the directory where the runtime file 1 will be placed, overriding `--directory` for this file only. `-2`, `-3`, `-4`, `-5`, `-6`, `-7`, `-8` and `-9` may be used similarly. Normally these options should not be given, since the program tries to use what is given in `-d` to optimally distribute the I/O.

`-t | --omp-num-threads` *n* Specify the number of OpenMP threads, as if the environment variable `OPENMP_NUM_THREADS` were set to *n*.

`--xml2txt` Convert Molpro XML output file to plain text. In this mode the input file should refer to a Molpro XML output file.

There are a number of other options, specific to parallel execution, which are summarized below and described in detail in the next section. All of the followng options are ignored when using serial MOLPRO.

`-n | --tasks` *tasks/tasks_per_node:smp_threads* *tasks* specifies the number of parallel processes to be set up.

`-N | --task-specification` *user1:node1:tasks1,user2:node2:tasks2...* *node1, node2* etc. specify the host names of the nodes on which to run.

`-G | --global-memory` *memory* Global Arrays shared memory.

`-S | --shared-file-implementation` *method* specifies the method by which the shared data are held in parallel.

`--multiple-helper-server` *nprocs_per_server* enables the multiple helper servers.

`--node-helper-server` specifies one helper server on every node.

`--single-helper-server` specifies only one single helper server for all processes.

`--no-helper-server` disables the helper server.

There are a number of other options for tuning and system parameters, but these do not usually concern the general user.

It is not usually necessary to specify any of these options as there are sensible defaults. Sometimes installation dependent options can be found in the system configuration file `molpro.rc` in the same directory as the MOLPROlibrary files.

2.2 Running MOLPRO on parallel computers

MOLPRO will run on distributed-memory multiprocessor systems, including workstation clusters, under the control of the Global Arrays parallel toolkit or the MPI-2 library. There are also some parts of the code that can take advantage of shared memory parallelism through the OpenMP protocol, although these are somewhat limited, and this facility is not at present recommended. It should be noted that there remain some parts of the code that are not, or only partly, parallelized, and therefore run with replicated work. Additionally, some of those parts which have been parallelized rely on fast inter-node communications, and can be very inefficient across ordinary networks. Therefore some caution and experimentation is needed to avoid waste of resources in a multiuser environment.

MOLPRO effects interprocess cooperation through the *ppidd* library, which, depending on how it was configured and built, draws on either the Global Arrays parallel toolkit or pure MPI. *ppidd* is described in Comp. Phys. Commun. 180, 2673-2679 (2009). Global Arrays handles distributed data objects using whatever one-sided remote memory access facilities are provided and supported. In the case of the MPI implementation, there is a choice of using either MPI-2 one-sided memory access, or devoting some of the processes to act as data 'helpers'. It is generally found that performance is significantly better, and competitive with Global Arrays, if at least one dedicated helper is used, and in some cases it is advisable to specify more. The scalable limit is to devote one core on each node in a typical multi-core cluster machine, but in most cases it is possible to manage with fewer, thereby making more cores available for computation. This aspect of configuration can be tuned through the `*-helper-server` options described below.

Molpro can be compiled in three different ways:

1. Serial execution only. In this case, no parallelism is possible at run time.
2. 'MPP': a number of copies of the program execute simultaneously a single task. For example, a single CCSD(T) calculation can run in parallel, with the work divided between the processors in order to achieve a reduced elapsed time.
3. 'MPPX': a number of copies of the program run in serial executing identical independent tasks. An example of this is the calculation of gradients and frequencies by finite difference: for the initial wavefunction calculation, the calculation is replicated on all processes, but thereafter each process works in serial on a different displaced geometry. At present, this is implemented only for numerical gradients and Hessians.

Which of these three modes is available is fixed at compilation time, and is reported in the job output. The options, described below, for selecting the number and location of processors are identical for MPP and MPPX.

2.2.1 Specifying parallel execution

The following additional options for the `molpro` command may be used to specify and control parallel execution.

`-n` | `--tasks` *tasks/tasks_per_node:smp_threads* *tasks* specifies the number of parallel processes to be set up, and defaults to 1. *tasks_per_node* sets the number of GA(or MPI-2) processes to run on each node, where appropriate. The default is installation dependent. In some environments (e.g., IBM running under Loadleveler; PBS batch job), the value given by `-n` is capped to the maximum allowed by the environment; in such circumstances it can be useful to give a very large number as the value for `-n` so that the control of the number of processes is by the batch job specification. *smp_threads* relates to the use of OpenMP shared-memory parallelism, and specifies the maximum number of OpenMP threads that will be opened, and defaults to 1. Any of these three components may be omitted, and appropriate combinations will allow GA(or MPI-2)-only, OpenMP-only, or mixed parallelism.

`-N` | `--task-specification` *user1:node1:tasks1,user2:node2:tasks2*... *node1, node2* etc. specify the host names of the nodes on which to run. On most parallel systems, node1 defaults to the local host name, and there is no default for node2 and higher. On Cray T3E and IBM SP systems, and on systems running under the PBS batch system, if -N is not specified, nodes are obtained from the system in the standard way. *tasks1, tasks2* etc. may be used to control the number of tasks on each node as a more flexible alternative to `-n` / *tasks_per_node*. If omitted, they are each set equal to `-n` / *tasks_per_node*. *user1, user2* etc. give the username under which processes are to be created. Most of these parameters may be omitted in favour of the usually sensible default values.

`-G` | `--global-memory` *memory* Some parts of the program make use of Global Arrays for holding and communicating temporary data structures. This option sets the amount of memory to allocate in total across all processors for such activities. This option is no longer activated.

`-S` | `--shared-file-implementation` *method* specifies the method by which the shared data are held in parallel. *method* can be *sf* or *ga*, and it is set automatically according to the properties of scratch directories by default. If *method* is manually set to *sf*, please ensure all the scratch directories are shared by all processes. Note that for GA version of MOLPRO, if *method* is set to *sf* manually or by default, the scratch directories can't be located in NFS when running `molpro` job on multiple nodes. The reason is that the SF facility in Global Arrays doesn't work well on multiple nodes with NFS. There is no such restriction for MPI-2 version of MOLPRO.

`--multiple-helper-server` *nprocs_per_server* enables the multiple helper servers, and *nprocs_per_server* sets how many processes own one helper server. For example, when total number of processes is specified as 32 and *nprocs_per_server* = 8, then every 8 processes(including helper server) will own one helper server, and there are 4 helper servers in total. For any unreasonable value of *nprocs_per_server* (i.e., any integer less than 2), it will be reset to a very large number automatically, and this will be equivalent to option `--single-helper-server`.

`--node-helper-server` specifies one helper server on every node if all the nodes are symmetric and have reasonable processes (i.e., every node has the same number of processes, and the number should be greater than 1), and this is the default behaviour. Otherwise, only one single helper server for all processes/nodes will be used, and this will be equivalent to option `--single-helper-server`

`--single-helper-server` specifies only one single helper server for all processes.

`--no-helper-server` disables the helper server.

`-t` | `--omp-num-threads` *n* Specify the number of OpenMP threads, as if the environment variable `OPENMP_NUM_THREADS` were set to *n*.

Note that options `--multiple-helper-server`, `--node-helper-server`, `--single-helper-server`, and `--no-helper-server` are only effective for MOLPRO built with MPI-2 library. In the cases of one or more helper servers enabled, one or more processes act as data helper servers, and the rest processes are used for computation. Even so, it is quite competitive in performance when it is run with a large number of processes. In the case of helper server disabled, all processes are used for computation; however, the performance may not be good because of the poor performance of some existing implementations of the MPI-2 standard for one-sided operations.

In addition, for MOLPRO built with GA library (TCGMSG_MPI or MPI over InfiniBand), GA data structures can't be too large (e.g., 2GB per node) when running `molpro` job on multiple nodes. In this case, setting environment variable ARMCI_DEFAULT_SHMMAX might be helpful. The number should be less than 2GB (e.g., to set 1600MB for ARMCI_DEFAULT_SHMMAX in bash: export ARMCI_DEFAULT_SHMMAX=1600). One can also use more computer nodes to run such jobs, thus allocated memory for GA data structures on each node becomes smaller. There is no such restriction for MPI-2 version of MOLPRO.

3 DEFINITION OF MOLPRO INPUT LANGUAGE

3.1 Input format

MOLPRO's execution is controlled by an input file. In general, each input record begins with a keyword, which may be followed by data or other keywords. Molpro input contains commands, directives, options and data. The commands and directives are sequentially executed in the order they are encountered. Furthermore, procedures can be defined anywhere in the input, which can include any number of commands and directives. They are only executed when called (which may be before or after the definition in the input file).

The input file can be written in free format. The following conversions take place:

, (comma)	move to next tab stop, i.e. this delimits input fields
; (semicolon)	end of record, i.e. a new record is started
! (exclamation mark)	ignore rest of input line (useful for comments)
`---` (three dashes)	end of file (rest of input is ignored)

Input may be given upper or lower case. The input processor converts all characters to upper case. All integers are appended with "." (only floating point numbers are read by the program).

Several logical input records can actually be typed on one line and separated by semicolons, i.e., a given input line may contain many actual commands (separated by semicolons), or just one, as you prefer. These basic command units (records) delimited by semicolons are also frequently referred to as *cards* throughout this manual.

Exception to these general rules are:

`***`	first data line always
`INCLUDE`	include other input file
`FILE`	definition of named files
`TEXT`	prints text
`TITLE`	defines a title for the run or a table
`CON`	specifies orbital configurations
`---`	last line of input

These commands always occupy a whole line. Using `INCLUDE` it is possible to open secondary input files. If an `INCLUDE` command is encountered, the new input file is opened and read until its end. Input is then continued after the include card in the first file. `INCLUDE`'s may be nested.

A MOLPRO input record (card) contains a number of input *fields*. Input fields may be up to 256 characters wide and contain either expressions or strings. The fields can be separated by commas or blanks. We recommend the general use of commas in order to avoid unexpected results.

Each line may start with a label. A label is separated from the body of the line by a colon (:). The colon is part of the label. The length of the label must not exceed 6 characters (including the colon) and the labels must be unique. Labels may be useful with `GOTO` commands. Example:

```
GOTO,START:
...
START: CCSD(T)
```

Here `START:` is a label, and `CCSD(T)` is a command.

Strings containing blanks can be entered using quotes. For instance, `'This is a string'` is interpreted as one string, but `This is a string` is a sequence of four strings in four subsequent fields. Strings in quotes are not converted to upper case.

Input lines may be concatenated using \ at the end of the line(s) to be continued. Any number of lines may be concatenated up to a total length of 1024 characters (only 500 characters are possible on older IBM systems).

Filenames may be up to 31 characters long, provided that long filenames are supported by the Unix system used. An exception are older CRAY systems, which allow only 8 characters for the names of binary MOLPRO files.

3.2 Commands

A command invokes a particular program. It may be followed by local input for this program, enclosed in curley brackets[1]

The general format is either

`COMMAND`, *options*

or

{ `COMMAND`,*options*
directives
data
}

Examples for commands are `HF, MP2, CCSD(T), MCSCF, MRCI`. Examples for directives are `OCC, CLOSED, WF, PRINT`. Directives can be in any order, unless otherwise noted. Data can follow certain directives. For the format of options, directives and data see subsections 3.3, 3.5, and 3.6, respectively.

In the following, such a sequence of input will be denoted a *command block*. Special command blocks are the geometry and basis blocks.

The options given on the command line may include any options relevant to the current program. For instance, in DF-LMP2-R12 this could be options for density fitting, local, explicit, and/or thresholds. Alternatively, options can be specified on individual directives like

`DFIT`,*options*
`LOCAL`,*options*
`EXPLICIT`,*options*
`THRESH`,*options*

In these cases, only the options belong to the corresponding directive are valid; thus, if an option for `EXPLICIT` would be specified, e.g., on the `DFIT` directive, an error would result. This error would be detected already in the input prechecking stage.

[1]Depending on the parameter `STRICTCHECK` in file `lib/variable.registry` the program may tolerate directives given after commands without curley brackets. The program checks for ambiguities in the input. A directive is considered ambiguous if a command or procedure with the same name is known, and the directive is not in a command block (i.e., no curley brackets are used). `STRICTCHECK=0`: The input checker tolerates ambiguous directives if they a are followed by a non ambiguous directive which is valid for the current command. `STRICTCHECK=1`: The input checker does not tolerate any ambiguous directives. `STRICTCHECK=2`: The input checker does not tolerate any directives outside curley brackets. The default is `STRICTCHECK=0`, which gives the maximum possible compatibility to previous Molpro versions.

As already mentioned, the use of curly brackets is normally compulsary if more than one input line is needed. In the case of one-line commands, curley brackets are needed as well if the next command or procedure has the same name as a directive valid for the current command.

Note: `DIRECT` and associated options cannot be specified on command lines any more.

3.3 Directives

Directives serve to specify input data and special options for programs. They start with a keyword, followed by data and/or options. The general format is

`DIRECTIVE`,*data*,*options*

The format of data and options is specified in the subsequent sections. Data must always be given before any options.

Examples for directives are

`OCC`,`CORE`, `CLOSED`, `WF`, `LOCAL`, `DFIT`, ...

3.4 Global directives

Certain directives can be given anywhere in the input, i.e. either inside or outside command blocks. If they are given inside of command blocks, the specified options are valid only locally for the current program. However, if they are given outside a command block, they act globally, and are used for all programs executed after the input has been encountered. Local options have preference over global options.

The following directives can be either local or global:

Wavefunction definition: `OCC`,`CORE`, `CLOSED`, `FROZEN`, `WF`
Thresholds and options: `LOCAL`, `DFIT`, `DIRECT`, `EXPLICIT`, `THRESH`, `PRINT`, `GRID`

If such options are given outside a command block, a context can be speciffied

`DIRECTIVE`,*data*,`CONTEXT=`*context*,

e.g.,

`OCC`,3,1,1,`CONTEXT=HF`
`OCC`,4,1,2,`CONTEXT=MCSCF`

`CONTEXT` can be any valid command name (or any valid alias for this), but internally these are converted to one of the following: `HF` (Hartree-Fock and DFT), `MC` (MCSCF and CASSCF), `CC` (single reference correlation methods, as implemented in the CCSD program), `CI` (multireference correlation methods, as implemented in the MRCI program). The directive will then be applied to one of the four cases. Several contexts can be specified separated by colon, e.g.,

`CONTEXT=HF:CCSD`

If only a single context is given (no colon), shortcuts for the specifying the `CONTEXT` option are obtained by postfixing *context* to the command name, e.g.,

`OCC_HF`,3,1,1
`OCC_MCSCF`,4,1,2

If no context is given, the default is `HF`. The default occupations for single reference methods (e.g., MP2, CCSD) are the ones used in `HF`, the defaults for multireference methods (e.g. RS2, MRCI) correspond to those used in `MCSCF`.

3.5 Options

Options have the general form NAME[=*value*]

where value can be a number, and expression, or a string. Several options are separated by comma or blank. NAME must begin with a character [A-Z]. If options are given on a COMMAND line or on directives within a command block, they are valid only for the corresponding program (see Sec. 3.3). If options are given in a procedure, they are valid only in the procedure and procedures called from the current procedure; whenever a procedure is terminated, the options of the previous level are restored.

Options can also be single keywords, like SYM or NOSYM. In this case, the option is switched on or off, depending whether or not the key begins with NO. Alternatively, such logical options can also be set or unset using NAME=ON or NAME=OFF. For instance, SYM=OFF is equivalent to NOSYM. Furthermore, YES and NO are aliases for ON and OFF, respectively.

3.6 Data

Data are defined as a sequence of numbers, expressions, or strings, separated by commas or blanks. Generally, the order of data is essential. Empty fields are interpreted as zeros. Strings and variables must begin with a character [A-Z]. If $+$ or $-$ follows blank and directly precedes a number or variable it is interpreted as sign and not a binary operator. If there are no blanks before and after such operators, or blanks follow them, they are interpreted as binary operators. Examples:

`3 - 4 4`	yields [-1,4]
`3-4 4`	yields [-1,4]
`3 -4 4`	yields [3,-4,4]
`3,-4 4`	yields [3,-4,4]
`3, -4, 4`	yields [3,-4,4]

Expressions (including numbers) may contain variables.

Examples for the use of data: geometry and basis input, LATTICE, OCC, CLOSED, CORE, WF directives.

In some cases several lines of data are needed for a certain command or directive; in such cases the data must follow directly the corresponding command|directive, and must be enclosed in square brackets:

COMMAND,options
[*data*]

Normally, the input format of data is MOLPRO style, i.e., numbers are separated by commas, and variables as well as expressions can be used.

If data are included using external files, the input format of *data* is free format: no commas are needed, but no variables and expressions can be used.

3.7 Expressions

In any input field, data can be entered in the form of expressions. Numbers and variables are special cases of expressions. An expression is typed in Fortran style and may contain any number of nested parenthesis. The standard intrinsic functions are also available (see next section).

MOLPRO understands both arithmetic and logical expressions. The result of an arithmetic expression is a *real* (double precision) number. Internally, all integers are also converted to *real* numbers. The result of a logical expression is either `.TRUE.` or `.FALSE.`. Internally, `.TRUE.` is stored as a one (1.0), and `.FALSE.` as zero (0.0). Expressions may contain any number of variables.

The following standard operations can be performed :

expr + *expr*	Addition
expr − *expr*	Subtraction
expr * *expr*	Multiplication
expr / *expr*	Division
expr `.OR.` *expr*	Logical `OR`
expr `.AND.` *expr*	Logical `AND`
expr `.XOR.` *expr*	Exclusive `OR`
`.NOT.`*expr*	Logical `NOT`
expr `.GT.` *expr*	Greater Than
expr `.EQ.` *expr*	Equal
expr `.LT.` *expr*	Less Than
expr `.GE.` *expr*	Greater Equal
expr `.LE.` *expr*	Less Equal
expr `.NE.` *expr*	Not Equal
expr `**`*expr*	Exponentiation
expr ^ *expr*	Exponentiation
(*expr*)	Parenthesis (no effect)
−*expr*	Change sign
+*expr*	Keep sign (no effect)

3.8 Intrinsic functions

Expressions may contain the following intrinsic functions:

`ABS(`*expr*`)`	Absolute value
`MAX(`*expr,expr*`, ...)`	Largest value of arbitrary number of numbers or expressions
`MIN(`*expr,expr*`, ...)`	Smallest value of arbitrary number of numbers of expressions
`EXP(`*expr*`)`	Exponential
`LOG(`*expr*`)`	Natural Logarithm
`LOG10(`*expr*`)`	Common Logarithm
`SQRT(`*expr*`)`	Square Root
`NINT(`*expr*`)`	Next nearest integer
`INT(`*expr*`)`	Truncate to integer
`SIN(`*expr*`)`	Sine

`COS(`*expr*`)`	Cosine
`TAN(`*expr*`)`	Tangent
`ASIN(`*expr*`)`	Arcsine
`ACOS(`*expr*`)`	Arccosine
`ATAN(`*expr*`)`	Arctangent
`COSH(`*expr*`)`	Hyperbolic cosine
`SINH(`*expr*`)`	Hyperbolic sine
`TANH(`*expr*`)`	Hyperbolic tangent
`MOD(`*expr1* , *expr2* `)`	Remainder: `expr1-INT(expr1/expr2)*expr2`

Note: all trigonometric functions use or produce angles in degrees.

3.9 Variables

3.9.1 Setting variables

Data and results can be stored in MOLPRO variables. Variables can be of type string, floating, or logical and may be used anywhere in the input.

The syntax for setting variables is

`VARNAME1`=*expression* [unit],`VARNAME2`=*expression* [unit]

where unit is optional. If a variable is undefined, zero is assumed.

Variables are useful for running the same input with different actual parameters (e.g. geometries or basis function exponents), and to store and manipulate the results. *Arrays* are variables with an index in parenthesis, e.g., `var(1)`. The number of elements in an array `var` is `#var`. The array length can be reset to zero by the `CLEAR` directive or simply by modifying `#var`. Variables and variable arrays may be displayed at any place in the output by the `SHOW` command, and whole tables of variables can be generated using the `TABLE` command. For more details about variables see section 8.

3.9.2 String variables

Special care is necessary when using strings. In order to avoid unexpected results, either a $ has to be prefixed whenever a string variable is set, or the string has to be given in quotes. Possible forms are

`$name`=string
`name`='string'
`name`=*string variable*
`$name`=*string variable*

Examples:

```
string1='This is a string'  !define a string variable. Text in quotes is not
                            ! converted to upper case.
string2=string1             !assign string variable string1 to a new variable
```

```
 $string3=string1            !equivalent to previous case.
 $string4=mystring           !define a string variable. Since ''mystring'' is
                             !given in quotes, !it will be converted to upper
 string5=mystring            !string5 will not be a string variable since $ i
```

yields

```
 SETTING STRING1        =    This is a string
 SETTING STRING2        =    This is a string
 SETTING STRING3        =    This is a string
 SETTING STRING4        =    MYSTRING
 VARIABLE MYSTRING UNDEFINED, ASSUMING 0
 SETTING STRING5        =         0.00000000
```

For more information concerning strings and string variables, see section 8.3

3.10 Procedures

3.10.1 Procedure definition

Procedures are sequences of commands and/or options. They can be defined anywhere in the input as

[`PROC`]*procname*={
command blocks
directives
}

or

`PROC` *procname*
command blocks
directives
`ENDPROC`

In order to avoid unexpected results, *procname* must differ from all known command names. Procedures must not contain geometry blocks.

Note that procedures are not executed when encountered in the input, but only when called. Procedure definitions must not be nested. Procedures can contain procedure calls up to a nesting level of 10.

3.10.2 Procedure calls

Procedures can be called anywhere in the input. The syntax is the same as for commands (cf. section 3.2), except that the procedure name replaces the command name.

`PROCEDURE`

No options are allowed on procedure calls. However, specific options may be set using directives within the procedure, and these are then valid for all programs within the procedure which follow the directive. When execution of the procedure is finished, the previous global options are restored. The hierarchy in which options are processed is as follows:

Global options
Options in procedures
Command line options
Options given on directives within a command block

The last option set is then actually used. Thus, options specified on command lines or within command blocks have preference over procedure options, and procedure options have preference over global options.

4 GENERAL PROGRAM STRUCTURE

This chapter gives an overview of the most important features of MOLPRO. For the new user, it is essential to understand the strategies and conventions described in this section, in particular the meaning of *files* and *records*, and the use of *symmetry*. This chapter will focus on general aspects; detailed information about each command will be given in later chapters. Information about commands and parameters can also be obtained using the MOLPRO help facility (see section 4.13).

4.1 Input structure

A typical MOLPRO input has the following structure:

```
***,title                      !title (optional)
memory,4,m                     !memory specification (optional)
file,1,name.int                !permanent named integral file (optional)
file,2,name.wfu                !permanent named wavefunction file (optional)

gprint,options                 !global print options (optional)
gthresh,options                !global thresholds (optional)
gdirect[,options]              !global direct (optional)
gexpec,opnames                 !global definition of one-electron operators (opt

basis=basisname                !basis specification. If not present, cc-pVDZ is
geometry={...}                 !geometry specification

var1=value,var2=value,...      !setting variables for geometry and/or wavefuncti

{command,options               !program or procedure name
 directive,data,option         !directives for command (optional)
...
}                              !end of command block
---                            !end of input (optional)
```

If the memory card is given, it should be the first card (after the optional title card). If any file cards are given, they should follow immediately. The order of `basis`, `geometry`, `gprint`, `gdirect`, `gthresh`, `gexpec`, and variable definitions is arbitrary. It is possible to call several programs one after each other. It is also possible to redefine basis set and/or geometry between the call to programs; the program will recognize automatically if the integrals have to be recomputed.

4.2 Files

MOLPRO uses three sequential text files, namely the *input file*, the *output file*, and the *punch file*. The punch file is a short form of the output which contains the most important data and results, such as geometries, energies, dipole moments, etc. The punch file can be processed by the separate program *READPUN*, which selects specific results by keywords and is able to produce ordered tables in user supplied format. Furthermore, there are up to 9 binary MOLPRO files available, each one known to the program simply by its number (1 to 9). By default, they are temporary files, usually allocated dynamically by the program, but they can be connected to permanent files with the `FILE` command. Each file is direct access, and word addressable (word=64 bit usually), but is organised in *records* of any length. The name, address and length of each record is held in a directory at the start of the file.

File 1 is the *main file*, holding basis set, geometry, and the one and two electron integrals. By default, file 2 is the *dump file* and used to store the wavefunction information, i.e. orbitals, CI coefficients, and density matrices. File 3 is an auxiliary file which can be used in addition to file 2 for restart purposes. Often files 1 and 2 (and 3) are declared as permanent files (see `FILE`) to enable restarts. Storing the wavefunction information on file 2 is useful, since the integral file is overwritten at each new geometry, while the orbitals and CI coefficients of one calculation can be used as a starting guess for the next calculation at a neighbouring geometry. Files 4 to 8 are used as scratch space, e.g., for sorting the integrals, storage of transformed integrals and of the CI vectors. These files should normally not be made permanent.

Note that the file name appearing in molpro input is always converted to lower case on unix machines.

4.3 Records

Record names are positive integers, and are usually referred to in the format *record.file*, e.g., 2100.2 means the record called 2100 on file 2. Note that these names are quite arbitrary, and their numerical values have nothing to do with the order of the records in the file. Record names $\leq$ 2000 are reserved for standard quantities (e.g. integrals, properties etc.) and you should never use these in an input, unless you know exactly what you are doing. Some important default records to remember are

2100	RHF dump record (closed and open-shell)
2200	UHF dump record
2140	MCSCF dump record
4100	CPHF restart information
5000	MCSCF gradient information
5100	CP-MCSCF gradient information
5200	MP2 gradient information
5300	Hessian restart information
5400	Frequencies restart information
6300	Domain restart information

If an input contains several wavefunction calculations of the same type, e.g., several MCSCF calculations with different active spaces, the record number will be increased by 1 for each calculation of the same type. Thus, the results of the first SCF calculation in an input are stored in dump record 2100.2, the second SCF in record 2101.2, the first MCSCF in 2140.2, the second MCSCF in 2141.2 and so on. Note that these numbers refer to the occurrence in the input and not on the order in which the calculations are performed in the actual run. If an input or part of it is repeated using DO loops, this ensures that each calculation will start with the orbitals from the corresponding orbitals from the previous cycle, as long as the order of the commands in the input remains unchanged. If for instance the first SCF would be skipped in the second cycle using some IF / ENDIF structure, the second SCF would still use record 2101.2. Thus, under most circumstances the program defaults are appropriate, and the user does not have to specify the records.

After a restart this logic will still work correctly if the number and sequence of SCF and MCSCF commands is kept unchanged. Thus, if you want to skip certain parts of the input after a restart, it is recommended to use IF / ENDIF structures or the GOTO command rather than to delete or comment certain commands. If for some reason this is not possible, the START and ORBITAL directives can be used to specify explicitely the records to be used.

In general we recommend the use of program defaults whenever possible, since this minimizes the probability of input errors and frustration!

After completion of each program step, MOLPRO prints a summary of the records on each file.

4.4 Restart

Information from the permanent files is automatically recovered in subsequent calculations. This can be controlled using the RESTART directive.

4.5 Data set manipulation

It is possible to truncate files and rename or copy records using the DATA command. Several standard matrix operations can be performed with MATROP, e.g., printing records, linearly combining or multiplying matrices, or forming the trace of a product of two matrices.

4.6 Memory allocation

MOLPRO allocates memory dynamically as required by the user on the MEMORY card. Thus it is not necessary to maintain different versions of the program with different memory sizes. If the MEMORY command is omitted, the program will use a default memory size, which depends on the hardware used and how the program was installed. Note that, on Unix machines, the default memory can be set on the molpro command line using the flag -m.

4.7 Multiple passes through the input

It is possible to perform loops over parts of the input using DO loops, very much as in FORTRAN programs. DO loops may be nested to any reasonable depth. This can be conveniently used, for instance, to compute automatically whole potential energy surfaces.

4.8 Symmetry

MOLPRO can use Abelian point group symmetry only. For molecules with degenerate symmetry, an Abelian subgroup must be used — e.g., C_{2v} or D_{2h} for linear molecules. The symmetry

Table 1: The symmetry generators for the point groups

Generators	Point group
(null card)	C_1 (i.e. no point group symmetry)
`X` (or `Y` or `Z`)	C_s
`XY`	C_2
`XYZ`	C_i
`X,Y`	C_{2v}
`XY,Z`	C_{2h}
`XZ,YZ`	D_2
`X,Y,Z`	D_{2h}

Table 2: Numbering of the irreducible representations in D_{2h}

	D_{2h}	
No.	Name	Function
1	A_g	s
2	B_{3u}	x
3	B_{2u}	y
4	B_{1g}	xy
5	B_{1u}	z
6	B_{2g}	xz
7	B_{3g}	yz
8	A_u	xyz

group which is used is defined in the integral input by combinations of the symmetry elements x, y, and z, which specify which coordinate axes change sign under the corresponding generating symmetry operation. It is usually wise to choose z to be the unique axis where appropriate (essential for C_2 and C_{2h}). The possibilities in this case are shown in Table 1.

Normally, MOLPRO determines the symmetry automatically, and rotates and translates the molecule accordingly. However, explicit symmetry specification is sometimes useful to fix the orientation of the molecule or to use lower symmetries.

The irreducible representations of each group are numbered 1 to 8. Their ordering is important and given in Tables 2 – 4. Also shown in the tables are the transformation properties of products of x, y, and z. s stands for an isotropic function, e.g., s orbital, and for these groups, this gives also the transformation properties of x^2, y^2, and z^2. Orbitals or basis functions are generally referred to in the format *number.irrep*, i.e. 3.2 means the third orbital in the second irreducible representation of the point group used.

Table 3: Numbering of the irreducible representations in the four-dimensional groups

	C_{2v}		C_{2h}		D_2	
No.	Name	Function	Name	Function	Name	Function
1	A_1	s,z	A_g	s,xy	A	s
2	B_1	x,xz	A_u	z	B_3	x,yz
3	B_2	y,yz	B_u	x,y	B_2	y,xz
4	A_2	xy	B_g	xz,yz	B_1	xy

Table 4: Numbering of the irreducible representations in the two-dimensional groups

	C_s		C_2		C_i	
No.	Name	Function	Name	Function	Name	Function
1	A'	s,x,y,xy	A	s,z,xy	A_g	s,xy,xz,yz
2	A''	z,xz,yz	B	x,y,xz,yz	A_u	x,y,z

4.9 Defining the wavefunction

In all program modules where such information is required, the total symmetry of the N-electron wavefunction is defined on `WF` (wavefunction) cards in the following way:

`WF`,*nelec*,*irrep*,*spin*

or, alternatively

`WF`,[`NELEC`=*nelec*],[`SYM[METRY]`=*irrep*],[`spin`=*spin*],[`CHARGE`=*charge*]

where *nelec* is the total number of electrons, *irrep* is the number of the irreducible representation, and *spin* equals $2 \times S$, where S is the total spin quantum number. Instead of *nelec* also *charge* can be given, which specifies the total charge of the molecule. For instance, for a calculation in C_{2v} symmetry with 10 electrons, `WF,10,3,0` denotes a $^1B_2$ state, and `WF,10,1,2` a $^3A_1$ state. The charge can also be defined by setting the variable `CHARGE`:

`SET,CHARGE`=*charge*

This charge will be used in all energy calculations following this input. Not that `SET` is required, since `CHARGE` is a system variable (cf. section 8.4).

Although in principle each program unit requires a `WF` command, in practice it is seldom necessary to give it. The program remembers the information on the `WF` card, and so one might typically specify the information in an SCF calculation, but then not in subsequent MCSCF or CI calculations; this also applies across restarts. Furthermore, *nelec* defaults to the sum of the nuclear charges, *irrep* to 1 and *spin* to 0 or 1; thus in many cases, it is not necessary to specify a `WF` card at all.

If the `WF` directive is given outside an command input block, it is treated as global, i.e., the given values are used for all subsequent calculations. Setting the variables `NELEC`, `SPIN`, or `SYMMETRY`, has the same effect giving these on a gobal `WF` directive. If the global `WF` directive is given after the variable definition, the values of the variables are replaced by the values given on the `WF` directive. Vice versa, if a variable definition follows a gobal `WF` directive, the new value of the variable is used in the following. Note that `WF` input cards in command blocks have preference over global `WF` directives or input variables.

4.10 Defining orbital subspaces

In the SCF, MCSCF, and CI programs it may be necessary to specify how many orbitals in each symmetry are *occupied* (or *internal* in CI), and which of these are *core* or *closed shell* (doubly occupied in all CSFs). This information is provided on the `OCC`, `CORE`, and `CLOSED` cards in the following way:

`OCC`,$m_1, m_2, \ldots, m_8$; `CORE`,$co_1, co_2, \ldots, co_8$; `CLOSED`,$cl_1, cl_2, \ldots, cl_8$; `FROZEN`,$fr_1, fr_2, \ldots, fr_8$;

where m_i is the number of occupied orbitals (including core/frozen and closed), co_i the number of core orbitals, and cl_i is the number of closed-shell orbitals (including the core orbitals) in the irreducible representation i. In general, $m_i \geq cl_i$, and $cl_i \geq co_i$. It is assumed that these numbers refer to the first orbitals in each irrep. `FROZEN` only exists in the MCSCF program and denotes frozen core orbitals that are not optimized (note that in older MOLPRO versions frozen core orbitals were denoted `CORE`).

Note that the `OCC` and `CLOSED` cards have slightly different meanings in the SCF, MCSCF and CI or CCSD programs. In SCF and MCSCF, *occupied* orbitals are those which occur in any of the CSFs. In electron correlation methods (CI, MPn, CCSD etc), however, `OCC` denotes the orbitals which are occupied in any of the *reference* CSFs. In the MCSCF, `FROZEN` orbitals are doubly occupied in all CSFs and *frozen* (not optimized), while *closed* denotes all doubly occupied orbitals (frozen plus optimized). In the CI and CCSD programs, *core* orbitals are those which are not correlated and *closed* orbitals are those which are doubly occupied in all reference CSFs.

`OCC`, `CORE` and `CLOSED` directives are generally required in each program module where they are relevant; however, the program remembers the most recently used values, and so the directives may be omitted if the orbital spaces are not to be changed from their previous values. Note that this information is also preserved across restarts. Note also, as with the `WF` information, sensible defaults are assumed for these orbital spaces. For full details, see the appropriate program description.

The orbital spaces may also be defined outside command blocks, and then the directive is treated as global, i.e., it is used in all subsequent programs. Spaces specific to certain wavefunction types can be defined by specifiying the program name with a `CONTEXT` option, e.g.,

```
OCC,4,2,1,CONTEXT=MULTI
```

Alternatively, the context can be appended to the directive name with an underscore. For example

```
OCC_MULTI,4,2,1
```

is equivalent to the previous form.

Local input given within command blocks has preference over global input.

4.11 Selecting orbitals and density matrices (`ORBITAL`, `DENSITY`)

As outlined in section 4.3, the information for each SCF or MCSCF calculation is stored in a *dump record*. Dump records contain orbitals, density matrices, orbital energies, occupation numbers, fock matrices and other information as wavefunction symmetries etc. Subsequent calculation can access the orbitals and density matrices from a particular record using the `ORBITAL` and `DENSITY` directives, respectively. These input cards have the same structure in all programs. The general format of the `ORBITAL` and `DENSITY` directives is as follows.

`ORBITAL[,[RECORD=]`*record*`]` `[,[TYPE=]`*orbtype*`]` `[,STATE=`*state*`]` `[,SYM[METRY]=`*symmetry*`]` `[,SPIN=`*spin*`]` `[,MS2=`*ms2*`]` `[,[N]ELEC=`*nelec*`]` `[,SET=`*iset*`]` `[,OVL]` `[,NOCHECK]` `[,IGNORE[_ERROR]]`

`DENSITY[,[RECORD=]`*record*`]` `[,[TYPE=]`*dentype*`]` `[,STATE=`*state*`]` `[,SYM[METRY]=`*symmetry*`]` `[,SPIN=`*spin*`]` `[,MS2=`*ms2*`]` `[,[N]ELEC=`*nelec*`]` `[,SET=`*iset*`]`

where the (optional) specifications can be used to select specific orbitals, if several different orbital sets are stored in the same record. The meaning of the individual specifications are as follows:

orbtype Orbital type. This can be one of
`CANONICAL`: canonical or pseudo-canonical orbitals;
`NATURAL`: natural orbitals;
`LOCAL`: localized orbitals;
`LOCAL(PM)`: localized Pipek-Mezey orbitals;
`LOCAL(BOYS)`: localized Boys orbitals;
`PROJECTED`: projected virtual orbitals used in local calculations.
Without further specification, the most recently computed orbitals of the specified type are used. If the orbital type is not specified, the program will try to find the most suitable orbitals automatically. For instance, in MRCI calculations `NATURAL` orbitals will be used preferentially if available; MRPT (CASPT2) calculations will first search for `CANONICAL` orbitals, and local calculations will first look for `LOCAL` orbitals. Therefore, in most cases the orbital type needs not to be specified.

state Specifies a particular state in the form *istate.isym*. For instance, 2.1 refers to the second state in symmetry 1. This can be used if density matrices or natural orbitals have been computed for different states in a state-averaged CASSCF calculation. If not given, the state-averaged orbitals are used. The specification of *isym* is optional; it can also be defined using the `SYMMETRY` key.

dentype Density type. This can be one of
`CHARGE`: charge density;
`SPIN`: UHF spin density;
`TRANSITION`: transition density matrix;
The default is `CHARGE`.

symmetry Specifies a particular state symmetry. Alternatively, the state symmetry can be specified using `STATE` (see above).

spin Spin quantum number, i.e. 0 for singlet, 1/2 for doublet, 1 for triplet, etc. Alternatively `MS2` can be used.

ms2 $2M_S$, i.e. 0 for singlet, 1 for doublet, 2 for triplet etc. Alternatively, `SPIN` can be used.

nelec Number of electrons.

iset Set number of orbitals. The orbital sets are numbered in the order they are stored.

If OVL is specified, the starting orbitals are obtained by maximizing the overlap with previous orbitals. By default, this is used if the basis dimension of the previous orbitals is different then the current one. If OVL is specified this procedure is used even if the basis dimensions are the same, which is occasionally useful if the contraction scheme changed.

If NOCHECK is specified, some consistency checks for finding correct orbitals are skipped, and error messages like "ORBITALS CORRESPOND TO DIFFERENT GEOMETRY" are ignored.

If IGNORE_ERROR is specified, MPn or triples calculations can be forced with other than canonical orbitals. Note that this can lead to meaningless results! Note that in MULTI IGNORE_ERROR must be given on the START directive, since in this program ORBITAL is used to define the new orbitals.

If any of the above options are given, they must be obeyed strictly, i.e., the program aborts if the request cannot be fulfilled.

Examples:

```
ORBITAL,2100.2                    !Use SCF orbitals
ORBITAL,2140.2                    !Use (state-averaged) MCSCF orbitals
ORBITAL,2140.2,CANONICAL          !use canonical MCSCF orbitals
ORBITAL,2140.2,NATURAL,STATE=2.1  !use natural MCSCF orbitals for second state
```

4.12 Summary of keywords known to the controlling program

This is a summary of all keywords presently implemented in the controlling program. Each module knows further keywords, which are described in the chapters about the individual programs. For detailed information about the use of the commands listed below, consult the following chapters.

Program control:

`***`	indicates start of a new calculation
`MEMORY`	allocates dynamic memory
`PUNCH`	opens a punch file
`FILE`	connects units to permanent files
`RESTART`	recovers file information
`INCLUDE`	includes other input files
`BASIS`	can be used to define default basis sets
`GEOMETRY`	can be used to specify the geometry
`ZMAT`	can be used to define the Z-matrix
`PARALLEL`	can be used to control parallelization
`STATUS`	checks status of program steps
`PRINT`,`GPRINT`	controls global print levels
`THRESH`,`GTHRESH`	controls global thresholds
`DIRECT`,`GDIRECT`	flags direct computation of integrals and for setting direct options
`EXPEC`,`GEXPEC`	controls computation of expectation values
`TEXT`	prints text
`EXIT`	stops execution
`DO`	controls do loops
`ENDDO`	end of do loops
`IF`	controls conditional actions
`ELSEIF`	controls conditional actions
`ENDIF`	end of `IF block`
`GOTO`	used to skip part of input and for loops over input
`LABEL`	no action
`DATA`	data set management
`DELETE`, `ERASE`	data set deletion
`MATROP`	performs matrix operations
`GRID`	Define grid
`CUBE`	Dump data to grid
`CARTESIAN`	Use cartesian basis functions
`SPHERICAL`	Use spherical harmonic basis functions
`USER`	calls user-supplied subroutine
`---`	last line of input

Variables:

`SET`	sets variables (obsolete)
`SETI`	sets variables or numbers to their inverse (obsolete)
`SETA`	sets variable arrays (obsolete)
`CLEAR`	clears variables

`CLEARALL`	clears all variables
`GETVAR`	recovers variables from file
`SHOW`	displays the values of variables
`TABLE`	prints tables

Wave function optimization:

`INT`	calls the machine default integral program. This is optional and needs not to be given.
`LSINT`	calls the spin-orbit integral program
`SORT`	calls two-electron sorting program. This is called automatically and needs not to be given
`CPP`	compute core polarization potential integrals
`HF`, `RHF`, `HF-SCF`, or `RHF-SCF`	calls spin-restricted Hartree-Fock program (open or closed shell)
`UHF` or `UHF-SCF`	calls spin-unrestricted Hartree-Fock program
`DFT`	calls the density functional program
`KS`, `RKS`	call the Kohn-Sham spin restricted density functional program
`UKS`	call the Kohn-Sham spin-unrestricted density functional program
`MULTI`, `MCSCF`, or `CASSCF`	calls MCSCF/CASSCF program
`CASVB`	calls the CASVB valence bond program
`CI`, `MRCI`, or `CI-PRO`	calls internally contracted MRCI program
`CIPT2`	calls internally contracted CIPT2 program
`ACPF`, `AQCC`	calls internally contracted MR-ACPF program
`CEPA`	calls single-reference CEPA program (closed- or open-shell)
`RS2`, `RS3`	calls internally contracted multireference perturbation theory
`RS2C`	faster program for internally contracted multireference perturbation theory
`MP2`	calls closed-shell MP2 program
`MP3`	calls closed-shell MP3 program
`MP4`	calls closed-shell MP4 program
`CISD`	calls closed-shell CISD program
`CCSD`	calls closed-shell coupled cluster program
`BCCD`	calls closed-shell Brueckner CCD program
`QCI`,`QCSID`	calls closed-shell quadratic configuration interaction program
`UCCSD`	calls spin-unrestricted open-shell coupled cluster program
`RCCSD`	calls spin-restricted open-shell coupled cluster program
`FCI` or `FULLCI`	calls determinant based full CI program

Local correlation methods:

`LMP2`	calls closed-shell local MP2 program
`LMP3`	calls closed-shell local MP3 program
`LMP4`	calls closed-shell local MP4 program

`LCISD`	calls closed-shell local CISD program
`LCCSD`	calls closed-shell local coupled cluster program
Explicitly correlated methods:	
`DF-MP2-R12`	MP2-R12 program with density fitting
`DF-MP2-F12`	MP2-F12 program with density fitting
`DF-LMP2-R12`	Local MP2-R12 program with density fitting
`DF-LMP2-F12`	Local MP2-F12 program with density fitting
Orbital manipulation:	
`LOCALI`	calls orbital localization program
`MERGE`	calls orbital manipulation program
Properties and wavefunction analysis:	
`POP`	calls population analysis program
`DMA`	calls distributed multipole analysis program
`PROPERTY`	calls properties program
`DIP`	adds dipole field to *h*
`QUAD`	adds quadrupole field to *h*
`LATTICE`	read or disable lattice of point charges
Gradients and geometry optimization:	
`FORCES`	calls gradient program
`OPTG`	performs automatic geometry optimization
`MIN`	performs energy minimization with respect to some parameters
`PUT`	print or write geometry to a file
`HESSIAN`	calculate Hessian
`FREQUENCY`	calculate vibrational frequencies
`MASS`	define atomic masses
`DDR`	evaluates approximate non-adiabatic coupling matrix elements

The command names for single reference coupled cluster methods `QCISD, CCSD, LQCISD, LCCSD` can be appended by `(T)` and then a perturbative correction for triple excitations will be computed (e.g., `CCSD(T)`).

`HF`, `KS`, `MP2` and all local correlation methods can be prepended by `DF-` to invoke density fitting.

4.13 MOLPRO help

The help command can be used to obtain a short description of commands, input parameters, and variables. The syntax is:

`HELP`,*set*,*name*,[*keys*]

where *set* is either `COMMAND`, `VARIABLE`, or the name of the input set (e.g., `THRESH`, `PRINT`, `LOCAL`, `EOM`, `CFIT`), and *name* is the name of the parameter. If *name* is blank, all parameters of the set are shown. Optionally, *keys* can be specified to request specific information (e.g., short_description, long_description, default_value, type, program). If *keys* are not given, short_description is assumed.

Currently, help is only available for a limited number of parameters and commands. However, the database will be extended in the near future.

5 INTRODUCTORY EXAMPLES

This section explains some very simple calculations in order to help the new user to understand how easy things can be.

5.1 Using the molpro command

1. Perform a simple SCF calculation for molecular hydrogen. The input is typed in directly and the output is sent to the terminal:

```
molpro <<!
basis=vdz;
geometry={angstrom;h1;h2,h1,.74}
hf
!
```

2. The same calculation, with the data taken from the file h2.com. The output is sent to h2.out. On completion, the file h2.pun is returned to the current directory and the file h2.wf to the directory $HOME/wfu (this is the default):

```
molpro h2.com
```

h2.com contains:

```
***,H2
file,2,h2.wf,new;
punch,h2.pun;
basis=vdz;
geometry={angstrom;h1;h2,h1,.74}
hf
```

http://www.molpro.net/info/current/examples/h2.com

3. As before, but the file h2.wf is sent to the directory /tmp/wfu:

```
molpro -W /tmp/wfu h2.com
```

5.2 Simple SCF calculations

The first example does an SCF calculation for H_2O, using all possible defaults.

```
***,h2o                    !A title
r=1.85,theta=104           !set geometry parameters
geometry={O;               !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
hf                         !closed-shell scf
```

http://www.molpro.net/info/current/examples/h2o_scf.com

In the above example, the default basis set (VDZ) is used. We can modify the default basis using a BASIS directive.

```
***,h2o cc-pVTZ basis      !A title
r=1.85,theta=104           !set geometry parameters
geometry={O;               !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
basis=VTZ                  !use VTZ basis
hf                         !closed-shell scf
```

http://www.molpro.net/info/current/examples/h2o_scf_vtz.com

5.3 Geometry optimizations

Now we can also do a geometry optimization, simply by adding the card OPTG.

```
***,h2o                    !A title
r=1.85,theta=104           !set geometry parameters
geometry={O;               !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
basis=6-31g**              !use Pople basis set
hf                         !closed-shell scf
optg                       !do scf geometry optimization
```

http://www.molpro.net/info/current/examples/h2o_scfopt_631g.com

5.4 CCSD(T)

The following job does a CCSD(T) calculation using a larger (VTZ) basis (this includes an *f* function on oxygen and a *d* function on the hydrogens).

```
***,h2o                    !A title
r=1.85,theta=104           !set geometry parameters
geometry={O;               !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
basis=VTZ                  !use VTZ basis
hf                         !closed-shell scf
ccsd(t)                    !do ccsd(t) calculation
```

http://www.molpro.net/info/current/examples/h2o_ccsdt_vtz.com

5.5 CASSCF and MRCI

Perhaps you want to do a CASSCF and subsequent MRCI for comparison. The following uses the full valence active space in the CASSCF and MRCI reference function.

```
***,h2o                   !A title
r=1.85,theta=104          !set geometry parameters
geometry={o;              !z-matrix geometry input
          h1,O,r;
          h2,O,r,H1,theta}
basis=vtz                 !use VTZ basis
hf                        !closed-shell scf
ccsd(t)                   !do ccsd(t) calculation
casscf                    !do casscf calculation
mrci                      !do mrci calculation
```

http://www.molpro.net/info/current/examples/h2o_mrci_vtz.com

5.6 Tables

You may now want to print a summary of all results in a table. To do so, you must store the computed energies in variables:

```
***,h2o                   !A title
r=1.85,theta=104          !set geometry parameters
geometry={o;              !z-matrix geometry input
          h1,O,r;
          h2,O,r,H1,theta}
basis=vtz                 !use VTZ basis
hf                        !closed-shell scf
e(1)=energy               !save scf energy in variable e(1)
method(1)=program         !save the string 'HF' in variable method(1)

ccsd(t)                   !do ccsd(t) calculation
e(2)=energy               !save ccsd(t) energy in variable e(2)
method(2)=program         !save the string 'CCSD(T)' in variable method(2)

casscf                    !do casscf calculation
e(3)=energy               !save scf energy in variable e(3)
method(3)=program         !save the string 'CASSCF' in variable method(3)
mrci                      !do mrci calculation
e(4)=energy               !save scf energy in variable e(4)
method(4)=program         !save the string 'MRCI' in variable method(4)

table,method,e            !print a table with results
title,Results for H2O, basis=$basis  !title for the table
```

http://www.molpro.net/info/current/examples/h2o_table.com

This job produces the following table:

```
 Results for H2O, basis=VTZ

 METHOD         E
 HF        -76.05480122
 CCSD(T)   -76.33149220
 CASSCF    -76.11006259
 MRCI      -76.31960943
```

5.7 Procedures

You could simplify this job by defining a procedure SAVE_E as follows:

```
***,h2o                    !A title

proc save_e                !define procedure save_e
if(#i.eq.0) i=0            !initialize variable i if it does not exist
i=i+1                      !increment i
e(i)=energy                !save scf energy in variable e(i)
method(i)=program          !save the present method in variable method(i)
endproc                    !end of procedure


r=1.85,theta=104           !set geometry parameters
geometry={o;               !z-matrix geometry input
          h1,O,r;
          h2,O,r,H1,theta}
basis=vtz                  !use VTZ basis
hf                         !closed-shell scf
save_e                     !call procedure, save results

ccsd(t)                    !do ccsd(t) calculation
save_e                     !call procedure, save results

casscf                     !do casscf calculation
save_e                     !call procedure, save results

mrci                       !do mrci calculation
save_e                     !call procedure, save results

table,method,e             !print a table with results
title,Results for H2O, basis=$basis  !title for the table
```

http://www.molpro.net/info/current/examples/h2o_proce.com

The job produces the same table as before.

5.8 Do loops

Now you have the idea that one geometry is not enough. Why not compute the whole surface? DO loops make it easy. Here is an example, which computes a whole potential energy surface for H_2O.

```
***,H2O potential
symmetry,x                          !use cs symmetry
geometry={
         o;                         !z-matrix
         h1,o,r1(i);
         h2,o,r2(i),h1,theta(i) }
basis=vdz                            !define basis set
angles=[100,104,110]                 !list of angles
distances=[1.6,1.7,1.8,1.9,2.0]      !list of distances
i=0                                  !initialize a counter
do ith=1,#angles                     !loop over all angles H1-O-H2
do ir1=1,#distances                  !loop over distances for O-H1
do ir2=1,ir1                         !loop over O-H2 distances(r1.ge.r2)
i=i+1                                !increment counter
r1(i)=distances(ir1)                 !save r1 for this geometry
r2(i)=distances(ir2)                 !save r2 for this geometry
theta(i)=angles(ith)                 !save theta for this geometry
hf;                                  !do SCF calculation
escf(i)=energy                       !save scf energy for this geometry
ccsd(t);                             !do CCSD(T) calculation
eccsd(i)=energc                      !save CCSD energy
eccsdt(i)=energy                     !save CCSD(T) energy
enddo                                !end of do loop ith
enddo                                !end of do loop ir1
enddo                                !end of do loop ir2
{table,r1,r2,theta,escf,eccsd,eccsdt  !produce a table with results
head, r1,r2,theta,scf,ccsd,ccsd(t)   !modify column headers for table
save,h2o.tab                         !save the table in file h2o.tab
title,Results for H2O, basis $basis  !title for table
sort,3,1,2}                          !sort table
```

http://www.molpro.net/info/current/examples/h2o_pes_ccsdt.com

This produces the following table.

```
 Results for H2O, basis VDZ

    R1     R2    THETA       SCF            CCSD           CCSD(T)
   1.6    1.6    100.0   -75.99757338   -76.20140563   -76.20403920
   1.7    1.6    100.0   -76.00908379   -76.21474489   -76.21747582
   1.7    1.7    100.0   -76.02060127   -76.22812261   -76.23095473
   ...
   2.0    1.9    110.0   -76.01128923   -76.22745359   -76.23081968
   2.0    2.0    110.0   -76.00369171   -76.22185092   -76.22537212
```

You can use also use `DO` loops to repeat your input for different methods.

```
***,h2o benchmark
$method=[hf,fci,ci,cepa(0),cepa(1),cepa(2),cepa(3),mp2,mp3,mp4,\
      qci,ccsd,bccd,qci(t),ccsd(t),bccd(t),casscf,mrci,acpf]
basis=dz                                !Double zeta basis set
geometry={o;h1,o,r;h2,o,r,h1,theta}     !Z-matrix for geometry
r=1 ang, theta=104                      !Geometry parameters
do i=1,#method                          !Loop over all requested methods
$method(i);                             !call program
e(i)=energy                             !save energy for this method
enddo
escf=e(1)                               !scf energy
efci=e(2)                               !fci energy
table,method,e,e-escf,e-efci            !print a table with results
!Title for table:
title,Results for H2O, basis $basis, R=$r Ang, Theta=$theta degree
```

http://www.molpro.net/info/current/examples/h2o_manymethods.com

This calculation produces the following table.

```
 Results for H2O, basis DZ, R=1 Ang, Theta=104 degree

 METHOD        E              E-ESCF       E-EFCI
 HF         -75.99897339    .00000000    .13712077
 FCI        -76.13609416   -.13712077    .00000000
 CI         -76.12844693   -.12947355    .00764722
 CEPA(0)    -76.13490643   -.13593304    .00118773
 CEPA(1)    -76.13304720   -.13407381    .00304696
 CEPA(2)    -76.13431548   -.13534209    .00177868
 CEPA(3)    -76.13179688   -.13282349    .00429728
 MP2        -76.12767140   -.12869801    .00842276
 MP3        -76.12839400   -.12942062    .00770015
 MP4        -76.13487266   -.13589927    .00122149
 QCI        -76.13461684   -.13564345    .00147732
 CCSD       -76.13431854   -.13534515    .00177561
 BCCD       -76.13410586   -.13513247    .00198830
 QCI(T)     -76.13555640   -.13658301    .00053776
 CCSD(T)    -76.13546225   -.13648886    .00063191
 BCCD(T)    -76.13546100   -.13648762    .00063315
 CASSCF     -76.05876129   -.05978790    .07733286
 MRCI       -76.13311835   -.13414496    .00297580
 ACPF       -76.13463018   -.13565679    .00146398
```

One can do even more fancy things, like, for instance, using macros, stored as string variables. See example `oh_macros.com` for a demonstration.

6 PROGRAM CONTROL

6.1 Starting a job (***)

The first card of each input should be:

***,*text*

where *text* is arbitrary. If file 1 is restarted, *text* must always be the same. The effect of this card is to reset all program counters, etc. If the `***` card is omitted, *text* assumes its default value, which is all blank.

6.2 Ending a job (`---`)

The end of the input is signaled by either an end of file, or a

`---`

card. All input following the `---` card is ignored.

Alternatively, a job can be stopped at at some place by inserting an `EXIT` card. This could also be in the middle of a `DO` loop or an `IF` block. If in such a case the `---` card would be used, an error would result, since the `ENDDO` or `ENDIF` cards would not be found.

6.3 Restarting a job (`RESTART`)

In contrast to MOLPRO92 and older versions, the current version of MOLPRO attempts to recover all information from all permanent files by default. If a restart is unwanted, the `NEW` option can be used on the `FILE` directive. The `RESTART` directive as described below can still be used as in MOLPRO92, but is usually not needed.

`RESTART`,$r_1, r_2, r_3, r_4, \ldots$;

The r_i specify which files are restarted. These files must have been allocated before using `FILE` cards. There are two possible formats for the r_i:

a) $0 < r_i < 10$:	Restart file r_i and restore all information.
b) $r_i = name.nr$:	Restart file *nr* but truncate before record *name*.

If all $r_i = 0$, then all permanent files are restarted. However, if at least one r_i is not equal to zero, only the specified files are restarted.

Examples:

`RESTART;`	will restart all permanent files allocated with `FILE` cards (default)
`RESTART,1;`	will restart file 1 only
`RESTART,2;`	will restart file 2 only
`RESTART,1,2,3;`	will restart files 1-3
`RESTART,2000.1;`	will restart file 1 and truncate before record 2000.

6.4 Including secondary input files (`INCLUDE`)

`INCLUDE`,*file*[,`ECHO`];

Insert the contents of the specified *file* in the input stream. In most implementations the file name given is used directly in a Fortran open statement. If the `ECHO` option is specified, the included file is echoed to the output in the normal way, but by default its contents are not printed. The included file may itself contain `INCLUDE` commands up to a maximum nesting depth of 10.

6.5 Allocating dynamic memory (MEMORY)

MEMORY,*n,scale*;

Sets the limit on dynamic memory to *n* floating point words. If *scale* is given as K, *n* is multiplied by 1000; if *scale* is M, *n* is multiplied by 1 000 000.

Note: The MEMORY card must precede all FILE cards!

Examples:

MEMORY,90000	allocates 90 000 words of memory
MEMORY,500,K	allocates 500 000 words of memory
MEMORY,2,M	allocates 2 000 000 words of memory

6.6 DO loops (DO/ENDDO)

DO loops can be constructed using the DO and ENDDO commands. The general format of the DO command is similar to Fortran:

DO *variable=start, end* [[,]*increment*] [[,]*unit*]

where *start, end, increment* may be expressions or variables. The default for *increment* is 1. In contrast to Fortran, these variables can be modified within the loop (to be used with care!). For instance:

```
DR=0.2
DO R=1.0,6.0,DR,ANG
IF (R.EQ.2) DR=0.5
IF (R.EQ.3) DR=1.0
....
ENDDO
```

performs the loop for the following values of R: 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0 Ångstrøm. The same could be achieved as follows:

```
RVEC=[1.0,1.2,1.4,1.6,1.8,2.0,2.5,3.0,4.0,5.0,6.0] ANG
DO I=1,#RVEC
R=RVEC(I)
....
ENDDO
```

Up to 20 DO loops may be nested. Each DO must end with its own ENDDO.

Jumps into DO loops are possible if the DO variables are known. This can be useful in restarts, since it allows to continue an interrupted calculation without changing the input (all variables are recovered in a restart).

6.6.1 Examples for do loops

The first example shows how to compute a potential energy surface for water.

```
***,H2O potential
symmetry,x                           !use cs symmetry
geometry={
         o;                          !z-matrix
         h1,o,r1(i);
         h2,o,r2(i),h1,theta(i) }
basis=vdz                             !define basis set
angles=[100,104,110]                  !list of angles
distances=[1.6,1.7,1.8,1.9,2.0]       !list of distances
i=0                                   !initialize a counter
do ith=1,#angles                      !loop over all angles H1-O-H2
do ir1=1,#distances                   !loop over distances for O-H1
do ir2=1,ir1                          !loop over O-H2 distances(r1.ge.r2)
i=i+1                                 !increment counter
r1(i)=distances(ir1)                  !save r1 for this geometry
r2(i)=distances(ir2)                  !save r2 for this geometry
theta(i)=angles(ith)                  !save theta for this geometry
hf;                                   !do SCF calculation
escf(i)=energy                        !save scf energy for this geometry
ccsd(t);                              !do CCSD(T) calculation
eccsd(i)=energc                       !save CCSD energy
eccsdt(i)=energy                      !save CCSD(T) energy
enddo                                 !end of do loop ith
enddo                                 !end of do loop ir1
enddo                                 !end of do loop ir2
{table,r1,r2,theta,escf,eccsd,eccsdt   !produce a table with results
head, r1,r2,theta,scf,ccsd,ccsd(t)    !modify column headers for table
save,h2o.tab                          !save the table in file h2o.tab
title,Results for H2O, basis $basis   !title for table
sort,3,1,2}                           !sort table
```

http://www.molpro.net/info/current/examples/h2o_pes_ccsdt.com

The next example shows how to loop over many methods.

```
***,h2o benchmark
$method=[hf,fci,ci,cepa(0),cepa(1),cepa(2),cepa(3),mp2,mp3,mp4,\
      qci,ccsd,bccd,qci(t),ccsd(t),bccd(t),casscf,mrci,acpf]
basis=dz                                !Double zeta basis set
geometry={o;h1,o,r;h2,o,r,h1,theta}     !Z-matrix for geometry
r=1 ang, theta=104                      !Geometry parameters
do i=1,#method                          !Loop over all requested methods
$method(i);                             !call program
e(i)=energy                             !save energy for this method
enddo
escf=e(1)                               !scf energy
efci=e(2)                               !fci energy
table,method,e,e-escf,e-efci            !print a table with results
!Title for table:
title,Results for H2O, basis $basis, R=$r Ang, Theta=$theta degree
```

http://www.molpro.net/info/current/examples/h2o_manymethods.com

6.7 Branching (IF/ELSEIF/ENDIF)

IF blocks and IF/ELSEIF blocks can be constructed as in FORTRAN.

6.7.1 `IF` statements

`IF` blocks have the same form as in Fortran:

`IF (`*logical expression*`) THEN`
statements
`ENDIF`

If only one statement is needed, the one-line form

`IF (`*logical expression*`)` *statement*

can be used, except if *statement* is a procedure name.

`ELSE` and `ELSE IF` can be used exactly as in Fortran. `IF` statements may be arbitrarily nested. Jumps into `IF` or `ELSE IF` blocks are allowed. In this case no testing is performed; when an `ELSE` is reached, control continues after `ENDIF`.

The logical expression may involve logical comparisons of algebraic expressions or of strings. Examples:

```
IF(STATUS.LT.0) THEN
TEXT,An error occurred, calculation stopped
STOP
ENDIF
```

```
IF($method.eq.'HF') then
...
ENDIF
```

In the previous example the dollar and the quotes are optional:

```
IF(METHOD.EQ.HF) then
...
ENDIF
```

6.7.2 `GOTO` commands

`GOTO` commands can be used to skip over parts of the input. The general form is

`GOTO,`*command*`,[`*n*`],[`*nrep*`]`

Program control skips to the $|n|$'th occurrence of *command* (Default: $n = 1$). *command* must be a keyword in the first field of an input line. If n is positive, the search is forward starting from the current position. If n is negative, search starts from the top of the input. The `GOTO` command is executed at most *nrep* times. The default for *nrep* is 1 if $n < 0$ and infinity otherwise. We recommend that `GOTO` commands are never used to construct loops.

Alternatively, one can jump to labels using

`GOTO,`*label*

Since labels must be unique, the search starts always from the top of the input. It is required that the *label* ends with a colon.

6.7.3 Labels (LABEL)

LABEL

This is a dummy command, sometimes useful in conjunction with GOTO.

6.8 Procedures (PROC/ENDPROC)

Procedures can be defined at the top of the input or in INCLUDE files as follows:

PROC *name*
statements
ENDPROC

Alternatively, one can use the form

PROC *name*[=]{*statements*}

In the latter case, it is required that the left curly bracket ({) appears on the same line as PROC, but *statements* can consist of several lines. If in the subsequent input *name* is found as a command *in the first field* of a line, it is substituted by the *statements*. Example:

```
PROC SCF
IF (#SPIN.EQ.0.OR.MOD(SPIN,2).NE.MOD(NELEC,2)) SET,SPIN=MOD(NELEC,2)
IF (SPIN.EQ.0) THEN
  HF
ELSE
  RHF
ENDIF
ENDPROC
```

Alternatively, this could be written as

```
PROC SCF={
IF (#SPIN.EQ.0.OR.MOD(SPIN,2).NE.MOD(NELEC,2)) SET,SPIN=MOD(NELEC,2)
IF (SPIN.EQ.0) THEN; HF; ELSE; RHF; ENDIF}
```

Procedures may be nested up to a depth of 10. In the following example SCF is a procedure:

```
PROC CC
SCF
IF (SPIN.EQ.0) THEN
  CCSD
ELSE
  RCCSD
ENDPROC
```

Note: Procedure names are substituted only if found in the first field of an input line. Therefore, they must not be used on one-line IF statements; please use IF / ENDIF structures instead.

If as first statement of a procedure ECHO is specified, the substituted commands of the present and lower level procedures will be printed. If ECHO is specified in the main input file, all subsequent procedures are printed.

Certain important input data can be passed to the program using variables. For instance, occupancy patterns, symmetries, number of electrons, and multiplicity can be defined in this way (see section 8.8 for more details). This allows the quite general use of procedures. For example, assume the following procedure has been defined at the top of the input:

```
PROC MRCI
IF (INTDONE.EQ.0) INT
IF (SCFDONE.EQ.0) THEN
SCF
ENDIF
MULTI
CI
ENDPROC
```

This procedure can be used for a calculation of a vertical ionization potential of H_2O as follows:

```
R=1 ANG            !Set bond distance
THETA=104 DEGREE   !Set bond angle

BASIS=VTZ          !Define basis set

GEOMETRY           !Geometry input block
O                  !Z-matrix
H1,O,R
H2,O,R,H1,THETA
ENDG               !End of geometry input
HF
MRCI               !Compute mrci energy of water using defaults
EH2O=ENERGY        !save mrci energy in variable EH2O

SET,NELEC=9        !Set number of electrons to 9
SET,SYMMETRY=2     !Set wavefunction symmetry to 2
HF
MRCI               !Compute mrci energy of H2O+ (2B2 state)

IPCI=(ENERGY-EH2O)*TOEV  !Compute MRCI ionization potential in eV
```

Note: At present, all variables are *global*, i.e., variables are commonly known to all procedures and all variables defined in procedures will be subsequently known outside the procedures as well. The reason is that procedures are included into the internal input deck at the beginning of the job and not at execution time; for the same reason, variable substitution of procedure names is not possible, e.g. one cannot use constructs like

```
method=scf
$method       !this does not work!
```

6.9 Text cards (TEXT)

TEXT,*xxxxxx*

will just print *xxxxxx* in the output. If the text contains variables which are preceded by a dollar ($), these are replaced by their actual values, e.g.

```
r=2.1
text,Results for R=\$r
```

will print

```
Results for R=2.1
```

6.10 Checking the program status (STATUS)

STATUS,[ALL|LAST|*commands*],[IGNORE|STOP|CRASH],[CLEAR]

This command checks and prints the status of the specified program steps. *commands* may be a list of commands for wavefunction calculations previously executed in the current job. If no *command* or LAST is specified, the status of the last step is checked. If ALL is given, all program steps are checked.

If CRASH or STOP is given, the program will crash or stop, respectively, if the status was not o.k. (STOP is default). If IGNORE is given, any bad status is ignored. If CLEAR is specified, all status information for the checked program steps is erased, so there will be no crash at subsequent status checks.

Examples:

STATUS,HF,CRASH; will check the status of the last HF-SCF step and crash if it was not o.k. (i.e. no convergence). CRASH is useful to avoid that the next program in a chain is executed.

STATUS,MULTI,CI,STOP; will check the status of the most previous MULTI and CI steps and stop if something did not converge.

STATUS,RHF,CLEAR; will clear the status flag for last RHF. No action even if RHF did not converge.

Note that the status variables are not recovered in a restart.

By default, the program automatically does the following checks:

1.) If an orbital optimization did not converge, and the resulting orbitals are used in a subsequent correlation calculation, an error will result. This the error exit can be avoided using the IGNORE_ERROR option on the ORBITAL directive.

2.) If a CCSD|QCI|BCC|LMPn calculation did not converge, further program steps which depend on the solution (e.g, Triples, CPHF, EOM) will not be done and an error will result. This can be avoided using the NOCHECK option on the command line.

3.) In geometry optimizations or frequency calculations no convergence will lead to immediate error exits.

6.11 Global Thresholds (GTHRESH)

A number of global thresholds can be set using the GTHRESH command outside the individual programs (the first letter G is optional, but should be used to avoid confusion with program specific THRESH cards). The syntax is

GTHRESH,*key1=value1*,*key2=value2*,...

key can be one of the following.

`ZERO`	Numerical zero (default 1.d-12)
`ONEINT`	Threshold for one-electron integrals (default 1.d-12, but not used at present)
`TWOINT`	Threshold for the neglect of two-electron integrals (default 1.d-12)
`PREFAC`	Threshold for test of prefactor in `TWOINT` (default 1.d-14)
`LOCALI`	Threshold for orbital localization (default 1.d-8)
`EORDER`	Threshold for reordering of orbital after localization (default 1.d-4)
`ENERGY`	Convergence threshold for energy (default 1.d-6)
`GRADIENT`	Convergence threshold for orbital gradient in `MCSCF` (default 1.d-2)
`STEP`	Convergence threshold for step length in MCSCF orbital optimization (default 1.d-3)
`ORBITAL`	Convergence threshold for orbital optimization in the SCF program (default 1.d-5).
`CIVEC`	Convergence threshold for CI coefficients in `MCSCF` and reference vector in `CI` (default 1.-d.5)
`COEFF`	Convergence threshold for coefficients in `CI` and `CCSD` (default 1.d-4)
`PRINTCI`	Threshold for printing CI coefficients (default 0.05)
`PUNCHCI`	Threshold for punching CI coefficients (default 99 - no punch)
`SYMTOL`	Threshold for finding symmetry equivalent atoms (default 1.d-6)
`GRADTOL`	Threshold for symmetry in gradient (default 1.d-6).
`THROVL`	Threshold for smallest allowed eigenvalue of the overlap matrix (default 1.d-8)
`THRORTH`	Threshold for orthonormality check (default 1.d-8)

6.12 Global Print Options (`GPRINT/NOGPRINT`)

Global print options can be set using the `GPRINT` command outside the individual programs (the first letter `G` is optional, but should be used to avoid confusion with program specific `PRINT` cards). The syntax is

`GPRINT`,*key1*[=*value1*],*key2*[=*value2*],...
`NOGPRINT`,*key1*,*key2*,...

Normally, *value* can be omitted, but values > 0 may be used for debugging purposes, giving more information in some cases. The default is no print for all options, except for `DISTANCE`, `ANGLES` (default=0), and `VARIABLE`. `NOGPRINT`,*key* is equivalent to `PRINT`,*key*`=-1`. *key* can be one of the following:

`BASIS`	Print basis information
`DISTANCE`	Print bond distances (default)
`ANGLES`	Print bond angle information (default). If > 0, dihedral angles are also printed.
`ORBITAL`	Print orbitals in `SCF` and `MCSCF`
`ORBEN`	Print orbital energies in `SCF`

`CIVECTOR`	Print `CI` vector in `MCSCF`
`PAIRS`	Print pair list in `CI, CCSD`
`CS`	Print information for singles in `CI, CCSD`
`CP`	Print information for pairs in `CI, CCSD`
`REF`	Print reference CSFs and their coefficients in `CI`
`PSPACE`	Print p-space configurations
`MICRO`	Print micro-iterations in `MCSCF` and `CI`
`CPU`	Print detailed CPU information
`IO`	Print detailed I/O information
`VARIABLE`	Print variables each time they are set or changed (default).

6.13 One-electron operators and expectation values (`GEXPEC`)

The operators for which expectation values are requested, are specified by keywords on the global `GEXPEC` directive. The first letter `G` is optional, but should be used to avoid confusion with program specific `EXPEC` cards, which have the same form as `GEXPEC`. For all operators specified on the `GEXPEC` card, expectation values are computed in all subsequent programs (if applicable).

For a number of operators it is possible to use *generic* operator names, e.g., `DM` for dipole moments, which means that all three components `DMX`, `DMY`, and `DMZ` are computed. Alternatively, individual components may be requested.

The general format is as follows:

`[G]EXPEC`,*opname*[,][*icen*,[*x,y,z*]],...

where

opname	operator name (string), either generic or component.
icen	z-matrix row number or z-matrix symbol used to determine the origin (x,y,z must not be specified). If *icen*= 0 or blank, the origin must be specified in *x,y,z*

Several `GEXPEC` cards may follow each other, or several operators may be specified on one card.

Examples:

`GEXPEC,QM` computes quadrupole moments with origin at (0,0,0),

`GEXPEC,QM1` computes quadrupole moments with origin at centre 1.

`GEXPEC,QM,O1` computes quadrupole moments with origin at atom `O1`.

`GEXPEC,QM,,1,2,3` computes quadrupole moments with origin at (1,2,3).

The following table summarizes all available operators:

Expectation values are only nonzero for symmetric operators (parity=1). Other operators can be used to compute transition quantities (spin-orbit operators need a special treatment). By default, the dipole moments are computed.

6.13.1 Example for computing expectation values

The following job computes dipole and quadrupole moments for H_2O.

```
***,h2o properties
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
gexpec,dm,sm,qm                       !compute dipole and quarupole moments
$methods=[hf,multi,ci]                !do hf, casscf, mrci
do i=1,#methods                       !loop over methods
$methods(i)                           !run energy calculation
e(i)=energy
dip(i)=dmz                            !save dipole moment in variable dip
quadxx(i)=qmxx                        !save quadrupole momemts
quadyy(i)=qmyy
quadzz(i)=qmzz
smxx(i)=xx                            !save second momemts
smyy(i)=yy
smzz(i)=zz
enddo
table,methods,dip,smxx,smyy,smzz        !print table of first and second moments
table,methods,e,quadxx,quadyy,quadzz    !print table of quadrupole moments
```

http://www.molpro.net/info/current/examples/h2o_gexpec2.com

This Job produces the following tables

METHODS	DIP	SMXX	SMYY	SMZZ
HF	0.82747571	-5.30079792	-3.01408114	-4.20611391
MULTI	0.76285513	-5.29145148	-3.11711397	-4.25941000
CI	0.76868508	-5.32191822	-3.15540500	-4.28542917

METHODS	E	QUADXX	QUADYY	QUADZZ
HF	-76.02145798	-1.69070039	1.73937477	-0.04867438
MULTI	-76.07843443	-1.60318949	1.65831677	-0.05512728
CI	-76.23369821	-1.60150114	1.64826869	-0.04676756

6.13.2 Example for computing relativistic corrections

```
***,ar2
geometry={ar1;ar2,ar1,r}   !geometry definition
r=2.5 ang                  !bond distance
{hf;                       !non-relativisitic scf calculation
expec,rel,darwin,massv}    !compute relativistic correction using Cowan-Griffin operator
e_nrel=energy              !save non-relativistic energy in variable enrel
show,massv,darwin,erel     !show individual contribution and their sum

dkroll=1                   !use douglas-kroll one-electron integrals
hf;                        !relativistic scf calculation
e_dk=energy                !save relativistic scf energy in variable e_dk.
show,massv,darwin,erel     !show mass-velocity and darwin contributions and their sum
show,e_dk-e_nrel           !show relativistic correction using Douglas-Kroll
```

http://www.molpro.net/info/current/examples/ar2_rel.com

This jobs shows at the end the following variables:

```
MASSV / AU              =       -14.84964285
DARWIN / AU             =        11.25455679
EREL / AU               =        -3.59508606
```

Table 5: One-electron operators and their components

Generic name	Parity	Components	Description
OV	1		Overlap
EKIN	1		Kinetic energy
POT	1		potential energy
DELTA	1		delta function
DEL4	1		Δ^4
DARW	1		one-electron Darwin term, i.e., DELTA with appropriate factors summed over atoms.
MASSV	1		mass-velocity term, i.e., DEL4 with appropriate factor.
REL	1		total Cowan-Griffin Relativistic correction, i.e., DARW+MASSV.
DM	1	DMX, DMY, DMZ	dipole moments
SM	1	XX, YY, ZZ, XY, XZ, YZ	second moments
TM	1	XXX, XXY, XXZ, XYY, XYZ, XZZ, YYY, YYZ, YZZ, ZZZ	third moments
MLTP*n*	1	all unique Cartesian products of order *n*	multipole moments
QM	1	QMXX, QMYY, QMZZ, QMXY, QMXZ, QMYZ, QMRR=XX + YY + ZZ, QMXX=(3 XX - RR)/2, QMXY=3 XY / 2 etc.	quadrupole moments and R^2
EF	1	EFX, EFY, EFZ	electric field
FG	1	FGXX, FGYY, FGZZ, FGXY, FGXZ, FGYZ	electric field gradients
DMS	1	DMSXX, DMSYX, DMSZX, DMSXY, DMSYY, DMSZY, DMSXZ, DMSYZ, DMSZZ	diamagnetic shielding tensor
LOP	-1	LX, LY, LZ	Angular momentum operators $\hat{L}_x$, $\hat{L}_y$, $\hat{L}_z$
LOP2	1	LXLX, LYLY, LZLZ, LXLY, LXLZ, LYLZ The symmetric combinations	one electron parts of products of angular momentum operators. $\frac{1}{2}(\hat{L}_x\hat{L}_y+\hat{L}_y\hat{L}_x)$ etc. are computed
VELO	-1	D/DX, D/DY, D/DZ	velocity
LS	-1	LSX, LSY, LSZ	spin-orbit operators
ECPLS	-1	ECPLSX, ECPLSY, ECPLSZ	ECP spin-orbit operators

7 FILE HANDLING

7.1 FILE

The FILE directive is used to open permanent files, which can be used for later restarts. The syntax in MOLPRO94 and later versions is

FILE,*file*,*name*,[*status*]

file is the logical MOLPRO file number (1-9). *name* is the file name (will be converted to lower case). *status* can be one of the following:

UNKNOWN	A permanent file is opened. If it exists, it is automatically restarted. This is the default.
OLD	Same effect as UNKNOWN. No error occurs if the file does not exist.
NEW	A permanent file is opened. If it already exists, it is erased and not restarted.
ERASE	Same effect as NEW.
SCRATCH	A temporary file is opened. If it already exists, it is erased and not restarted. After the job has finished, the file is no longer existent.
DELETE	Same effect as SCRATCH.

Note that RESTART is now the default for all permanent files. All temporary files are usually allocated automatically where needed. I/O buffers are allocated at the top of the dynamic memory, and the available memory decreases by the size of the buffers. The MEMORY card must therefore be presented before the first FILE card!

Examples:

FILE,1,H2O.INT	allocates permanent file 1 with name H2O.INT. Previous information on the file is recovered.
FILE,2,H2O.WFU,NEW	allocates permanent file 2 with name H2O.WFU. All previous information on the file is erased.

Note that filenames are converted to lower case on unix machines.

7.2 DELETE

DELETE,*file1*, *file2*, ...

Deletes the specified files. *file* refers to the logical MOLPRO file numbers as specified on the FILE card.

7.3 ERASE

ERASE,*file1*, *file2*, ...

Erases the specified files. *file* refers to the logical MOLPRO file numbers as specified on the FILE card.

7.4 DATA

The DATA command can be used to modify the MOLPRO binary files.

UNIT	Alias for NPL (should never be used)
RENAME,*rec1*,*rec2*	used to rename *rec1* to *rec2*. *rec1* and *rec2* must be given in the form *name.ifil*, where *ifil* is the number of a MOLPRO binary file (alias for NAME).
TRUNCATE,*nen*	used to truncate files after *nen-1* records (alias for NEN).
TRUNCATE,*rec*	used to truncate before record *rec*. *rec* must be given in the form *name.ifil*, where *ifil* is the number of a MOLPRO binary file.
COUNT	Alias for NRE (presently not used)
COPY,*rec1*,*rec2*	Copies record *rec1* to *rec2*. *rec1* and *rec2* must be given in the form *nam1.ifil1*, *nam2.ifil2*. If *nam2=0*, *nam2=nam1*. If *nam1=0*, all records are copied from file *ifil1* to file *ifil2*.

7.5 Assigning punch files (PUNCH)

PUNCH,*filename*,[REWIND]

Opens punch file named *filename*. If this file already exists, it is appended, unless the REWIND or NEW option is specified; in that case, any previous information on the punch file is overwritten. See FILE for machine dependent interpretation of filename. The punch file contains all important results (geometries, energies, dipole, transition moments etc). It can be read by a separate program *READPUN*, which can produce tables in user supplied format.

Example:

PUNCH,H2O.PUN allocates punch file H2O.PUN

Note that the file name is converted to lower case on unix machines.

7.6 MOLPRO system parameters (GPARAM)

The GPARAM card allows to change MOLPRO system parameters. This should only be used by experts!

GPARAM,*option*=*value*,...

The following options can be given in any order.

NOBUFF	if present, disable system buffering
LSEG	disk sector length
INTREL	number of integer words per real word (should never be modified!)
IBANK	number of memory banks. Default is 2, which should always be o.k.
IVECT	0=scalar, 1=vector machine
MINVEC	minimum vector length for call to mxmb
LTRACK	page size in buffer routines (must be multiple of *lseg*)

LENBUF	length of integral buffer (file 1)
NTR	length of integral records (must be multiple of 3·*ltrack*)
LTR	disk sector length assumed in CI (default 1 is reasonable)
NCACHE	machine cache size in bytes
IASYN	if nonzero, use asynchronous I/O on CONVEX
MXMBLK	column/row block size for mxma
MXMBLN	link block size for mxma
NCPUS	maximum number of cpus to be used in multitasking
MINBR1	min number of floating point ops per processor
MXDMP	highest file number to be treated as dump file with full functionality ($1 \leq .$ MXDMP$\leq .3$).

The MXDMP option is for experts only! This prevents basis and geometry information from being written to dump files with higher file number than the given *value*, and can sometimes be useful for counterpoise corrected geometry optimizations. Note that some functionality is lost by giving this option, and errors will result unless all input is correct!

8 VARIABLES

Data may be stored in *variables*. A variable can be of type *string*, *real* or *logical*, depending on the type of the expression in its definition. Any sequence of characters which is not recognized as expression or variable is treated as string. In this section, we will discuss only *real* and *logical* variables. *String* variables will be discussed in more detail in section 8.3. Variables can be used anywhere in the input, but they can be set only outside the input blocks for specific programs. For example, if a variable is used within the input block for HF, it must have been set before the HF{...} input block.

MOLPRO automatically stores various results and data in system variables (see section 8.8.1), which can be used for further processing. A new feature of MOLPRO2002 is that most system variables are write protected and cannot be overwritten by the user. The input is automatically checked before the job starts, and should a system variable be set in the input the job will stop immediately with an error message. Only in some exceptions (see section 8.4), system variables can be modified using the SET command (but not with the simple NAME=*value* syntax). Note that due to the changed usage and syntax of the SET command, compatibility with MOLPRO92 input syntax is no longer maintained.

8.1 Setting variables

A variable can be defined using

variable1=*value1*, *variable2*=*value2*, ...

A variable definition is recognized by the equals sign in the *first* field of the input card. For example,

THRESH,ENERGY=1.d-8,GRADIENT=1.d-5

does *not* define variables; here ENERGY and GRADIENT are options for the THRESH directive.

Variables can have different types:

Numbers:	The value is a number or an expression. The general form of *value* is *expression* [,] [*unit*] *unit* is an optional string which can be used to associate a unit to the value. `ANG[STROM]`, `DEGREE`, `HARTREE` are examples. Undefined variables in expressions are assumed to be zero (and defined to be zero at the same time).
Logicals:	The value can be `.TRUE.` or `.FALSE.` (`.T.` and .F. also work), or a logical expression. Internally, `.TRUE.` is stored as 1 and `.FALSE.` as zero.
Strings:	The value can either be a string enclosed in quotes or a string variable. See section 8.3 for more details.

8.2 Indexed variables

Variables can be indexed, but only one-dimensional indexing is available. Indexed variables can be defined either individually, e.g.

```
R(1)=1.0 ANG
R(2)=1.2 ANG
R(3)=1.3 ANG
```

or as a vector of values enclosed by square brackets:

```
R=[1.0,1.1,1.2] ANG
```

Subranges can also be defined, e.g.

```
R(1)=1.0 ANG
R(2:3)=[1.1,1.2] ANG
```

leads to the same result as the above two forms.

The type of each element depends on the type of the assigned value, and it is possible to mix types in one variable. Example:

```
geometry={he}
hf
result=[program,energy,status.gt.0]
```

yields:

```
RESULT(1)      =    HF-SCF
RESULT(2)      =        -2.85516048   AU
RESULT(3)      =    TRUE
```

In this example the variables `PROGRAM`, `ENERGY`, and `STATUS` are system variables, which are set by the program (see section 8.4).

8.3 String variables

As explained already in section 8.1, string variables can be set as other variables in the form

variable = '*string*'
variable = *string_variable*

Strings must be enclosed by quotes. Otherwise the string is assumed to be a variable, and if this is undefined it is assumed to be zero.

Alternatively, if the name of the variable is preceded by a dollar ($), all values is assumed to be a string. This can a string variable, a quoted string, or an unquoted string. Note that unquoted strings are converted to upper case. Also note that quotes are compulsory if the string contains blanks.

Example:

```
$str=[a,b+4,'This is an example for strings']
```

yields

```
STR(1)          =    A
STR(2)          =    B+4
STR(3)          =    This is an example for strings
```

As a general rule, string variables are replaced by their value only if they are preceded by a dollar ($) (exceptions: in variable definitions, on `SHOW` cards, and in logical expressions on `IF` cards, the dollar is optional). This is a precaution to avoid commands which have the same name as a variable being interpreted as variables. Variables may also appear on `TEXT` or `TITLE` cards or in strings, but must be preceded by $ in these cases. Example:

```
$METHOD=MCSCF
R=1.5
TEXT,$method results for R=$R Bohr
```

prints

```
MCSCF results for R=1.5 Bohr
```

String variables can be concatenated with strings or other string variables in the following way. Assume that variable `PROGRAM` has the value `MRCI`. Setting

```
METHOD='$PROGRAM+Q'
```

sets `METHOD` to `MRCI+Q`. Alternatively, if we would also have a variable `VERSION` with value `Q`, we could write

```
METHOD='$PROGRAM+$VERSION'
```

Again, the value of `METHOD` would be `MRCI+Q`. Note that the quotes are necessary in these cases.

Substring operations are not implemented.

8.4 System variables

As mentioned above, most system variables cannot be written by the user. In some exceptions, it is possible to redefine them using the `SET` command:

`SET`,*variable* = *expression* [,] [*unit*]

This holds for the following variables:

`CHARGE`	Total charge of the molecule
`NELEC`	Number of electrons
`SPIN`	Spin quantum number, given as $2 \cdot M\_S$ (integer)
`SCFSPIN`	Same as `SPIN`, but only for HF
`MCSPIN`	Same as `SPIN`, but only for MCSCF
`CISPIN`	Same as `SPIN`, but only for MRCI
`STATE`	State to be optimized
`MCSTATE`	Same as `STATE` but only for MCSCF
`CISTATE`	Same as `STATE` but only for MRCI
`SYMMETRY`	State symmetry
`SCFSYM[METRY]`	Same as `SYMMETRY` but only for HF
`MCSYM[METRY]`	Same as `SYMMETRY` but only for MCSCF
`CISYM[METRY]`	Same as `SYMMETRY` but only for MRCI
`ZSYMEL`	Symmetry elements
`LQUANT`	Lambda quantum number for linear molecules
`OPTCONV`	Geometry optimization convergence criterion
`PROGRAM`	Last program name
`CPUSTEP`	CPU-time of last program step
`SYSSTEP`	System-time of last program step
`WALLSTEP`	Elapsed-time of last program step
`FOCKDONE`	Indicates if closed-shell fock operator is available.
`MAXBASIS`	Max number of basis sets stored on dump files. If the maximum is reached, the last one is overwritten when a new one is made, and all information (including dump records etc) of the previous basis is lost. The default is the maximum possible number of basis sets (30), which cannot be exceeded.

8.5 Macro definitions using string variables

String variables for which the stored string has the form of an algebraic expression are evaluated to a number if they are preceded by two dollars ($$). Example:

```
string='a+b'
a=3
b=4
text,This is string $string which evaluates to $$string
```

prints

```
** This is string a+b which evaluates to 7
```

This can be used to define simple macros, which can be used at various places in the subsequent input. For instance,

```
ECORR='ENERGY-ESCF'  !define a macro
HF                   !do SCF calculation
ESCF=ENERGY          !store SCF energy in variable ESCF
MULTI                !do CASSCF
DEMC=$$ECORR         !store CASSCF correlation energy in variable DEMC
MRCI                 !do MRCI
DECI=$$ECORR         !store MRCI correlation energy in variable DECI
```

Here is an example of advanced use of macros and string variables:

```
***,test for parser
text,This fancy input demonstrates how string variables and macros can be used
text
basis=vdz                !define basis set
geometry={O;H,O,r}       !define geometry (z-matrix)

text,methods
$method=[rhf,2[casscf,2[mrci]]]
text,active spaces
$spaces=['[3,1,1]',3['[4,2,2]'],3['[5,2,2]']]
text,symmetries
$symset=['1',2['[1,2,3]','1','2']]
text,weight factors for state averaged casscf
$weights=['1','[1,1,1]',2[' '],'[1,0.5,0.5]',2[' ']]
text,scf occupation
set,scfocc=[3,2[1]]
text,bond distance
r=1.85

hf
do i=1,#method          !loop over methods
mcocc=$$spaces(i)       !set active space for this run
set,symmetry=$$symset(i)    !set symmetries for this run
set,weight=$$weights(i)     !set weights for this run
$method(i)              !now run method
e(i)='$energy'          !save energies in strings
dipol(i)='$dmz'         !save dipole moments in strings
enddo
table,method,spaces,symset,weights,e,dipol
title,Results for OH, r=$r, basis=$basis
head,method,spaces,symmetries,weights,energies,'dipole moments'
exit
```

http://www.molpro.net/info/current/examples/oh_macros.com

8.6 Indexed Variables (Vectors)

Variables may be indexed, but only one-dimensional arrays (vectors) are supported. The index may itself be a variable. For instance

```
METHOD(I)=PROGRAM
E(I)=ENERGY
```

are valid variable definitions, provided `I`, `PROGRAM`, and `ENERGY` are also defined variables. Indices may be nested to any depth.

Different elements of an array can be of different type (either *real* or *logical*). However, only one *unit* can be assigned to an array. String variables have no associated value and cannot be mixed with the other variable types. Therefore, a given variable name can only be used either for a *string* variable or a *real* (*logical*) variable.

Vectors (arrays) can be conveniently defined using square brackets:

```
R=[1.0,1.2,1.3] ANG
```

This defines an array with three elements, which can be accessed using indices; for instance, `R(2)` has the value `1.2 ANG`. A repeat specifier can be given in front of the left bracket: `5[0]` is equivalent to `[0,0,0,0,0]`. Brackets can even be nested: for instance, `2[1,2,2[2.1,3.1]]` is equivalent to `[1,2,2.1,3.1,2.1,3.1,1,2,2.1,3.1,2.1,3.1]`.

Arrays can be appended from a given position just by entering additional elements; for instance,

```
R(4)=[1.4,1.5] ANG
```

or

```
R(4:)=[1.4,1.5] ANG
```

extends the above array to length 5. Previously defined values can be overwritten. For instance

```
R(2)=[1.25,1.35,1.45]
```

modifies the above vector to (1.0, 1.25, 1.35, 1.45, 1.5).

If no index is given on the left hand side of the equal sign, an existing variable of the same name is replaced by the new values, and all old values are lost. For instance

`THETA=[100,110,120,130]`	set four values
...	
`THETA(1)=104`	replace `THETA(1)` by a new value; `THETA(2:4)` are unchanged
...	
`THETA=[140,150]`	old variable `THETA` is replaced; `THETA(3:4)` are deleted

Square brackets can also be used to define an array of strings, e.g.,

```
METHOD=[INT,HF,CASSCF,MRCI]
```

These could be used as follows:

```
DO I=1,4
$METHOD(I)
ENDDO
```

The above input would be equivalent to

```
INT
HF
CASSCF
MRCI
```

The current length of an array can be accessed by preceding # to the variable name. For instance, in the above examples #R and #METHOD have the values 5 and 4, respectively. If a variable is not defined, zero is returned but no error occurs. This can be used to test for the existence of a variable, for example:

```
IF(#SPIN.EQ.0.AND.#NELEC.EQ.1) SET,SPIN=MOD(NELEC,2)
```

This defines variable SPIN if it is unknown and if NELEC is a scalar (one dimensional) variable.

8.7 Vector operations

The following simple vector operations are possible:

- Copying or appending a vector to another vector. For instance S=R copies a vector R to a vector S. S(3)=R copies R to S(3), S(4), ... S(#S+1)=R appends vector R to vector S. It is also possible to access a range of subsequent elements in a vector: S=R(2:4) copies elements 2 to 4 of R to S(1), S(2), S(3). Note that R(2:) denotes elements R(2) to R(#R), but R(2) denotes a single element of R.
- Vector-scalar operations: R=R*2 multiplies each element of R by 2. Instead of the number 2, also scalar (one dimensional) variables or expressions can be used, e.g., R=R*ANG converts all elements of R from Ångstrøm to bohr, or Z=R*COS(THETA) creates a vector Z with elements Z(i) = R(i)*COS(THETA). All other algebraic operators can be used instead of "*". Note that the scalar must come last since the first variable in the expression determines the vector length.
- Vector-vector operations: If A and B are vectors of the same length, then A × B is also a vector of this length. Here × stands for any algebraic operator, and the operation is done for each pair of corresponding elements. For instance, A + B adds the vectors A and B, and A * B multiplies their elements. Note that the latter case is not a scalar product. If an attempt is made to connect two vectors of different lengths by an algebraic operator, an error occurs.
- Intrinsic functions: Assume THETA=[100,110,120,-130] to be a vector of angles (in degrees). In this case X=2*COS(THETA) is also a vector containing the cosines of each element of THETA multiplied by two, i.e., X(i) = 2*COS(THETA(i)). MAX(THETA) or MIN(THETA) return the maximum and minimum values, respectively, in array THETA. Vector operations can also be nested, e.g., MAX(ABS(THETA)) returns the maximum value in array ABS(THETA).

At present, vector operations are not supported with string variables.

8.8 Special variables

8.8.1 Variables set by the program

A number of variables are predefined by the program. The following variables can be used to convert between atomic units and other units:

```
EV=1.d0/27.2113961d0 HARTREE
KELVIN=1.d0/3.157733d5 HARTREE
KJOULE=1.d0/2625.500d0 HARTREE
```

```
KCAL=1.d0/627.5096d0 HARTREE
CM=1.d0/219474.63067d0 HARTREE
CM-1=1.d0/219474.63067d0 HARTREE
HZ=1.d0/6.5796838999d15 HARTREE
HERTZ=1.d0/6.5796838999d15 HARTREE
ANG=1.d0/0.529177249d0 BOHR
ANGSTROM=1.d0/0.529177249d0 BOHR

TOEV=27.2113961d0 EV
TOK=3.157733d5 K
TOKELVIN=3.157733d5 K
TOCM=219474.63067d0 CM-1
TOHERTZ=6.5796838999d15 HZ
TOHZ=6.5796838999d15 HZ
TOKJ=2625.500d0 KJ/MOL
TOKJOULE=2625.500d0 KJ/MOL
TOKCAL=627.5096d0 KCAL/MOL
TOA=0.529177249d0 ANGSTROM
TOANG=0.529177249d0 ANGSTROM
TODEBYE=2.54158d0 DEBYE
```

Further variables which are set during execution of the program:

`INTYP`	defines integral program to be used. Either `INTS` (Seward) or `INTP` (Argos).
`INTDONE`	has the value `.true.` if the integrals are done for the current geometry.
`CARTESIAN`	Set to one if Cartesian basis functions are used.
`SCFDONE`	has the value `.true.` if an SCF calculation has been done for the current geometry.
`NUMVAR`	number of variables presently defined
`STATUS`	status of last step (1=no error, -1=error or no convergence)
`CHARGE`	Total charge of the molecule
`NELEC`	number of electrons in last wavefunction
`SPIN`	spin multiplicity minus one of last wavefunction
`ORBITAL`	record of last optimized orbitals (set but never used in the program)
`LASTORB`	Type of last optimized orbitals (`RHF`, `UHF`, `UHFNAT`, or `MCSCF`.
`LASTSYM`	Symmetry of wavefunction for last optimized orbitals.
`LASTSPIN`	$2*M_S$ for wavefunctions for last optimized orbitals.
`LASTNELEC`	Number of electrons in wavefunction for last optimized orbitals.
`ENERGR(istate)`	Reference energy for state *istate* in MRCI and CCSD.
`ENERGY(istate)`	last computed total energy for state *istate* for the method specified in the input (e.g., `HF`, `MULTI`, `CCSD(T)`, or `CCSD[T]`.
`ENERGD(istate)`	Total energy for state *istate* including Davidson correction (set only in `CI`).

ENERGP(istate)	Total energy for state *istate* including Pople correction (set only in CI).
ENERGT(1)	Total energy including perturbative triples (T) correction (set only in CCSD(T), QCI(T)).
ENERGT(2)	Total energy including perturbative triples [T] correction (set only in CCSD(T), QCI(T)).
ENERGT(3)	Total energy including perturbative triples -t correction (set only in CCSD(T), QCI(T)).
EMP2	holds MP2 energy in MPn, CCSD, BCCD, or QCISD calculations, and RS2 energy in MRPT2 (CASPT2) calculations.
EMP3	holds MP3 energy in MP3 and MP4 calculations, and RS3 energy in MRPR3 (CASPT3) calculations.
EMP4	holds MP4(SDQ) energy in MP4 calculations. The MP4(SDTQ) energy is stored in variable ENERGY.
METHODC	String variable holding name of the methods used for ENERGC, e.g., CCSD, BCCD, QCI.
METHODT(1)	String variable holding name of the methods used for ENERGT(1), e.g., CCSD(T), BCCD(T), QCI(T).
METHODT(2)	String variable holding name of the methods used for ENERGT(2), e.g., CCSD[T], BCCD[T], QCI[T].
METHODT(3)	String variable holding name of the methods used for ENERGT(3), e.g., CCSD-T, BCCD-T, QCI-T.
ENERGC	Total energy excluding perturbative triples correction (set only in QCI or CCSD with triples correction enabled).
DFTFUN	total value of density functional in DFT or KS.
DFTFUNS(ifun)	value of ifun'th component of density functional in DFT or KS.
DFTNAME(ifun)	name of ifun'th component of density functional in DFT or KS.
DFTFAC(ifun)	factor multiplying ifun'th component of density functional in DFT or KS.
DFTEXFAC	factor multiplying exact exchange in KS.
PROP(istate)	computed property for state *istate*. See below for the names PROP of various properties.
PROGRAM	last program called, as specified in the input (e.g., HF, CCSD(T), etc.)
ITERATIONS	Number of iterations used. Set negative if no convergence or max number of iterations reached.
CPUSTEP	User-CPU time in seconds for last program called.
SYSSTEP	System-CPU time in seconds for last program called.
WALLSTEP	Elapsed time in seconds for last program called.

The variable names for properties are the same as used on the EXPEC input cards.

OV	Overlap
EKIN	Kinetic energy

POT	Potential
DELTA	Delta function
DEL4	∇^4
DARWIN	Darwin term of relativistic correction
MASSV	Mass-velocity term of relativistic correction
EREL	Total relativistic correction
DMX, DMY, DMZ	Dipole moments
XX, YY, ZZ, XY, XZ, XY	Second moments
XXX, XXY, XXZ, XYY, XYZ, XZZ, YYY, YYZ, YZZ, ZZZ	Third moments
QMXX, QMYY, QMZZ, QMXY, QMXZ, QMXY	Quadrupole moments
EFX, EFY, EFZ	Electric field
FGXX, FGYY, FGZZ, FGXY, FGXZ, FGXY	Electric field gradients
D/DX, D/DY, D/DZ	Velocity
LSX, LSY, LSZ	One-electron spin-orbit
LL	Total angular momentum squared L^2
LX, LY, LZ	Electronic angular momentum
LXLX, LYLY, LZLZ, LXLY, LXLZ, LYLZ	Two-electron angular momentum

By default, only the dipole moments are computed and defined. The values of other properties are only stored in variables if they are requested by EXPEC cards. If more than one state is computed (e.g., in state-averaged MCSCF, corresponding arrays PROP(istate) are returned. If properties are computed for more than one center, the center number is appended to the name, e.g. EFX1, EFX2 etc.

If transition properties are computed, their values are stored in corresponding variables with prefix TR, e.g., TRDMX, TRDMY, TRDMZ for transition dipole moments. If more than two states are computed, the index is $(i-1)*(i-2)/2+j$, where $i > j \geq 1$ are state numbers. In a state-averaged calculation, states are counted sequentially for all state symmetries.

For instance, in the following state-averaged MCSCF

```
MULTI;WF,14,1,0;STATE,3;WF,14,2,0;STATE,2;WF,3,0
```

the states are counted as

i	1	2	3	4	5	6
Symmetry	1	1	1	2	2	3
Root in Sym.	1	2	3	1	2	1

8.8.2 Variables recognized by the program

All variables described below are checked by the program, but not set (except NELEC and SPIN). If these are not defined by the user, the program uses its internal defaults. The variables are only recognized and used if defined using the SET command, e.g.

```
SET,MCOCC=[6,3,2]
SET,STATE=2
```

etc.

Variables recognized by the SCF program:

CHARGE	Total charge of the molecule (can be given instead of nelec)
NELEC	number of electrons
SPIN	spin multiplicity minus one
SCFSYM[METRY]	wavefunction symmetry
SYMMETRY	as SCFSYMM; only used if SCFSYMM is not present.
SCFOC[C]	number of occupied orbitals in each symmetry for SCF
SCFCL[OSED]	number of closed-shell orbitals in each symmetry for SCF
SCFORB	record of saved orbitals in SCF
SCFSTART	record of starting orbitals used in SCF

Variables recognized by the MCSCF program:

CHARGE	Total charge of the molecule (can be given instead of nelec)
NELEC	number of electrons
MCSYM[METRY]	wavefunction symmetry. This can be an array for state-averaged calculations.
SYMMETRY	as MCSYMM; only used if MCSYMM is not present.
MCSPIN	spin multiplicity minus one. This can be an array for state-averaged calculations, but different spin multiplicities can only be used in determinant CASSCF. If only one value is specified, this is used for all states
SPIN	as MCSPIN; only used if MCSPIN is not present.
MCSTATE	number of states for each symmetry in MCSCF
STATE	as MCSTATE; only used if MCSTATE is not present.
WEIGHT	weight factors for all states defined by SYMMETRY and STATE
LQUANT	Eigenvalues of L_z^2 for linear molecules for each state defined by SYMMETRY and STATE.
MCSELECT	records from which configurations can be selected and selection threshold
SELECT	as MCSELECT; only used if MCSELECT is not present.
MCRESTRICT	can be used to define occupancy restrictions
RESTRICT	as MCRESTRCT; only used if MCRESTRICT is not present:
CONFIG	if set to .true. or to one triggers use of CSFs
MCOC[C]	number of occupied orbitals in each symmetry
OCC	as MCOCC; only used if MCOCC is not present.
MCCL[OSED]	number of optimized closed-shell orbitals in each symmetry
CLOSED	as MCCLOSED; only used if MCCLOSED is not present.
MCFROZEN	number of frozen core orbitals in each symmetry
FROZEN	as MCFROZEN; only used if MCFROZEN is not present.
MCSTART	record of starting orbitals
COREORB	record of frozen core orbitals

`MCORB`	record for saving optimized orbitals
`MCSAVE`	records for saving CI wavefunction (like `SAVE` card in MCSCF)

Variables recognized by the CI/CCSD program:

`CHARGE`	Total charge of the molecule (can be given instead of nelec)
`NELEC`	number of electrons
`SPIN`	spin multiplicity minus one
`CISYM[METRY]`	wavefunction symmetry. If this is an array, only SYMMETRY(1) is used.
`SYMMETRY`	as `CISYMM`; only used if `CISYMM` is not present.
`CISTATE`	number of states in CI
`STATE`	as `CISTATE`, only used if `CISTATE` is not present.
`CISELECT`	records from which configurations can be selected
`SELECT`	as `CISELECT`; only used if `CISELCT` is not present.
`CIRESTRICT`	defines occupancy restrictions
`RESTRICT`	as `RESTRICT`; only used if `CIRESTRICT` is not present.
`CIOC[C]`	number of occupied orbitals in each symmetry
`OCC`	`as CIOCC`; only used if `CIOCC` is not present.
`CICL[OSED]`	number of closed-shell orbitals in each symmetry
`CLOSED`	`as CICLOSED`; only used if `CICLOSED` is not present.
`CICO[RE]`	number of core orbitals in each symmetry
`CORE`	`as CICORE`; only used if `CICORE` is not present.
`CIORB`	record of orbitals used in CI
`CISAVE`	records for saving CI wavefunction (like `SAVE` card in CI)
`CISTART`	records for restarting with previous CI wavefunction (like `START` card in CI)

Variables recognized by the DFT/KS program:

`DF(ifun)` or `DFTNAME(ifun)`	name of `ifun`'th component of density functional.
`DFTFAC(ifun)`	factor multiplying `ifun`'th component of density functional.
`DFTEXFAC`	factor multiplying exact exchange in `KS`.

Example for the use of these variables for a state-averaged MCSCF (note that system variables can only be modified using the SET command, see section 8.4):

`SET,NELEC=9`	defines number of electrons
`SET,SPIN=1`	defines wavefunction to be a doublet
`SET,SYMMETRY=[1,2,3]`	defines wavefunction symmetries for state averaged calculation
`SET,STATE=[2,1,1]`	defines number of states to be averaged in each symmetry
`WEIGHT=[2,2,1,1]`	defines weights for the above four states

`OCC=[5,2,2]`	number of occupied orbitals in each symmetry
`CLOSED=2`	number of closed-shell orbitals in symmetry 1
`MCORB=3100.2`	record for optimized orbitals
`MULTI`	do mcscf with above parameters

Note: Setting the variables `NELEC`, `SPIN`, or `SYMMETRY`, has the same effect giving these on a gobal `WF` directive. If the global `WF` directive is given after the variable definition, the values of the variables are replaced by the values given on the `WF` directive. Vice versa, if a variable definition follows a gobal `WF` directive, the new value of the variable is used in the following. Note that `WF` input cards in command blocks have preference over global `WF` directives or input variables.

8.9 Displaying variables

Variables or the results of expressions can be displayed in the output using `SHOW` and `TABLE`.

8.9.1 The `SHOW` command

The general form of the `SHOW` command is as follows:

`SHOW[`*ncol*`,`*format*`],`*expression*

where *expression* can be an expression or variable, *ncol* is the number of values printed per line (default 6), and *format* is a format (default 6F15.8). This can be used to print vectors in matrix form. The specification of *ncol* and *format* is optional. Assume that `E` is a vector:

`SHOW,E`	prints `E` using defaults.
`SHOW[n],E`	prints `E` with `n` elements per line; (if `n>6`, more than one line is needed, but in any case a new line is started after `n` elements).
`SHOW[n,10f10.4],E`	prints `E` in the format given, with newline forced after `n` elements.

Note that the total length of the format should not exceed 100 characters (a left margin of 30 characters is always needed).

A *wild card* format can be used to show several variables more easily:

`SHOW,qm*,dm*`

shows all variables whose names begin with `QM` and `DM`. Note that no letters must appear after the `*`, i.e., the wild card format is less general than in `UNIX` commands.

See the `TABLE` command for another possibility to tabulate results.

8.10 Clearing variables

Variables can be deleted using

`CLEAR,`*name1, name2, . . .*

Wild cards can be used as in `SHOW`, e.g.,

`CLEAR,ENERG*`

clears all variables whose names begin with ENERG. All variables can be cleared using

CLEARALL

The length of vectors can be truncated simply by redefining the length specifier: #R=2 truncates the array R to length 2. Higher elements are no longer available (but could be redefined). Setting #R=0 is equivalent to the command CLEAR,R.

8.11 Reading variables from an external file

Variables can be read from an external file using

READVAR,*filename*, [*option*]

Such files can be save, for instance by the geometry optimization program, and reused later to recover a certain optimized geometry. The format of the input in *filename* is the same as for ordinary input.

If *option*=NOINDEX|IGNOREINDEX is given then variable indices are ignored and only the last value read is saved (without index). This can be useful if for example a file saved with SAVEACT in a geometry optimization is read, and it is intended to continue with the variables that were saved last.

9 TABLES AND PLOTTING

9.1 Tables

Variables can be printed in Table form using the command

TABLE,*var1*,*var2*,...

The values of each variable are printed in one column, so all variables used must be defined for the same range, and corresponding elements should belong together. For example, if in a calculation one has stored R(i), THETA(i), ECI(i) for each geometry *i*, one can print these data simply using

TABLE, R, THETA, ECI

By default, the number of rows equals the number of elements of the first variable. This can be changed, however, using the RANGE subcommand.

The first ten columns of a table may contain string variables. For instance,

```
hf;etot(1)=energy;method(1)=program;cpu(1)=cpustep
ccsd;etot(2)=energy;method(2)=program;cpu(2)=cpustep
qci;etot(3)=energy;method(3)=program;cpu(3)=cpustep
table,method,etot,cpu
```

prints a table with the SCF, CCSD, and QCI results in the first, second, and third row, respectively. For other use of string variables and tables see, e.g. the examples h2o_tab.com and oh_macros.com

The apparence of the table may be modified using the following commands, which may be given (in any order) directly after the the TABLE card:

`HEADING`,*head1, head2,...*	Specify a heading for each column. By default, the names of the variables are used as headings.
`FORMAT`,*format*	Specify a format for each row in fortran style. *format* must be enclosed by quotes. Normally, the program determines automatically an appropriate format, which depends on the type and size of the printed data.
`FTYP`,*typ1, typ2, typ3, ...*	Simplified form to modify the format. This gives the type (`A`, `F`, or `D`) for each column (sensible defaults are normally used).
`DIGITS`,*dig1, dig2, dig3, ...*	Give the number of digits after the decimal points to be printed for each column (sensible defaults are normally used).
`TYPE`	Specify a data format for the table. The default is `TEXT` which gives a plain text file. Other possibilities are `CSV` (comma-separated fields suitable for a spreadsheet), `LATEX` (a LaTeX `table` environment), `MATHEMATICA` (Mathematica code that assigns the table to an array), `MATLAB` (Matlab code that assigns the table to an array), `MAPLE` (Maple code that assigns the table to an array), `HTML` (an HTML TABLE construction), and `XML` (an XML document containing a tree representing the table. The actual format is XHTML).
`SAVE`,*file,status*	Specify a file on which the table will be written. If status is `NEW`, the file is rewound, otherwise it is appended. If *file* has a suffix that is one of `txt, csv, tex, m, mpl, html, xml`, and a `TYPE` command is not specified, then the type will be set to that which is conventionally appropriate for the suffix. If *file* is omitted, then a file name is automatically generated, with the form *input*.`table`*n*.*ext*: *input* is the basename of the input file (or `molpro` if running from standard input); *n* is a sequence number that is incremented by one each time a table is produced; *ext* is a suffix appropriate to the file format, eg `txt`, `html`, etc.
`TITLE`,*title*	Specify one line of a title (several `TITLE` cards may follow each other). Note that titles are only displayed in the `SAVE` file, if the `SAVE` command is given before the `TITLE` card.
`SORT`,*col1,col2,...*	Sort rows according to increasing values of the given columns. The columns are sorted in the order they are specified.
`PRINT`,*key1,key2,...*	Specify print options (`TABLE, HEADING, TITLE, WARNING, FORMAT, SORT`). The default is print for the first three, and noprint for the last three.
`NOPRINT`,*key1,key2,...*	Disable print for given keys.
`NOPUNCH`	Don't write data to the punch file (data are written by default).
`RANGE`,*start,end*	Specify start and end indices of the variables to be printed.
`STATISTICS`	Print also linear regression and quadratic fits of the data columns.

9.2 Plotting

[`PLOT`[,*col1*,*col2*,...][,*options*]

Construct input for a plotting program using the table as data. `PLOT` is a subcommand of `TABLE` and must follow `TABLE` or any of its valid subcommands given in the previous section. More than one `PLOT` command can be included within a single `TABLE`, and each invocation generates a new plot. However, `PLOT` must appear after all other `TABLE` subcommands.

col1, *col2*,... are the names of the table columns to be plotted. These must be an exact subset of those given on the `TABLE` command. The first column is taken as abscissa, and the values of the remainder will be plotted against it. If no columns are specified, then the entire table is plotted; if a single column is specified, it will be used as abscissa, and all other columns in the table will be plotted as ordinate. *options* can be chosen from the following.

`CMD=`*unix_plot_command*	*unix_plot_command* consists of the system command needed to start the plotting program, followed by any required options. The whole thing should normally be enclosed in quotation marks to preserve lower-case letters. The default is `'xmgrace'`. At present, the *Grace* program (also known as *xmgrace, grace, gracebat*), with only numerical data, is supported. The output is also compatible with the portable drop-in replacement for Grace, *AptPlot*, and if *Grace* is not found on the system, Molpro will attempt to use *AptPlot* as default instead.
`FILE=`*plotfile*	By default the input file for the plotting program is saved in *input*.`table`*n*.`plot`*m*.`agr`, where *m* is an automatically generated sequence number. The name of the plotfile can be modified using the `FILE` option.
`INTERACTIVE`	By default, the plot is not shown on the screen but all plot data are saved in the given file. The plotting program can be started interactively by giving the `INTERACTIVE` option.
`TYPE=`*type*	If `TYPE` is specified, *type* should be set to one of `pdf`, `svg`, `png`, `jpeg` or `eps`. The result is that the `gracebat` program is executed on the plot input file to generate the graph output file in the desired format. This feature depends on the availability of `gracebat`, and on it supporting the requested output format (for example, at present pdf is supported under Mac OS X, but not in some Linux systems).
`BACKGROUND=`*rgb*	*rgb* should be a string of six hexadecimal digits specifying the red-green-blue colour to use for the background of the plot.
`BACKGROUND=TRANSPARENT`	The background of the plot is made transparent (currently implemented only for `TYPE=svg`).
`NOSPLINE`	Prevents spline interpolation of data points
`NSPLINE=`*number*	Number of interpolation points (default 20)
`LEGEND='`*x, y*`'`	Position legend at (x,y) on plot.
`LEGEND=OFF`	Do not draw legend; this behaviour is chosen automatically when there is only a single ordinate dataset.
`PCOMMAND='`*command*`'`	Insert arbitrary *Grace* command into the plot file; for details, consult http://plasma-gate.weizmann.ac.il/Grace/doc/UsersGuide.html#s5.

The following additional directives can be given *before* the `PLOT` directive:

`COLOUR,`*icolour1, icolour2,...*	Colour map to be used for columns 1,2,...; zero means to use default values (colours black, blue, red, green cycle)

COLOUR,*rgb1, rgb2,...*	Absolute colours (6-hex-digit rgb values) to be used for columns 1,2,...;
SYMBOL,*isymb1, isymb2,...*	Symbol types to be used for columns 1,2,...; -1 means no symbols; zero means to use default values.
LINEWIDTH,*width1, width2,...*	Line widths to be used for columns 1,2,...; omit to use default values.
LINESTYLE,*style1, style2,...*	Line styles to be used for columns 1,2,...; omit to use default values.

9.3 Diatomic potential curve analysis

For the case that a table contains one or more potential energy functions for a diatomic molecule, with the first column containing bond lengths in Bohr or Ångstrom, it is possible to calculate spectroscopic constants using

DIATOMIC[,DEGREE=*n*][,MASS=*m*][,PRINT=*p*]

The data are fitted to a polynomial of degree *n* (default is number of points minus 1, ie interpolation), and spectroscopic constants calculated using reduced mass *m* expressed in u. Note that it is possible to constrain which bond lengths are used through the use of the RANGE subcommand.

10 MOLECULAR GEOMETRY

10.1 Geometry specifications

The geometry may be given in standard Z-matrix form, or XYZ form. The geometry specifications are given in the form

[SYMMETRY, *options*]

[ORIENT, *options*]

[ANGSTROM]

GEOMETRY={
atom specifications
}

GEOMETRY must come after the other commands that modify the way the geometry is constructed. The following are permitted as SYMMETRY options:

	Any valid combination of symmetry generators, as described in section 10.2 below.
NOSYM	Disable use of symmetry. Instead of SYMMETRY,NOSYM also just NOSYM can be used.

The following are permitted as ORIENT options:

CHARGE	Orient molecule such that origin is centre of charge, and axes are eigenvectors of quadrupole moment.

`MASS` Orient molecule such that origin is centre of mass, and axes are eigenvectors of inertia tensor (default for Z-matrix input). Alternatively, the symmetry centre can be specified as `CENTRE=MASS|CHARGE`.

`NOORIENT` Disable re-orientation of molecule (default for XYZ-input).

`SIGNX=±1` Force first non-zero x-coordinate to be positive or negative, respectively. Similarly, `SIGNY`, `SIGNZ` can be set for the y- and z-coordinates, respectively. This can be useful to fix the orientation of the molecule across different calculations and geometries. Alternatively, the system variables `ZSIGNX`, `ZSIGNZ`, `ZSIGNZ` can be set to positive or negative values to achieve the same effect.

`PLANEXZ` For the C_{2v} and D_{2h} point groups, force the primary plane to be xz instead of the default yz. The geometry builder attempts by swapping coordinate axes to place as many atoms as possible in the primary plane, so for the particular case of a planar molecule, this means that all the atoms will lie in the primary plane. The default implements recommendation 5*a* and the first part of recommendation 5*b* specified in J. Chem. Phys. 55, 1997 (1955). `PLANEYZ` and `PLANEXY` may also be specified, but note that the latter presently generates an error for C_{2v}.

`ANGSTROM` Forces bond lengths that are specified by numbers, or variables without associated units, to use the values as a number of Ångstrom, rather than Bohr.

10.1.1 Z-matrix input

The general form of an atom specification line is

`[`*group*`[,]]`*atom*, p_1, r, p_2, α, p_3, β, J

or, alternatively,

`[`*group*`[,]]`*atom*, p_1, x, y, z

where

group atomic group number (optional). Can be used if different basis sets are used for different atoms of the same kind. The basis set is then referred to by this group number and not by the atomic symbol.

atom chemical symbol of the new atom placed at position p_0. This may optionally be appended (without blank) by an integer, which can act as sequence number, e.g., `C1`, `H2`, etc. Dummy centres with no charge and basis functions are denoted either `Q` or `X`, optionally appended by a number, e.g, `Q1`; note that the first atom in the z-matrix must not be called `X`, since this may be confused with a symmetry specification (use `Q` instead).

p_1 atom to which the present atom is connected. This may be either a number n, where n refers to the n'th line of the Z-matrix, or an alphanumeric string as specified in the *atom* field of a previous card, e.g., `C1`, `H2` etc. The latter form works only if the atoms are numbered in a unique way.

r	Distance of new atom from p_1. This value is given in bohr, unless `ANG` has been specified directly before or after the symmetry specification.
p_2	A second atom needed to define the angle $\alpha(p_0, p_1, p_2)$. The same rules hold for the specification as for p_1.
α	Internuclear angle $\alpha(p_0, p_1, p_2)$. This angle is given in degrees and must be in the range $0 < \alpha < 180^0$.
p_3	A third atom needed to define the dihedral angle $\beta(p_0, p_1, p_2, p_3)$. Only applies if $J = 0$, see below.
β	Dihedral angle $\beta(p_0, p_1, p_2, p_3)$ in degree. This angle is defined as the angle between the planes defined by (p_0, p_1, p_2) and (p_1, p_2, p_3) $(-180^0 \leq \beta \leq 180^o)$. Only applies if $J = 0$, see below.
J	If this is specified and nonzero, the new position is specified by two bond angles rather than a bond angle and a dihedral angle. If $J = \pm 1$, β is the angle $\beta(p_0, p_1, p_3)$. If $J = 1$, the triple vector product $(\mathbf{p}_1 - \mathbf{p}_0) \cdot [(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3)]$ is positive, while this quantity is negative if $J = -1$.
x,y,z	Cartesian coordinates of the new atom. This form is assumed if $p_1 \leq 0$; if $p_1 < 0$, the coordinates are frozen in geometry optimizations.

All atoms, including those related by symmetry transformations, should be specified in the Z-matrix. Note that for the first atom, no coordinates need be given, for the second atom only p_1, r are needed, whilst for the third atom p_3, β, J may be omitted. The 6 missing coordinates are obtained automatically by the program, which translates and re-orients the molecule such that the origin is at the centre of mass, and the axes correspond to the eigenvectors of the inertia tensor (see also `CHARGE` option above).

Variable names, and in general expressions that are linear in all dependent variables, may be used as well as fixed numerical values for the parameters r, α and β. These expressions are evaluated as late as possible, so that it is possible, for example, to set up loops in which these parameters are changed; the geometry optimizer also understands this construction, and will optimize the energy with respect to the value of the variables. Non-linear expressions should not be used, because the geometry optimization module is unable to differentiate them.

Once the reorientation has been done, the program then looks for symmetry (D_{2h} and subgroups), unless the `NOSYM` option has been given. It is possible to request that reduced symmetry be used by using appropriate combinations of the options `X,Y,Z,XY,XZ,YZ,XYZ`. These specify symmetry operations, the symbol defining which coordinate axes change sign under the operation. The point group is constructed by taking all combinations of specified elements. If symmetry is explicitly specified in this way, the program checks to see that the group requested can be used, swapping the coordinate axes if necessary. This provides a mechanism for ensuring that the same point group is used, for example, at all points in the complete generation of a potential energy surface, allowing the safe re-utilization of neighbouring geometry molecular orbitals as starting guesses, etc..

Note that symmetry is not implemented in density fitting methods, and in these cases the NOSYM option is implied automatically.

Also note that by default the automatic orientation of the molecule only takes place if the geometry is defined by internal (Z-matrix) coordinates. In case of `XYZ` Input the orientation is unchanged, unless the `MASS` option is specified in the geomnetry block.

10.1.2 *XYZ* input

Simple cartesian coordinates in Ångstrom units can be read as an alternative to a Z matrix. This facility is triggered by setting the MOLPRO variable GEOMTYP to the value XYZ before the geometry specification is given, but usually this does not need to be done, as a geometry specification where the first line is a single integer will be recognized as XYZ format, as will the case of the first line consisting of a chemical symbol followed by three cartesian coordinates. The geometry block should then contain the cartesian coordinates in Minnesota Computer Centre, Inc. XYZ format. Variable names, and in general expressions that are linear in all dependent variables, may be used as well as fixed numerical values. Non-linear expressions should not be used, because the geometry optimization module is unable to differentiate them.

The XYZ file format consists of two header lines, the first of which contains the number of atoms, and the second of which is a title. The remaining lines each specify the coordinates of one atom, with the chemical symbol in the first field, and the x, y, z coordinates following. A sequence number may be appended to the chemical symbol; it is then interpreted as the atomic group number, which can be used when different basis sets are wanted for different atoms of the same kind. The basis set is then specified for this group number rather than the atomic symbol. As a further extension, the first two header lines can be omitted.

Note that for XYZ input the default is not to reorient the molecule. Orientation can be forced, however, by the MASS or CHARGE options on the ORIENT directive.

```
geomtyp=xyz
geometry={
3           ! number of atoms
This is an example of geometry input for water with an XYZ file
O ,0.0000000000,0.0000000000,-0.1302052882
H ,1.4891244004,0.0000000000, 1.0332262019
H,-1.4891244004,0.0000000000, 1.0332262019
}
hf
```

http://www.molpro.net/info/current/examples/h2o_xyzinput.com

The XYZ format is specified within the documentation distributed with MSCI's XMol package. Note that MOLPRO has the facility to write XYZ files with the PUT command (see section 10.3).

10.2 Symmetry specification

If standard Z-matrix input is used, MOLPRO determines the symmetry automatically by default. However, sometimes it is necessary to use a lower symmetry or a different orientation than obtained by the default, and this can be achieved by explicit specification of the symmetry elements to be used, as described below.

Generating symmetry elements, which uniquely specify the point group, can be specified on the SYMMETRY directive. This must be given *before* the geometry block. Each symmetry directive only affects the subsequent geometry block; after a geometry block has been processed, the defaults are restored. Note that the specification of symmetry elements inside the geometry block is no longer allowed.

The dimension of the point group is 2**(number of fields given). Each field consists of one or more of X, Y, or Z (with no intervening spaces) which specify which coordinate axes change sign under the corresponding generating symmetry operation. It is usually wise to choose z to be the unique axis where appropriate (essential for C_2 and C_{2h}). In that case, the possibilities are:

(null card)	C_1 (i.e., no point group symmetry)
`Z`	C_s
`XY`	C_2
`XYZ`	C_i
`X,Y`	C_{2v}
`XY,Z`	C_{2h}
`XZ,YZ`	D_2
`X,Y,Z`	D_{2h}

Note that Abelian point group symmetry only is available, so for molecules with degenerate symmetry, an Abelian subgroup must be used — e.g, C_{2v} or D_{2h} for linear molecules.

See section 10.2 for more details of symmetry groups and ordering of the irreducible representations. Also see section 10.1.1 for more information about automatic generation of symmetry planes.

Note that by default the automatic orientation of the molecule only takes place if the geometry is defined by internal (Z-matrix) coordinates. In case of `XYZ` Input the orientation is unchanged, unless the `MASS` option is specified in the geometry block.

10.3 Writing Gaussian, XMol or MOLDEN input (`PUT`)

The `PUT` command may be used at any point in the input to print, or write to a file, the current geometry. The syntax is

`PUT`,*style,file,status,info*

If *style* is `GAUSSIAN`, a complete Gaussian input file will be written; in that case, `info` will be used for the first (route) data line, and defaults to '`# SP`'.

If *style* is `XYZ`, an XYZ file will be written (see also section 10.1.2). If *style* is `CRD`, the coordinates will be written in CHARMm CRD format.

If *style* is `MOLDEN`, an interface file for the MOLDEN visualization program is created; further details and examples are given below.

If *style* is omitted, the Z-matrix, current geometry, and, where applicable, gradient are written.

file specifies a file name to which the data is written; if blank, the the data is written to the output stream. If *status* is omitted or set to `NEW`, any old contents of the file are destroyed; otherwise the file is appended.

10.3.1 Visualization of results using Molden

Geometry, molecular orbital, and normal mode information, when available, is dumped by `PUT,MOLDEN` in the format that is usable by MOLDEN.

The interface to the gOpenMol program offers an alternative visualization possibility, and is described in section 34.8.

The example below generates all the information required to plot the molecular orbitals of water, and to visualize the normal modes of vibration:

```
***,H2O
geometry={angstrom;o;h,o,roh;h,o,roh,h,theta};
roh=1.0
theta=104.0
rhf;
optg;
{frequencies;
print,low,img;}
put,molden,h2o.molden;
```

http://www.molpro.net/info/current/examples/h2o_put_molden.com

The example below does a difference density by presenting its natural orbitals to MOLDEN. Note that it although MOLDEN has internal features for difference density plots, the approach show here is more general in that it bypasses the restriction to STO-3G, 3-21G, 4-31G and 6-31G basis sets.

```
gprint,orbitals
symmetry,y,planexz
geometry={O;H1,O,r;h2,O,r,h1,alpha}
r=1.8
alpha=104
int;
{hf;wf,10,1;orbital,2100.2}
{multi;wf,10,1;orbital,2140.2}

{matrop
load,dscf,density,2100.2        !load scf density
load,dmcscf,density,2140.2      !load mcscf density
add,ddiff,dmcscf,-1,dscf        !compute dmcscf-dscf
natorb,neworb1,dscf
natorb,neworb2,dmcscf
natorb,neworbs,ddiff
save,neworbs,2110.2
save,ddiff,2110.2}

put,molden,h2o_ddens.molden;orb,2110.2
```

http://www.molpro.net/info/current/examples/h2o_diffden_molden.com

10.4 Geometry Files

Using the format

GEOMETRY=*file*

the geometry definitions are read from *file*, instead of inline.

10.5 Lattice of point charges

LATTICE,[INFILE=*input_file*,] [OUTFILE=*output_file*,] [VARGRAD,] [NUCONLY,] [REMOVE]

Includes a lattice of point charges, for use in QM/MM calculations for example (see section 53). An external file (*input_file*) should be given as input, with the following format:

Comment line
number of point charges N
$x1, y1, z1, q1$,*flag*1
⋮
xN, yN, zN, qN,*flagN*

The x, y and z fields stand for the point charge coordinates (in Å), q for its charge and *flag=1* indicates that gradients should be computed for this lattice point (0 means no gradient).

outfile specifies a file name to which the lattice gradient is written; if blank, it will be written to the output stream.

VARGRAD	(logical) Stores the lattice gradient in variable VARGRAD.
NUCONLY	(logical) Disables gradient evaluation with respect to the lattice, independent of *flag* in the lattice file.
REMOVE	(logical) Removes the lattice.

Symmetry is not supported for lattice gradients.

10.6 Redefining and printing atomic masses

The current masses of all atoms can be printed using

MASS,PRINT

The atomic masses can be redefined using

MASS, [*type*,] [*symbol*=*mass*, . . .]

The optional keyword *type* can take either the value AVER[AGE] for using average isotope masses, or ISO[TOPE] for using the masses of the most abundant isotopes. This affects only the rotational constants and vibrational frequencies. As in most quantum chemistry packages, the default for *type* is AVERAGE. If INIT is given, all previous mass definitions are deleted and the defaults are reset.

Individual masses can be changed by the following entries, where *symbol* is the chemical symbol of the atom and *mass* is the associated mass. Several entries can be given on one MASS card, and/or several MASS cards can follow each other. The last given mass is used.

Note that specifying different isotope masses for symmetry related atoms lowers the symmetry of the system if the molecular centre of mass is taken as the origin. This effect can be avoided by using the charge centre as origin, i.e., specifying CHARGE as first entry in the GEOMETRY input:

GEOMETRY={CHARGE; ...}

10.7 Dummy centres

DUMMY,*atom1,atom2,*. . .

Sets nuclear charges on atoms 1,2 etc. to zero, for doing counterpoise calculations, for example. *atom1, atom2,*. . . can be Z-matrix row numbers or tag names. Note that the current setting of dummies is remembered by the program across restarts via the MOLPRO variable

DUMMYATOMS. Dummies can be reset to their original charges using a DUMMY card with no entries. Dummy centres are also reset to their original charges if (i) and INT command is encountered, or (ii) a new geometry input is encountered.

The program does not recognize automatically if the symmetry is reduced by defining dummy atoms. Therefore, for a given dummy atom, either all symmetry equivalent atoms must also be dummies, or the symmetry must be reduced manually as required. An error will result if the symmetry is not consistent with the dummy centre definitions.

10.7.1 Counterpoise calculations

Counterpoise corrections are easily performed using dummy cards. One first computes the energy of the total system, and then for the subsystems using dummy cards.

10.7.2 Example: interaction energy of OH-Ar

```
***,OH(2Sig+)-Ar linear
memory,2,m
geometry={q1;                  !dummy center in center of mass
o,q1,ro;h,q1,rh,o,180;         !geometry of OH
ar,q1,rar,o,theta,h,0}         !geometry of Ar
roh=1.8                        !OH bond-length
rar=7.5                        !distance of Ar from center of mass
theta=0                        !angle OH-Ar
ro=roh*16/17                   !distance of O from center of mass
rh=roh*1/17                    !distance of H from center of mass
basis=avdz                     !basis set

text,calculation for complex
{rhf;occ,8,3,3;wf,27,1,1}      !RHF for total system
rccsd(t)                       !CCSD(T) for total system
e_ohar=energy                  !save energy in variable e_ohar

text,cp calculation for OH
dummy,ar                       !make Ar a dummy center
{rhf;occ,3,1,1;wf,9,1,1}       !RHF for OH
rccsd(t)                       !CCSD(T) for OH
e_oh=energy                    !save energy in variable e_oh

text,cp calculation for Ar
dummy,o,h                      !make OH dummy
hf                             !scf for Ar
ccsd(t)                        !CCSD(T) for Ar
e_ar=energy                    !save energy in variable e_ar

text,separate calculation for OH
geometry={O;H,O,roh}           !geometry for OH alone
{rhf;occ,3,1,1;wf,9,1,1}       !RHF for OH
rccsd(t)                       !CCSD(T) for OH
e_oh_inf=energy                !save energy in variable e_oh_inf

text,separate calculation for Ar
geometry={AR}                  !geometry for OH alone
hf                             !scf for Ar
ccsd(t)                        !CCSD(T) for Ar
e_ar_inf=energy                !save energy in variable e_ar_inf

de=(e_ohar-e_oh_inf-e_ar_inf)*tocm    !compute uncorrected interaction energy
de_cp=(e_ohar-e_oh-e_ar)*tocm         !compute counter-poise corrected interaction energy
bsse_oh=(e_oh-e_oh_inf)*tocm          !BSSE for OH
bsse_ar=(e_ar-e_ar_inf)*tocm          !BSSE for Ar
bsse_tot=bsse_oh+bsse_ar              !total BSSE
```

http://www.molpro.net/info/current/examples/ohar_bsse.com

For performing counterpoise corrected geometry optimizations see section 42.4.7.

11 BASIS INPUT

11.1 Overview: sets and the basis library

Basis functions are used in Molpro not just for representing orbitals, but also for providing auxiliary sets for density fitting (see 15) and for simplifying integrals through approximate identity resolution in explicitly-correlated methods (see 31). In order to accommodate this, the program maintains internally a number of different *sets*. The first of these always has the name `ORBITAL` and is the primary basis set for representing orbitals, and others can be defined as necessary as described below, or else are constructed automatically by the program when required. In the latter case, the density-fitting and other modules attempt to guess a reasonable libary fitting basis that should be appropriate for the orbital basis set; it is advisable to check the choice when using anything other than a standard orbital basis set.

The basis sets may either be taken from the library, or may be specified explicitly, or any combination. Optionally, the basis function type can be chosen using the `CARTESIAN` or `SPHERICAL` commands.

11.2 Cartesian and spherical harmonic basis functions

MOLPRO uses spherical harmonics ($5d$, $7f$, etc) by default, even for Pople basis sets like `6-31G**`. This behaviour may be different to that of other programs; However, cartesian functions can be requested using the `CARTESIAN` command.

`CARTESIAN`

If this command is encountered, the logical MOLPRO variable `CARTESIAN` is set to true (1.0), and all subsequent calculations use cartesian basis functions. This is remembered across restarts. One can switch back to spherical harmonics using the command

`SPHERICAL`

11.3 The basis set library

The basis set library consists of a set of plain text files, together with an associated index, that constitute a database of commonly-used basis sets (primitive gaussians and associated contractions) and effective core potentials. These files can be found in the source tree as `lib/*.libmol` and `lib/libmol.index`, but it is usually more convenient to query the database using one of the provided tools.

Many of the basis sets are taken directly from the Pacific Northwest National Laboratory basis set database, but there are others, notably the Stuttgart effective core potentials and bases.

A simple command-line interface to the database is provided through the `libmol` program. It requires the environment variable `LIBMOL` to point to the `lib/` directory, but this will default to the location of the source tree at compile time, so it is often not necessary to specify it. The command-line syntax is

`libmol [-p` *print*`] [-e` *element*`] [-k` *key*`] [-t` *type*`] [-f` *format*`]`

where the parameters are

print:	Output level; 0 means list matching keys, 1 means print also the entry.
element:	Specify chemical element. If omitted, all elements are searched.
key:	Specify record key. If omitted, all keys are searched.
type:	Specify entry type, i.e. $s, p, \ldots$. If omitted, all types are searched.
format:	One of `text` (default), `molpro` (MOLPRO input format), `table` (tabular) or `html` (html table) to govern the output format.

A more convenient way of browsing the basis library is through the web interface at http://www.molpro.net/info/basis.

11.4 Default basis sets

If a basis is not specified at all for any unique atom group, then the program assumes a global default. Presently, this default is `VDZ`, but may be overridden using

`BASIS`,*basis*

or

`BASIS`=*basis*

basis is looked up in the file `lib/defbas`, which generates an appropriate request for a complete contracted set, together in some cases with an ECP, from the library. This mapping includes the following commonly-used basis sets:

- All of the Dunning correlation consistent basis sets, through the use of either the standard name of the basis set (`cc-pVXZ`, `aug-cc-pVXZ`) or an abbreviation (`VXZ`, `AVXZ`). For Al-Ar the tight-d augmented sets are obtained through the standard name `cc-pV(X+d)Z`, `aug-cc-pV(X+d)Z` or `VXZ+d`, `AVXZ+d`. Sets X=D,T,Q,5 are available for H-Kr with X=6 available for B-Ne and Al-Ar.
- The correlation consistent basis sets for core correlation, `cc-pCVXZ`, `aug-cc-pCVXZ` or `CVXZ`, `ACVXZ` (X=D,T,Q,5), and the newer ”weighted sets” `cc-pwCVXZ`, `aug-cc-pwCVXZ` or `WCVXZ`, `AWCVXZ` (X=D,T,Q,5). These are available for Li-Kr (CVXZ do not include Sc-Zn).
- Douglas-Kroll-Hess relativistic versions of the correlation consistent basis sets are available through use of the standard or abbreviated names with extension -DK, e.g., `cc-pVXZ-DK` or `VXZ-DK`. X=D-5 are available for H-Kr, while X=T are available for Y-Cd and Hf-Hg. Sets contracted for 3rd-order DKH are available for Hf-Hg with extension -DK3.
- The F12 basis sets of Peterson et al. for explicitly correlated calculations, `cc-pVXZ-F12`, `cc-pCVXZ-F12` or `VXZ-F12`, `CVXZ-F12` with X=D,T,Q. These are available for H-Ar.
- The Turbomole def2 family of basis sets, `SV(P)`, `SVP`, `TZVP`, `TZVPP`, `QZVP`, and `QZVPP`. These are available for the entire periodic table except for the f-block elements.
- The older segmented Dunning/Hay double-zeta sets for the first row (`DZ` and `DZP`).
- The Roos ANO basis sets for H-Ar (`ROOS`).
- The Stuttgart ECPs and associated basis sets (e.g., `ECP10MDF`), as well as the ECP-based correlation consistent basis sets of Peterson and co-workers, `cc-pVXZ-PP`, `aug-cc-pVXZ-PP`, `cc-pwCVXZ-PP`, `aug-cc-pwCVXZ-PP` or `VXZ-PP`, `AVXZ-PP`, `WCVXZ-PP`, `AWCVXZ-PP`. The latter are available for Cu-Kr, Y-Xe, and Hf-Rn (core correlation sets currently only for transition metals).
- The Hay ECPs and corresponding basis sets (`ECP1` and `ECP2`).
- Other members of the Karslruhe basis sets (`SV`, `TZV`, and, for some elements, `TZVPPP`).
- The Binning/Curtiss sets for Ga–Kr (`BINNING-SV`, `BINNING-SVP`, `BINNING-VTZ` and `BINNING-VTZP`)

- Most of the Pople basis sets, using their standard names (e.g., `6-31G*, 6-311++G(D,P)`, etc.). Note that specially in this case, the mechanism described below using parenthesized modifiers to restrict the basis set is disabled to allow the full range of standard basis sets to be specified.

In addition, many density fitting and resolution of the identity (RI) basis sets are available. For the correlation consistent basis sets of Dunning, the appropriate VXZ/JKFIT, VXZ/MP2FIT, AVXZ/MP2FIT sets of Weigend are chosen automatically in density fitted calculations (augmented versions AVXZ/JKFIT for Fock-matrix fitting are also available, but not used by default). For the def2 family of orbital basis sets, the appropriate auxiliary sets (e.g., TZVPP/JFIT, TZVPP/JKFIT, TZVPP/MP2FIT) are used. In principle these JKFIT sets are universal and also applicable in combination with the AVXZ basis sets. Initial results indicate that they also work well with the cc-pVXZ-PP and aug-cc-pVXZ-PP series of basis sets.

For explicitly correlated F12 calculations that use the cc-pVXZ-F12 orbital basis sets, the corresponding VXZ-F12/OPTRI basis sets are used by default to construct the complementary auxiliary orbital basis (CABS). For other orbital basis sets, appropriate JKFIT sets are utilized by default.

Example:

```
BASIS=VTZ
```

generates valence triple zeta basis set for all atoms. Thus, the input

```
***,h2o cc-pVTZ basis      !A title
r=1.85,theta=104           !set geometry parameters
geometry={O;               !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
basis=VTZ                  !use VTZ basis
hf                         !closed-shell scf
```

http://www.molpro.net/info/current/examples/h2o_scf_vtz.com

performs a Hartree-Fock calculation for H_2O using the cc-pVTZ basis set.

Default basis sets can be defined anywhere in the input before the energy calculation to which it should apply using a single `BASIS` card as shown above. The default basis set applies to all types of atoms but can be superceded by different basis sets for specific atoms as explained in the next section. Some restrictions concerning the maximum angular momentum functions to be used, or the number of contracted functions are possible as follows:

The maximum angular momentum in the basis set can be reduced using syntax such as

```
BASIS,VQZ(D)
```

which would omit the f and g functions that would normally be present in the VQZ basis set.

```
BASIS,VQZ(D/P)
```

would specify additionally a maximum angular momentum of 1 on hydrogen, i.e. would omit d orbitals on hydrogen.

For generally contracted basis sets, an extended syntax can be used to explicitly give the number of contracted functions of each angular momentum. For example,

```
BASIS,ROOS(3s2p1d/2s)
```

generates a `6-31G*`-sized basis set from the Roos ANO compilation.

11.5 Default basis sets for individual atoms

Different default basis sets for individual atoms can be specified one one-line `BASIS` commands by adding after the default basis *atom1=name1, atom2=name2,...*, where *atomi* are the chemical symbols, and *namei* are the associated basis set names. The name conventions for the atom specific basis sets work exactly as described above for default basis sets. Examples:

```
basis=vtz,h=vdz
```

uses cc-pVTZ as a general default, but for hydrogen atoms cc-pVDZ is used.

```
basis,vtz,h=vdz
```

or

```
basis,default=vtz,h=vdz
```

are equivalent to the above. Note that the default basis has to be specified before any atom specific sets.

11.6 Basis blocks

More specific basis set definitions for individual atoms can be given `BASIS` input blocks, which have the following general form:

```
BASIS
SET,setname1,[options]
DEFAULT=name
atom1=name1
atom2=name2
primitive basis set specifications
SET,setname2,[options]
...
END
```

Instead of the `BASIS` ... `END` block one can also use the structure `BASIS={...}`

Any number of basis sets can be be given in a basis block. The definition of each basis sets is started by a `SET` directive, on which the name of the basis and further options can be specified.

By default, the first set in a basis block is the orbital basis, and in this case the directive `SET,ORBITAL` can be omitted.

`DEFAULT` specifies the default basis set, exactly as on one line basis input. It can be followed by specifications for individual atoms, e.g. O=AVTZ. The default and atom specifications can also be merged to one line, separated by commas, e.g.

```
DEFAULT=VTZ,O=AVTZ,H=VDZ
```

Here the basis sets `AVTZ`, `VDZ` overwrite the default basis set VTZ for the atoms O and H, respectively. This is exactly as described in section 11.5 for one-line basis inputs.

The specifications `SET`, `DEFAULT`, `atom=name` are all optional. If `DEFAULT` is not given, the previous default, as specified on the last previous `BASIS` card, is used.

Several `BASIS` cards and/or blocks can immediately follow each other. Always the last specification for a given atom and setname is used (the default setname is `ORBITAL`).

If a basis is not specified at all for any unique atom group, then the program assumes VDZ.

11.7 Auxiliary basis sets for density fitting or resolution of the identity

As described in the previous section, several basis sets can be defined in a basis block. The definition of each basis starts with a line

`SET`,*name*,[`CONTEXT=`*context*]

where *name* is an arbitary name that can be used later to choose the basis set using options like `df_basis=`*name*, `ri_basis=`*name* etc. `CONTEXT` can optionally be specified to select the basis types `JFIT`, `JKFIT`, `MP2FIT`, `CCSDFIT`, or `OPTRI`. This affects the choice of default basis sets. For example

```
basis={
default=avtz    !default orbital basis set
set,df,context=mp2fit
default,avtz    !use avtz/mp2fit
set,jk,context=jkfit
default,avtz    !use avtz/jkfit
}
```

is equivalent to

```
basis={
default=avtz           !default orbital basis set
set,df
default,avtz/mp2fit    !use avtz/mp2fit
set,jk
default,avtz/jkfit     !use vtz/jkfit
}
```

If the setname begins with `JFIT`, `JKFIT`, `MP2FIT`, `CCSDFIT`, or `OPTRI`, these strings define the default context.

Specific basis sets for individual atoms or explicit input of exponents and contraction coefficients can be given exactly in the same way as for orbital basis sets.

11.8 Primitive set definition

Default basis sets given using one-line `BASIS` commands or `DEFAULT` directives in a basis block can be overwritten by explicit specifications of basis functions (type, exponents, contraction coefficients).

A group of basis functions is defined by a data card specifying a set of primitive gaussians, optionally followed by one or more cards specifying particular contractions of primitives to be included in the final basis (see section 11.9 for specification of contractions).

If an individual basis function type (s, p, d, etc.) is specified for an atom, it is required that all other types are also defined, i.e., as soon as an explicit definition of a basis function for an atom is given, all defaults are erased for this atom.

There are four different input forms for basis functions, as explained below under a) to d). In case that options (e.g. `SCALE`, `NPRIM`) are specified, they can be given in any order, but no value without option key must be given after an option.

In all four cases *type* defines the angular symmetry (S, P, D, F, G, H, or I). *type* can include several types, e.g., SPD or DF (this usually makes sense only with or default library contractions or no contractions). The basis is loaded for all atoms with tag name *atom* in the geometry input. If *atom* is an integer, it refers to a z-matrix row.

a) Library basis sets:

type,atom,name,scale2,nprim;

or

type,atom,name,[SCALE=*scale*|SCALE2=*scale2*],[NPRIM=*nprim*|DELETE=*ndel*];

Load basis named *name* from the library

If *scale* or *scale2* is present, all exponents are scaled by *scale* or *scale***2, respectively. If *nprim* is specified, the first *nprim* exponents only are taken from the library. If *nprim* is negative or *ndel* is given, the last $|nprim|$ (*ndel*) basis functions from the library set are deleted. Associated with the library basis may be a set of default contraction coefficients which may be accessed in subsequent contraction cards. *type* can include several types, e.g., SPD or DF. This usually makes sense only with default contractions, i.e., such cards should be followed only by "C" without any other specifications for contractions.

b) Explicit basis input:

type,atom,exp1,exp2,...expn;expn+1,...;

General specification of exponents; continuation onto subsequent cards (separated by semicolon) is permitted as shown (the first card can hold up to 19 exponents, each following card 20 exponents.

The exponents (and other numerical parameters described below such as numbers of functions, and contraction coefficients) can be given as general input expressions, possibly involving variables. It is important to note, however, that these expressions are evaluated typically just once, at the same time as the complete basis set is parsed. This generally happens the first time that the basis set is required, perhaps before the first SCF calculation can be done. If the variables on which the basis depends are altered, this will not be noticed by the program, and the new basis set will not be used for subsequent stages of the computation. If, however, a new basis block is presented in the input, then the program marks as outdated any quantities such as integrals that have been calculated with the old basis set; subsequent job steps will then use the new basis.

c) Even tempered basis sets:

type,atom,EVEN,*nprim,ratio,centre,dratio*

or

type,atom,EVEN,NPRIM=*nprim*,[RATIO=*ratio*],[CENTRE=*centre*],[DRATIO=*dratio*]

Generates a generalized even tempered set of functions. The number of functions n is specified by *nprim*, their geometric mean c by *centre*, the mean ratio of successive exponents r by *ratio*, and the variation of this ratio, d, by *dratio*. If *centre* is not given, the previous basis of the same type is extended by diffuse functions. If in this case *ratio* is not given, r is determined from the exponents of the last two function of the previous basis. If this is not possible, the default $r = 2.5$ is adopted. $d = 1$ (the default) specifies a true even-tempered set, but otherwise the ratio between successive exponents changes linearly; the exponents are given explicitly by

$$\log e_i = \log c + ((n+1)/2 - i)\,\log r + \frac{1}{2}((n+1)/2 - i)^2\,\log d \quad i = 1,2,\ldots,n$$

Example 1 `SP,1,VTZ;C;SP,1,EVEN,1;`
generates the generally contracted *s* and *p* triple-zeta basis sets for atom 1 and extends these by one diffuse function.

Example 2 `SPD,1,VTZ,DELETE=1;C;`
`SP,1,EVEN,NPRIM=2,RATIO=2.5;`
generates the generally contracted *s*, *p* triple-zeta basis sets for atom 1. Two energy optimized *d*-functions of Dunning are included. The last *s* and *p* functions are deleted and replaced by two even tempered functions with ratio 2.5.

d) 3-term tempered basis sets:

type,atom,`EVEN3`*,nprim,*α*,* β*,* γ

Generates a 3-parameter set of *nprim* functions with exponents given by

$$e_i = \alpha; \qquad e_i = e_{i-1}\beta\left(1 + \frac{\gamma i^2}{(nprim+1)^2}\right)$$

e) Regular even tempered basis sets:

type,atom,`EVENR`*,nprim,aa,ap,bb,bp*

Generates an even tempered set of *nprim* functions according to the "regular" prescription described in M W Schmidt and K Ruedenberg, J. Chem. Phys. 71 (1970) 3951. If any of the parameters *aa, ap, bb, bp* is zero or omitted, the values are taken from table III of the above.

f) Even tempered basis set with confined progression:

type,atom,`EVENP`*,nprim,*α,β,γ

Generates an even tempered basis set with *nprim* functions and a maximal exponent given by α. The progression (ratio) between the first and second exponent is adjusted using parameter β and the progression between the last but one and the last exponent is adjusted with parameter γ. In between the progression is linearly interpolated. The explicit values of the progression factors are given by:

$p(\beta) = \frac{\text{exponent}^{\text{i}}}{\text{exponent}^{\text{i+1}}} = \frac{5}{\pi}(\arctan(\beta - 2.5) + \frac{\pi}{2}) + \sqrt{2}$

so that for $\beta \ll 0 : p \to \sqrt{2}$ and for $\beta \gg 0 : p \to 5 + \sqrt{2}$ which limits the progression factors in between these two values and enables unconstrained basis set optimisations. For $\beta \approx 0$ the progression has a factor of about 2.

type,atom,`EVENP2`*,nprim,*$\alpha,\beta,\gamma,\delta$

Generalises confined progression tempered basis sets by a third paramter (now γ) which defines the progression as above in the centre. The ratio factors are then determined by interpolating between $p(\beta) \to p(\gamma)$ and $p(\gamma) \to p(\delta)$.

11.9 Contracted set definitions

a) `C`*,first.last,c1,c2,...cn;cn+1,...*;

General specification of a contracted function. *first.last* defines the range of primitives to be contracted. The order corresponds to the primitives as specified on the previous input card. *c1,* *c2*... are the $last - first + 1$ contraction coefficients. Continuation onto a subsequent card is permitted as shown.

b) `C`;

Use default contractions from the library. This applies to both the number of contracted primitives and also to the number of different contraction sets.

c) *n*`C`,*first.last*;

n contracted functions taken from library. *first.last* defines the range of primitives to be contracted. If *n* is omitted and *first.last* is specified, $n = 1$. If *first.last* is omitted, the library default values are used. If both *n* and *first.last* are omitted, default values for both are used.

d) *n*`C`,*first.last,record.file,orb.sym*;

n contracted functions taken from orbitals *orb*, $orb+1$,..,$orb+n-1$ of symmetry *sym* on molpro file *record.file*. The first nonzero coefficient in the specified orbital corresponds to the first associated basis function. *first.last* specifies the range of primitives to be contracted. If *first.last* is omitted, all coefficients from the specified orbitals are used.

Example `2C,1.12,2100.2,1.1`
generates two contractions, using the first 12 coefficients from orbitals 1.1 and 2.1. The orbitals are read from record 2100.2.

11.10 Examples

This shows the use of default basis sets for H_2O:

```
***,H2O
basis=VQZ(f/p)
R=0.95 ANG,THETA=104 DEGREE
geometry={O;H1,O,R;H2,O,R,H1,THETA}
hf              !do closed-shell SCF
```

http://www.molpro.net/info/current/examples/h2o_vqz_fp.com

This is equivalent to the explicit input form

```
***,H2O
R=0.95 ANG,THETA=104 DEGREE
geometry={O;H1,O,R;H2,O,R,H1,THETA}
basis={spdf,o,vqz;c;sp,h,vqz,c;}
hf              !do closed-shell SCF
```

http://www.molpro.net/info/current/examples/h2o_vqz_fp_explicit.com

This is an example for using multiple basis sets for density fitting and resolution of the identity

```
***,h2o
geom={o;
      h1,o,r;
      h2,o,r,h1,theta}
r=0.97 ang
theta=104

basis={
default,avtz
set,df
default,avtz/mp2fit   !density fitting basis
set,jk
default,avtz/jkfit    !density fitting basis for Fock and exchange matrices
set,ri
default,avtz/optri    !ri cabs basis
}
hf
ccsd(t)-f12,df_basis=df,df_basis_exch=jk,ri_basis=ri
```

http://www.molpro.net/info/current/examples/h2o_basissets1.com

The following two examples yield identical results:

```
***,h2o
geom={o;
      h1,o,r;
      h2,o,r,h1,theta}
r=0.97 ang
theta=104

basis={
default,avtz
set,df,context=mp2fit
default,avtz          !density fitting basis
set,jk,context=jkfit
default,avtz          !density fitting basis for Fock and exchange matrices
set,ri,context=optri
default,avtz          !ri cabs basis
}
hf
ccsd(t)-f12,df_basis=df,df_basis_exch=jk,ri_basis=ri
```

http://www.molpro.net/info/current/examples/h2o_basissets2.com

```
***,h2o
geom={o;
      h1,o,r;
      h2,o,r,h1,theta}
r=0.97 ang
theta=104

basis=avtz
hf
ccsd(t)-f12,df_basis=avtz/mp2fit,df_basis_exch=avtz/jkfit,ri_basis=avtz/optri
```

http://www.molpro.net/info/current/examples/h2o_basissets3.com

In the latter example, the speciations mp2fit and jkfit, respectively, can be omitted since these contexts are defaults for df_basis and df_basis_exch, respectively.

12 EFFECTIVE CORE POTENTIALS

Pseudopotentials (effective core potentials, ECPs) may be defined at the beginning of `BASIS` blocks.

The general form of the input cards is

`ECP`,*atom*,[*ECP specification*]

which defines a pseudopotential for an atom specified either by a chemical symbol or a group number. The *ECP specification* may consist either of a single keyword, which references a pseudopotential stored in the library, or else of an explicit definition (extending over several input cards), cf. below.

12.1 Input from ECP library

The basis set library presently contains the pseudopotentials and associated valence basis sets by a) the Los Alamos group (P. J. Hay and W. R. Wadt, J. Chem. Phys. **82**, 270 (1985) and following two papers), and b) the Stuttgart/Köln group (e.g., A. Nicklass, M. Dolg, H. Stoll and H. Preuß, J. Chem. Phys. **102**, 8942 (1995); for more details and proper references, see the web page `http://www.theochem.uni-stuttgart.de/pseudopotentials/`). Pseudopotentials a) are adjusted to orbital energies and densities of a suitable atomic reference state, while pseudopotentials b) are generated using total valence energies of a multitude of atomic states.

Library keywords in case a) are `ECP1` and `ECP2`; `ECP2` is used when more than one pseudopotential is available for a given atom and then denotes the ECP with the smaller core definition. (For Cu, e.g., `ECP1` refers to an Ar-like $18e^-$-core, while `ECP2` simulates a Ne-like $10e^-$ one with the $3s$ and $3p$ electrons promoted to the valence shell). For accurate calculations including electron correlation, promotion of all core orbitals with main quantum number equal to any of the valence orbitals is recommended.

Library keywords in case b) are of the form `ECP`nXY; n is the number of core electrons which are replaced by the pseudopotential, X denotes the reference system used for generating the pseudopotential ($X = S$: single-valence-electron ion; $X = M$: neutral atom), and Y stands for the theoretical level of the reference data ($Y = HF$: Hartree-Fock, $Y = WB$: quasi-relativistic; $Y = DF$: relativistic). For one- or two-valence electron atoms $X = S$, $Y = DF$ is a good choice, while otherwise $X = M$, $Y = WB$ (or $Y = DF$) is recommended. (For light atoms, or for the discussion of relativistic effects, the corresponding $Y = HF$ pseudopotentials may be useful.) Additionally, spin-orbit (SO) potentials and core-polarization potentials (CPP) are available, to be used in connection with case b) ECPs, but these are not currently contained in the library, so explicit input is necessary here (cf. below).

In both cases, a) and b), the same keywords refer to the pseudopotential and the corresponding basis set, with a prefix `MBS-`... in case a).

12.2 Explicit input for ECPs

For each of the pseudopotentials the following information has to be provided:

- a card of the form
  `ECP`,*atom*,$n_{core}, l_{max}, l'_{max}$;
  where n_{core} is the number of core electrons replaced by the pseudopotential V_{ps}, l_{max}

is the number of semi-local terms in the scalar-relativistic part of V_{ps}, while l'_{max} is the corresponding number of terms in the SO part:

$$V_{ps} = -\frac{Z - n_{core}}{r} + V_{l_{max}} + \sum_{l=0}^{l_{max}-1} (V_l - V_{l_{max}})\mathscr{P}_l + \sum_{l=1}^{l'max} \Delta V_l \mathscr{P}_l \vec{l} \cdot \vec{s} \mathscr{P}_l;$$

the semi-local terms (with angular-momentum projectors $\mathscr{P}_l$) are supplemented by a local term for $l = l_{max}$.

- a number of cards specifying $V_{l_{max}}$, the first giving the expansion length $n_{l_{max}}$ in

$$V_{l_{max}} = \sum_{j=1}^{n_{l_{max}}} c_j r^{m_j-2} e^{-\gamma_j r^2}$$

and the following $n_{l_{max}}$ ones giving the parameters in the form

$$m_1, \gamma_1, c_1; m_2, \gamma_2, c_2; \ldots$$

- a number of cards specifying the scalar-relativistic semi-local terms in the order $l = 0, 1, \ldots, l_{max} - 1$. For each of these terms a card with the expansion length n_l in

$$V_l - V_{l_{max}} = \sum_{j=1}^{n_l} c_j^l r^{m_j^l-2} e^{-\gamma_j^l r^2}$$

has to be given, and immediately following n_l cards with the corresponding parameters in the form $m_1^l, \gamma_1^l, c_1^l; m_2^l, \gamma_2^l, c_2^l; \ldots$

- analogously, a number of cards specifying the coefficients of the radial potentials ΔV_l of the SO part of V_{ps}.

12.3 Example for explicit ECP input

```
***,CU
! SCF d10s1 -> d9s2 excitation energy of the Cu atom
!  using the relativistic Ne-core pseudopotential
! and basis of the Stuttgart/Koeln group.
gprint,basis,orbitals
geometry={cu}
basis
ECP,1,10,3;   ! ECP input
 1; ! NO LOCAL POTENTIAL
 2,1.,0.;
 2; ! S POTENTIAL
 2,30.22,355.770158;2,13.19,70.865357;
 2; ! P POTENTIAL
 2,33.13,233.891976;2,13.22,53.947299;
 2; ! D POTENTIAL
 2,38.42,-31.272165;2,13.26,-2.741104;
! (8s7p6d)/[6s5p3d] BASIS SET
s,1,27.69632,13.50535,8.815355,2.380805,.952616,.112662,.040486,.01;
c,1.3,.231132,-.656811,-.545875;
p,1,93.504327,16.285464,5.994236,2.536875,.897934,.131729,.030878;
c,1.2,.022829,-1.009513;C,3.4,.24645,.792024;
d,1,41.225006,12.34325,4.20192,1.379825,.383453,.1;
c,1.4,.044694,.212106,.453423,.533465;
end
rhf;
e1=energy
{rhf;occ,4,1,1,1,1,1,1;closed,4,1,1,1,1,1;wf,19,7,1;}
e2=energy
de=(e2-e1)*toev  ! Delta E = -0.075 eV
```

http://www.molpro.net/info/current/examples/cu_ecp_explicit.com

12.4 Example for ECP input from library

```
***,AuH
! CCSD(T) binding energy of the AuH molecule at r(exp)
! using the scalar-relativistic 19-valence-electron
! pseudopotential of the Stuttgart/Koeln group
gprint,basis,orbitals;
geometry={au}
basis={
ecp,au,ECP60MWB;              ! ECP input
spd,au,ECP60MWB;c,1.2;        ! basis set
f,au,1.41,0.47,0.15;
g,au,1.2,0.4;
spd,h,avtz;c;
}
rhf;
{rccsd(t);core,1,1,1,,1;}
e1=energy
geometry={h}
rhf
e2=energy;
rAuH=1.524 ang                ! molecular calculation
geometry={au;h,au,rAuH}
hf;
{ccsd(t);core,2,1,1;}
e3=energy
de=(e3-e2-e1)*toev            ! binding energy = 3.11 eV
```

http://www.molpro.net/info/current/examples/auh_ecp_lib.com

13 CORE POLARIZATION POTENTIALS

13.1 Input options

The calculation of core-polarization matrix elements is invoked by the CPP card, which can be called at an arbitrary position in the MOLPRO input, provided the integrals have been calculated before. The CPP card can have the following three formats:

- CPP,INIT,*ncentres*;
- CPP,ADD[,*factor*];
- CPP,SET[,*fcpp*];

CPP,INIT,$< ncentres >$;

abs($< ncentres >$) further cards will be read in the following format:

$< atomtype >, < ntype >, < \alpha_d >, < \alpha_q >, < \beta_d >, < cutoff >$;

$< atomtype >$ corresponds to the recognition of the atomic centres in the integral part of the program,
$< ntype >$ fixes the form of the cutoff-function (choose 1 for Stoll/Fuentealba and 2 for Mueller/Meyer);
$< \alpha_d >$ is the static dipole polarizability,
$< \alpha_q >$ is the static quadrupole polarizability,
$< \beta_d >$ is the first non-adiabatic correction to the dipole-polarizability and

$< cutoff >$ is the exponential parameter of the cutoff-function.

When $< ncentres >$ is lower than zero, only the integrals are calculated and saved in the record 1490.1. Otherwise, the h_0 matrix (records 1200.1 and 1210.1) and the two-electron-integrals (record 1300.1) will be modified.

CPP,ADD,$< factor >$;

With this variant, previously calculated matrix elements of the polarization matrix can be added with the variable factor $< factor >$ (default: $< factor >$ = 1) to the h_0-matrix as well as to the two-electron-integrals. In particular, CPP,ADD,-1.; can be used to retrieve the integrals without the polarization contribution.

CPP,SET,$< fcpp >$;

normally not necessary but may be used to tell MOLPRO after a restart, with what factor the polarization integrals are effective at the moment. Currently the CPP integrals are restricted to basis functions up to and including angular momentum 4, i.e. g functions.

13.2 Example for ECP/CPP

```
***,Na2
! Potential curve of the Na2 molecule
! using 1-ve ECP + CPP
gprint,basis,orbitals;
rvec=[2.9,3.0,3.1,3.2,3.3] ang
do i=1,#rvec
rNa2=rvec(i)
geometry={na;na,na,rNa2}
basis={
ecp,na,ecp10sdf;         ! ecp input
s,na,even,8,3,.5;        ! basis input
p,na,even,6,3,.2;
d,na,.12,.03;
}
cpp,init,1;              ! CPP input
na,1,.9947,,,.62;
hf;
ehf(i)=energy
{cisd;core;}
eci(i)=energy
enddo
table,rvec,ehf,eci
---
```

http://www.molpro.net/info/current/examples/na2_ecp_cpp.com

14 INTEGRATION

Before starting any energy calculations, the geometry and basis set must be defined in `GEOMETRY` and `BASIS` blocks, respectively. By default, two electron integrals are evaluated once and stored on disk. This behaviour may be overridden by using the input command `gdirect` (see section 14.2) to force evaluation of integrals on the fly. MOLPRO checks if the one-and two-electron integrals are available for the current basis set and geometry, automatically computing them if necessary. The program also recognizes automatically if only the nuclear charges have been changed, as is the case in counterpoise calculations. In this case, the two-electron integrals are not recomputed.

14.1 Sorted integrals

If the integrals are stored on disk, immediately after evaluation they are sorted into complete symmetry-packed matrices, so that later program modules that use them can do so as efficiently as possible. As discussed above, it is normally not necessary to call the integral and sorting programs explicitly, but sometimes additional options are desired, and can be specified using the `INT` command, which should appear after geometry and basis specifications, and before any commands to evaluate an energy.

`INT`, [`[NO]SORT`,] [`SPRI=`*value*]

`SORT`, [`SPRI=`*value*]

`INT,NOSORT;SORT` can be used to explicitly separate the integral evaluation and sorting steps, for example to collect separate timing data. With *value* set to more than 1 in the `SPRI` option, all the two-electron integrals are printed.

The detailed options for the integral sort can be specified using the `AOINT` parameter set, using the input form

`AOINT`, *key1=value1*, *key2=value2*, ...

`AOINT` can be used with or without an explicit `INT` command.

The following summarizes the possible keys, together with their meaning, and default values.

c_final	Integer specifying the compression algorithm to be used for the final sorted integrals. Possible values are 0 (no compression), 1 (compression using 1, 2, 4 or 8-byte values), 2 (2, 4 or 8 bytes), 4 (4, 8 bytes) and 8. Default: 0
c_sort1	Integer specifying the compression algorithm for the intermediate file during the sort. Default: 0
c_seward	Integer specifying the format of label tagging and compression written by the integral program and read by the sort program. Default: 0
compress	Overall compression; c_final, c_seward and c_sort1 are forced internally to be not less than this parameter. Default: 1
thresh	Real giving the truncation threshold for compression. Default: 0.0, which means use the integral evaluation threshold (`GTHRESH,TWOINT`)
io	String specifying how the sorted integrals are written. Possible values are `molpro` (standard MOLPRO record on file 1) and `eaf` (Exclusive-access file). `eaf` is permissible only if the program has been configured for MPP usage, and at present `molpro` is implemented only for serial execution. `molpro` is required if the integrals are to be used in a restart job. For maximum efficiency on a parallel machine, `eaf` should be used, since in that case the integrals are distributed on separate processor-local files.

For backward-compatibility purposes, two convenience commands are also defined: `COMPRESS` is equivalent to `AOINT,COMPRESS=1`, and `UNCOMPRESS` is equivalent to `AOINT,COMPRESS=0`.

14.2 INTEGRAL-DIRECT CALCULATIONS (GDIRECT)

References:

Direct methods, general: M. Schütz, R. Lindh, and H.-J. Werner, Mol. Phys. 96, 719 (1999).
Linear scaling LMP2: M. Schütz, G. Hetzer, and H.-J. Werner J. Chem. Phys. **111**, 5691 (1999).

All methods implemented in MOLPRO apart from full CI (FCI) and perturbative triple excitations (T) can be performed integral-direct, i.e., the methods are integral driven with the two-electron integrals in the AO basis being recomputed whenever needed, avoiding the bottleneck of storing these quantities on disk. For small molecules, this requires significantly more CPU time, but reduces the disk space requirements when using large basis sets. However, due to efficient prescreening techniques, the scaling of the computational cost with molecular size is lower in integral-direct mode than in conventional mode, and therefore integral-direct calculations for extended molecules may even be less expensive than conventional ones. The break-even point depends strongly on the size of the molecule, the hardware, and the basis set. Depending on the available disk space, calculations with more than 150–200 basis functions in one symmetry should normally be done in integral-direct mode.

Integral-direct calculations are requested by the DIRECT or GDIRECT directives. If one of these cards is given outside the input of specific programs it acts globally, i.e. all subsequent calculations are performed in integral-direct mode. On the other hand, if the DIRECT card is part of the input of specific programs (e.g. HF, CCSD), it affects only this program. The GDIRECT directive is not recognized by individual programs and always acts globally. Normally, all calculations in one job will be done integral-direct, and then a DIRECT or GDIRECT card is required before the first energy calculation. However, further DIRECT or GDIRECT directives can be given in order to modify specific options or thresholds for particular programs.

The integral-direct implementation in MOLPRO involves three different procedures: (i) Fock matrix evaluation (DFOCK), (ii) integral transformation (DTRAF), and (iii) external exchange operators (DKEXT). Specific options and thresholds exist for all three programs, but it is also possible to specify the most important thresholds by general parameters, which are used as defaults for all programs.

Normally, appropriate default values are automatically used by the program, and in most cases no parameters need to be specified on the DIRECT directive. However, in order to guarantee sufficient accuracy, the default thresholds are quite strict, and in calculations for extended systems larger values might be useful to reduce the CPU time.

The format of the DIRECT directive is

DIRECT, key1=*value1*, key2=*value2*. . .

The following table summarizes the possible keys and their meaning. The default values are given in the subsequent table. In various cases there is a hierarchy of default values. For instance, if THREST_D2EXT is not given, one of the following is used: [THR_D2EXT, THREST_DTRAF, THR_DTRAF, THREST, *default*]. The list in brackets is checked from left to right, and the first one found in the input is used. *default* is a default value which depends on the energy threshold and the basis set (the threshold is reduced if the overlap matrix contains very small eigenvalues).

General Options (apply to all programs):

THREST	Integral prescreening threshold. The calculation of an integral shell block is skipped if the product of the largest estimated integral value (based on the Cauchy-Schwarz inequality) and the largest density matrix element contributing to the shell block is

	smaller than this value. In DTRAF and DKEXT effective density matrices are constructed from the MO coefficients and amplitudes, respectively.
THRINT	Integral prescreening threshold. This applies to the product of the exact (i.e. computed) integral value and a density matrix. This threshold is only used in DTRAF and DKEXT. A shell block of integrals is skipped if the product of the largest integral and the largest element of the effective density matrix contributing to the shell block is smaller than this threshold. If it set negative, no computed integrals will be neglected.
THRPROD	Prescreening threshold for products of integrals and MO-coefficients (DTRAF) or amplitudes (DKEXT). Shell blocks of MO coefficients or amplitudes are neglected if the product of the largest integral in the shell block and the largest coefficient is smaller than this value. If this is set negative, no product screening is performed.
THRMAX	Initial value of the prescreening threshold THREST for DFOCK and DKEXT in iterative methods (SCF, CI, CCSD). If nonzero, it will also be used for DKEXT in MP3 and MP4(SDQ) calculations. The threshold will be reduced to THREST once a certain accuracy has been reached (see VARRED), or latest after MAXRED iterations. In CI and CCSD calculations, also the initial thresholds THRINT_DKEXT and THRPROD_DKEXT are influenced by this value. For a description, see THRMAX_DKEXT. If THRMAX=0, the final thresholds will be used from the beginning in all methods.
SCREEN	Enables or disables prescreening. SCREEN$\geq$ 0: full screening enabled. SCREEN$<$ 0: THRPROD is unused. No density screening in direct SCF. SCREEN< -1: THRINT is unused. SCREEN< -2: THREST is unused.
MAXRED	Maximum number of iterations after which thresholds are reduced to their final values in CI and CCSD calculations. If MAXRED=0, the final thresholds will be used in CI and CCSD from the beginning (same as THRMAX=0, but MAXRED has no effect on DSCF. In the latter case a fixed value of 10 is used.
VARRED	Thresholds are reduced to their final values if the sum of squared amplitude changes is smaller than this value.
SWAP	Enables or disables label swapping in SEWARD. Test purpose only.

Specific options for direct SCF (DFOCK):

THREST_DSCF	Final prescreening threshold in direct SCF. If given, it replaces the value of THREST.
THRMAX_DSCF	Initial prescreening threshold in direct SCF. This is used for the first 7-10 iterations. Once a certain accuracy is reached, the threshold is reduced to THREST_DSCF

SWAP_DFOCK — Enables or disables label swapping in fock matrix calculation (test purpose only).

General options for direct integral transformation (DTRAF):

PAGE_DTRAF — Selects the transformation method.
PAGE_DTRAF=0: use minimum memory algorithm, requiring four integral evaluations.
PAGE_DTRAF=1: use paging algorithm,leading to the minimum CPU time (one integral evaluation for DMP2/LMP2 and two otherwise).

SCREEN_DTRAF — If given, replaces value of SCREEN for DTRAF.

MAXSHLQ1_DTRAF — Maximum size of merged shells in the first quarter transformation step (0: not used).

MINSHLQ1_DTRAF — Shells are only merged if their size is smaller than this value (0: not used).

MAXSHLQ2_DTRAF — Maximum size of merged shells in the second quarter transformation step (0: not used).

MINSHLQ2_DTRAF — Shells are only merged if their size is smaller than this value (0: not used).

MAXCEN_DTRAF — Maximum number of centres in merged shells (0: no limit).

PRINT_DTRAF — Print parameter for DTRAF.

General thresholds for all direct integral transformations:

THR_DTRAF — General threshold for DTRAF. If given, this is taken as default value for all thresholds described below.

THREST_DTRAF — AO prescreening threshold for DTRAF.
Defaults: [THR_DTRAF, THREST, *default*].

THRINT_DTRAF — Integral threshold for DTRAF.
Defaults: [THR_DTRAF, THRINT, *default*].

THRPROD_DTRAF — Product threshold for DTRAF.
Defaults: [THR_DTRAF, THRPROD, *default*].

Thresholds specific to direct integral transformations:

THR_D2EXT — General threshold for generation of 2-external integrals. If given, this is used as a default for all D2EXT thresholds described below.

THREST_D2EXT — Prescreening threshold for generation of 2-external integrals.
Defaults: [THR_D2EXT, THREST_DTRAF, THR_DTRAF, THREST, *default*].

THRINT_D2EXT — Integral threshold for generation of 2-external integrals.
Defaults: [THR_D2EXT, THRINT_DTRAF, THR_DTRAF, THRINT, *default*].

THRPROD_D2EXT — Product threshold for generation of 2-external integrals.
Defaults: [THR_D2EXT, THRPROD_DTRAF, THR_DTRAF, THRPROD, *default*].

THR_D3EXT	General threshold for generation of 3-external integrals. If given, this is used as a default for all D3EXT thresholds described below.
THREST_D3EXT	Prescreening threshold for generation of 3-external integrals. Defaults: [THR_D3EXT, THREST_DTRAF, THR_DTRAF, THREST, *default*].
THRINT_D3EXT	Integral threshold for generation of 3-external integrals. Defaults: [THR_D3EXT, THRINT_DTRAF, THR_DTRAF, THRINT, *default*].
THRPROD_D3EXT	Product threshold for generation of 3-external integrals. Defaults: [THR_D3EXT, THRPROD_DTRAF, THR_DTRAF, THRPROD, *default*].
THR_D4EXT	General threshold for generation of 4-external integrals. If given, this is used as a default for all D4EXT thresholds described below.
THREST_D4EXT	Prescreening threshold for generation of 4-external integrals. Defaults: [THR_D4EXT, THREST_DTRAF, THR_DTRAF, THREST, *default*].
THRINT_D4EXT	Integral threshold for generation of 4-external integrals. Defaults: [THR_D4EXT, THRINT_DTRAF, THR_DTRAF, THRINT, *default*].
THRPROD_D4EXT	Product threshold for generation of 4-external integrals. Defaults: [THR_D4EXT, THRPROD_DTRAF, THR_DTRAF, THRPROD, *default*].
THR_DCCSD	General threshold for generalized transformation needed in each CCSD iteration. If given, this is used as a default for THREST_DCCSD, THRINT_DCCSD, and THRPROD_DCCSD described below.
THREST_DCCSD	Prescreening threshold for DCCSD transformation. Defaults: [THR_DCCSD, THREST_DTRAF, THR_DTRAF, THREST, *default*].
THRINT_DCCSD	Integral threshold for DCCSD transformation. Defaults: [THR_DCCSD, THRINT_DTRAF, THR_DTRAF, THRINT, *default*].
THRPROD_DCCSD	Product threshold for DCCSD transformation. Defaults: [THR_DCCSD, THRPROD_DTRAF, THR_DTRAF, THRPROD, *default*].
THRMAX_DCCSD	Initial value for THREST_DCCSD in CCSD calculations. The threshold will be reduced to THREST_DCCSD once a certain accuracy has been reached (see VARRED), or latest after MAXRED iterations. The initial thresholds THRINT_DCCSD and THRPROD_DCCSD are obtained by multiplying their input (or default) values by THRMAX_DCCSD/THREST_DCCSD, with the restriction that the initial values cannot be smaller than the final ones.

Specific options for direct MP2 (DMP2):

DMP2 — Selects the transformation method for direct MP2:
DMP2=−1: automatic selection, depending on the available memory.
DMP2=0: use fully direct method for DMP2 (min. two integral evaluations, possibly multipassing, no disk space).
DMP2=1: use semi-direct method for DMP2 (one to four integral evaluations, depending on PAGE_DTRAF).
DMP2=2: use DKEXT to compute exchange operators in DMP2 (one integral evaluation). This is only useful in local DMP2 calculations with many distant pairs.

THR_DMP2 — General threshold for generation of 2-external integrals in DMP2. If given, this is used as a default for all DMP2 thresholds described below.

THREST_DMP2 — Prescreening threshold for generation of 2-external integrals. Defaults: [THR_DMP2, THREST_DTRAF, THR_DTRAF, THREST, *default*].

THRINT_DMP2 — Integral threshold for generation of 2-external integrals. Defaults: [THR_DMP2, THRINT_DTRAF, THR_DTRAF, THRINT, *default*].

THRPROD_DMP2 — Product threshold for generation of 2-external integrals Defaults: [THR_DMP2, THRPROD_DTRAF, THR_DTRAF, THRPROD, *default*].

Specific options for direct local MP2 (LMP2):

DTRAF — Selects the transformation method for direct LMP2:
DTRAF ≥ 0: generates the 2-external integrals (exchange operators) first in AO basis and transforms these thereafter in a second step to the projected, local basis. The disk storage requirements hence scale cubically with molecular size.
DTRAF $= -1$: generates the 2-external integrals (exchange operators) directly in projected basis. The disk storage requirements hence scale linearly with molecular size. This (together with PAGE_DTRAF = 0) is the recommended algorithm for very large molecules (cf. linear scaling LMP2, chapter 29).
DTRAF $= -2$: alternative algorithm to generate the exchange operators directly in projected basis. Usually, this algorithm turns out to be computationally more expensive than the one selected with DTRAF $= -1$. Note, that neither DTRAF $= -1$ nor DTRAF $= -2$ work in the context of LMP2 gradients.

THR_LMP2 — General threshold for generation of 2-external integrals in linear scaling LMP2. If given, this is used as a default for all LMP2 thresholds described below.

THREST_LMP2 — Prescreening threshold for generation of 2-external integrals. Defaults: [THR_LMP2, THREST_DTRAF, THR_DTRAF, THREST, *default*].

THRQ1_LMP2 — Threshold used in the first quarter transformation. Defaults: [THR_LMP2, THRPROD_DTRAF, THR_DTRAF, THRPROD, *default*].

THRQ2_LMP2	Threshold used in the second and subsequent quarter transformations. Defaults: [THR_LMP2, THRINT_DTRAF, THR_DTRAF, THRINT, *default*].
THRAO_ATTEN	Special threshold for prescreening of attenuated integrals $(\mu\mu\|\nu\nu)$ Default: THREST_LMP2

Options for integral-direct computation of external exchange operators (DKEXT):

DKEXT	Selects driver for DKEXT. DKEXT=−1: use paging algorithm (minimum memory). This is automatically used if in-core algorithm would need more than one integral pass. DKEXT=0: use in-core algorithm, no integral triples. DKEXT=1: use in-core algorithm and integral triples. DKEXT=2: use in-core algorithm and integral triples if at least two integrals of a triple differ. DKEXT=3: use in-core algorithm and integral triples if all integrals of a triple differ.
SCREEN_DKEXT	if given, replaces value of SCREEN for DKEXT.
MAXSIZE_DKEXT	Largest size of merged shells in DKEXT (0: not used).
MINSIZE_DKEXT	Shells are only merged if their size is smaller than this value. (0: not used).
MAXCEN_DKEXT	Maximum number of centres in merged shells (0: no limit).
SCREEN_DKEXT	Enables of disables screening in DKEXT.
PRINT_DKEXT	Print parameter for DKEXT.
SWAP_DKEXT	Enables of disables label swapping in DKEXT (test purpose only)
MXMBLK_DKEXT	Largest matrix block size in DKEXT (only used with DKEXT≥ 1).

Thresholds for integral-direct computation of external exchange operators (DKEXT):

THR_DKEXT	General threshold for DKEXT. If given, this is used as a default for all DKEXT thresholds described below.
THREST_DKEXT	Prescreening threshold for DKEXT. Defaults: [THR_DKEXT, THREST, *default*].
THRINT_DKEXT	Integral threshold for DKEXT. Defaults: [THR_DKEXT, THRINT, *default*].
THRPROD_DKEXT	Product threshold for DKEXT. Defaults: [THR_DKEXT, THRPROD, *default*].
THRMAX_DKEXT	Initial value for THREST_DKEXT in CI, and CCSD calculations. If nonzero. it will also be used for DKEXT in MP3 and MP4(SDQ) calculations. The threshold will be reduced to THREST_DKEXT once a certain accuracy has been reached (see VARRED), or latest after MAXRED iterations. The initial thresholds THRINT_DKEXT and THRPROD_DKEXT are obtained by multiplying their input (or default) values by THRMAX_DKEXT/THREST_DKEXT,

with the restriction that the initial values cannot be smaller than the final ones.

For historical reasons, many options have alias names. The following tables summarize the default values for all options and thresholds and also gives possible alias names.

Table 6: Default values and alias names for `direct` options.

Parameter	Alias	Default value
SCREEN		1
MAXRED		7
VARRED		1.d-7
SWAP		1
SWAP_DFOCK		SWAP
DMP2	DTRAF	−1
PAGE_DTRAF	PAGE	1
SCREEN_DTRAF		SCREEN
MAXSHLQ1_DTRAF	NSHLQ1	32
MINSHLQ1_DTRAF		0
MAXSHLQ2_DTRAF	NSHLQ2	16
MINSHLQ2_DTRAF		0
MAXCEN_DTRAF		0
PRINT_DTRAF		−1
SWAP_DTRAF		SWAP
DKEXT	DRVKEXT	3
SCREEN_DKEXT		SCREEN
MAXSIZE_DKEXT		0
MINSIZE_DKEXT		5
MAXCEN_DKEXT		1
PRINT_DKEXT		−1
SWAP_DKEXT		SWAP
MXMBLK_DKEXT		depends on hardware (-B parameter on `molpro` command)

Table 7: Default thresholds and alias names for `direct` calculations

Parameter	Alias	Default value
THREST	THRAO	$\min(\Delta E \cdot 1.d-2, 1.d-9)^{a,b}$
THRINT	THRSO	$\min(\Delta E \cdot 1.d-2, 1.d-9)^{a,b}$
THRPROD	THRP	$\min(\Delta E \cdot 1.d-3, 1.d-10)^{a,b}$
THRMAX		1.d-8[b]
THREST_DSCF	THRDSCF	≤ 1.d-10 (depending on accuracy and basis set)
THRMAX_DSCF	THRDSCF_MAX	THRMAX
THR_DTRAF	THRDTRAF	
THREST_DTRAF	THRAO_DTRAF	[THR_DTRAF, THREST]
THRINT_DTRAF	THRAO_DTRAF	[THR_DTRAF, THRINT]
THRPROD_DTRAF	THRP_DTRAF	[THR_DTRAF, THRPROD]
THR_D2EXT	THR2EXT	THR_DTRAF
THREST_D2EXT	THRAO_D2EXT	[THR_D2EXT, THREST_DTRAF]
THRINT_D2EXT	THRSO_D2EXT	[THR_D2EXT, THRINT_DTRAF]
THRPROD_D2EXT	THRP_D2EXT	[THR_D2EXT, THRPROD_DTRAF]
THR_D3EXT	THR3EXT	THR_DTRAF
THREST_D3EXT	THRAO_D3EXT	[THR_D3EXT, THREST_DTRAF]
THRINT_D3EXT	THRSO_D3EXT	[THR_D3EXT, THRINT_DTRAF]
THRPROD_D3EXT	THRP_D3EXT	[THR_D3EXT, THRPROD_DTRAF]
THR_D4EXT	THR4EXT	THR_DTRAF
THREST_D4EXT	THRAO_D4EXT	[THR_D4EXT, THREST_DTRAF]
THRINT_D4EXT	THRSO_D4EXT	[THR_D4EXT, THRINT_DTRAF]
THRPROD_D4EXT	THRP_D4EXT	[THR_D4EXT, THRPROD_DTRAF]
THR_DCCSD	THRCCSD	THR_DTRAF
THREST_DCCSD	THRAO_DCCSD	[THR_DCCSD, THREST_DTRAF]
THRINT_DCCSD	THRSO_DCCSD	[THR_DCCSD, THRINT_DTRAF]
THRPROD_DCCSD	THRP_DCCSD	[THR_DCCSD, THRPROD_DTRAF]
THRMAX_DCCSD	THRMAX_DTRAF	THRMAX
THR_DMP2	THRDMP2	THR_DTRAF
THREST_DMP2	THRAO_DMP2	[THR_DMP2, THREST_DTRAF, *default*[c]]
THRINT_DMP2	THRSO_DMP2	[THR_DMP2, THRINT_DTRAF, *default*[c]]
THRPROD_DMP2	THRP_DMP2	[THR_DMP2, THRPROD_DTRAF, *default*[c]]
THR_LMP2	THRLMP2	THR_DTRAF
THREST_LMP2	THRAO_LMP2	[THR_LMP2, THREST_DTRAF, *default*[c]]
THRQ1_LMP2	THRQ1	[THR_LMP2, THRPROD_DTRAF, *default*[c]]
THRQ2_LMP2	THRQ2	[THR_LMP2, THRINT_DTRAF, *default*[c]]
THRAO_ATTEN]	THRATTEN	THREST_LMP2
THR_DKEXT	THRKEXT	
THREST_DKEXT	THRAO_DKEXT	[THR_DKEXT, THREST]
THRINT_DKEXT	THRSO_DKEXT	[THR_DKEXT, THRINT]
THRPROD_DKEXT	THRP_DKEXT	[THR_DKEXT, THRPROD]
THRMAX_DKEXT		THRMAX

a) ΔE is the requested accuracy in the energy (default 1.d-6).

b) The thresholds are reduced if the overlap matrix has small eigenvalues.

c) The default thresholds for DMP2 and LMP2 are $0.1 \cdot \Delta E$.

14.2.1 Example for integral-direct calculations

```
memory,2,m
$method=[hf,mp2,ccsd,qci,bccd,multi,mrci,acpf,rs3]             !some methods
basis=vdz                                                      !basis
geometry={o;h1,o,r;h2,o,r,h1,theta}                            !geometry
gdirect                                                        !direct option
r=1 ang,theta=104                                              !bond length and angle
do i=1,#method                                                 !loop over methods
$method(i)                                                     !run method(i)
e(i)=energy                                                    !save results in variables
dip(i)=dmz
enddo
table,method,e,dip                                             !print table of results
```

http://www.molpro.net/info/current/examples/h2o_direct.com

This jobs produces the following table:

```
 METHOD        E             DIP
 HF       -76.02145798   0.82747348
 MP2      -76.22620591   0.00000000
 CCSD     -76.23580191   0.00000000
 QCI      -76.23596211   0.00000000
 BCCD     -76.23565813   0.00000000
 MULTI    -76.07843443   0.76283026
 MRCI     -76.23369819   0.76875001
 ACPF     -76.23820180   0.76872802
 RS3      -76.23549448   0.75869972
```

15 DENSITY FITTING

Density fitting can be used to approximate the integrals in spin restricted Hartree-Fock (HF), density functional theory (KS), second-order Møller-Plesset perturbation theory (MP2 and RMP2), explicitly correlated MP2 (MP2-F12), and all levels of closed-shell local correlation methods (LCC2, LMP2-LMP4, LQCISD(T), LCCSD(T)). Density fitting is invoked by adding the prefix DF- to the command name, e.g. DF-HF, DF-KS, DF-MP2 and so on. Gradients are available for DF-HF, DF-KS, and DF-LMP2. Symmetry is not implemented for density fitting programs. Therefore, symmetry is turned off automatically if DF- is found in the input.

By default, a fitting basis set will be chosen automatically that corresponds to the current orbital basis set and is appropriate for the method. For instance, if the orbital basis set is VTZ, the default fitting basis is VTZ/JKFIT for DF-HF or DF-KS, and VTZ/MP2FIT for DF-MP2. Other fitting basis sets from the library can be chosen using the DF_BASIS option, e.g.

```
BASIS=VTZ              !use VTZ orbital basis
DF-HF,DF_BASIS=VQZ     !use VQZ/JKFIT fitting basis
DF-MP2,DF_BASIS=VQZ    !use VQZ/MP2FIT fitting basis
```

The program then chooses automatically the set which is appropriate for the method. Optionally, the basis type can appended to the basis name and then this supercedes the default, e.g.

```
DF-HF,DF_BASIS=VQZ/JKFIT    !use VQZ/JKFIT fitting basis
```

Orbital basis sets can be chosen using type ORBITAL (but this is not recommended normally!). Contraction/uncontraction can be forced appending (CONTRACT) or (UNCONTRACT) to the basis name, e.g.
DF_BASIS=AVQZ(UNCONTRACT)/ORBITAL.
If other options are given in parenthesis, these can be separeted by commas, e.g.
DF_BASIS=AVQZ(f/d,UNCONTRACT)/ORBITAL.
Alternative forms, which should work as well, are
DF_BASIS=AVQZ(f/d)(UNCONTRACT)/ORBITAL
or
DF_BASIS=AVQZ(f/d)/ORBITAL(UNCONTRACT).
Note that the CONTRACT/UNCONTRACT option cannot be used with basis set names previously defined in a basis block (see below).

Alternatively, fitting basis sets can be defined in a preceding basis block (see 11), and then be refered to with their set names, e.g.,

```
DF-HF, DF_BASIS=MYJKBASIS
DF-MP2, DF_BASIS=MYMP2BASIS
```

where MYJKBASIS and MYMP2BASIS are sets defined in a basis block. In this case it is the responsibility of the user to ensure that the basis set is appropriate for the method.

Further options, as fully described in section 15.1, can be added on the command line. In this case they are valid only for the current command. Alternatively, the options can be specifed on a separate DFIT directive. If this is given within a command block, the options are used only for the current program; this is entirely equivalent to the case that the options are specified on the command line. However, if a DFIT (or GDFIT) directive is given outside of a command block, the specified options are used globally in all subsequent density fitting calculations in the same run.

The options specified on a global DFIT directive are also passed down to procedures. However, if a DFIT is given within a procedure, the corresponding options are used only in the same procedure and procedures called from it. When the procedure terminates, the options from the previous level are recovered.

15.1 Options for density fitting

The options described in this section have sensible default values and usually do not have to be given. Many options described below have alias names. These can be obtained using

HELP,CFIT,ALIASES.

15.1.1 Options to select the fitting basis sets

BASIS	Basis set for fitting (Default: set corresponding to the orbital basis)
BASIS_COUL	Basis set for Coulomb fitting (default BASIS)
BASIS_EXCH	Basis set for exchange fitting (default BASIS)
BASIS_MP2	Fitting basis set for DF-MP2 (default BASIS)
BASIS_CCSD	Fitting basis set for DF-LCCSD (default BASIS)

15.1.2 Screening thresholds

THRAO	Threshold for neglecting contracted 3-index integrals in the AO basis (default 1.d-8).
THRMO	Threshold for neglecting half-transformed 3-index integrals (default 1.d-8).
THRSW	Threshold for Schwarz screening (default 1.d-5).
THROV	Threshold for neglecting 2-index integrals in the AO (default 1.d-10.
THRPROD	Product screening threshold for first half transformation (default 1.d-8).

Analogous thresholds for specfic programs can be set by appending the above keywords by the following specifications

_SCF	Coulomb and exchange fitting in DF-HF/DF-KS
_COUL	Coulomb fitting in DF-HF/DF-KS
_EXCH	Exchange fitting in DF-HF/DF-KS
_CPHF	Coulomb and exchange fitting in CPHF
_SCFGRD	Coulomb and exchange fitting in DF-HF/DF-KS gradients

The default values are the same as for the general thresholds.

Further thresholds:

THR2HLF	Threshold for second-half transformation in exchange fitting (default THRAO_SCF)
THRASM_SCF	Threshold for local assembly of exchange matrix (default THRAO_SCF)
THRAO_FOCK	Threshold for Coulomb fitting in DF-KS (default MIN(THRAO_SCF*1.d-2,1.d-12))

15.1.3 Parameters to enable local fitting

Local fitting as described in H.-J. Werner, F. R. Manby, and P. J. Knowles, J. Chem. Phys. **118**, 8149 (2003), Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, Mol. Phys. **102**, 2311 (2004), and M. Schütz, H.-J. Werner, R. Lindh and F. R. Manby, J. Chem. Phys. **121**, 737 (2004). can be activated by setting LOCFIT=1. By default, local fitting is disabled, because under certain circumstances it can lead to unacceptable errors. For instance, local fitting must not be used in counter-poise calculations, since the lack of fitting functions at the dummy atoms can lead to wrong results.

Local fitting can be restricted to certain programs, using the following options:

LOCFIT	If positive, use local fitting in all programs in which it is available (default 0).
LOCFIT_SCF	If positive, use local fitting in SCF (default LOCFIT)
LOCFIT_MP2	If positive, use local fitting in DF-LMP2; 1: use orbital domains; 2: use pair domains (default LOCFIT)

LOCFIT_F12	If positive, use local fitting in DF-LMP2-F12 (default LOCFIT)
LOCFIT_CCSD	If positive, use local fitting in DF-LCCSD (default LOCFIT)
LOCFIT_2EXT	If positive, use local fitting in LCCSD 2ext transformation (default LOCFIT_CCSD)
LOCFIT_3EXT	If positive, use local fitting in LCCSD 3ext transformation (default LOCFIT_CCSD)
LOCFIT_4EXT	If positive, use local fitting in LCCSD 4ext transformation (default LOCFIT_CCSD)
LOCFIT_CPHF	If positive, use local fitting in CPHF (default LOCFIT)
LOCFIT_SCFGRD	If positive, use local fitting in gradient calculations (default LOCFIT)
LOCORB	If positive, use localized orbitals in DF-HF (default 1)
LOCTRA	If positive, use local screening in first half transformation (default LOCFIT).
DSCREEN	If positive, enable density screening in LMP2 (default 0)
KSCREEN	If positive, enable fit-basis Schwarz screening in LMP2 (default depends on LOCTRA).

15.1.4 Parameters for fitting domains

The following options can be used to modify the domains used in local fitting. These parameters only have an effect if LOCFIT=1. The local fitting domains are determined in two steps: first *primary* orbital domains are deterimined. In the LMP2 and LCCSD programs, the primary orbital domains are the same as used for excitation domains and determined by the Boughton-Pulay procedure, as described in Sect. 29. Depending on the value of FITDOM_MP2 or FITDOM_CCSD for LMP2 and LCCSD, respectively, either the orbital domains are used directly or united pair domains are generated. In DF-HF the primary orbital domains include all basis functions at atoms which have Löwdin charges greater or equal to THRCHG_SCF. In the second step the primary fitting domains are extended using either distance criteria (RDOMAUX, in bohr) or bond connectivity criteria (IDOMAUX). IDOMAUX=1 means to include all functions at atoms wich are at most one bond distant from the primary domains. By default, distance criteria are used. However, if IDOMAUX.ge.0, the distance criteria are ignored and connectivity is used.

THRCHG_SCF	Parameter to select the primary orbital domains in local exchange fitting (default 0.1). All atoms are include which have L"owdin charges greater than this value. The primary domains are extended according to RDOMAUX_SCF or IDOMAUX_SCF.
FITDOM_MP2	Parameter to select primary fitting domains in LMP2 transformation (default 3). 1: use orbital domains; 2: use united orbital domains of strong pairs; 3: use united orbital domains of strong and weak pairs (default 3). The primary domains are extended according to RDOMAUX_MP2 or IDOMAUX_MP2
FITDOM_CCSD	Similar to FITDOM_MP2 but used for LCCSD 2-ext transformation.
RDOMAUX_SCF	Distance criterion for fitting domain extension in SCF (default 5.0)

IDOMAUX_SCF	Connectivity criterion for fitting domain extension in SCF (default 0)
RDOMAUX_CORE	Distance criterion for core orbital fitting domain extension in SCF (default RDOMAUX_SCF).
IDOMAUX_CORE	Connectivity criterion for core orbital fitting domain extension in SCF (default IDOMAUX_SCF).
RDOMSCF_START	Distance criterion for fitting domain extension in the initial SCF iterations (default 3.0).
IDOMSCF_START	Connectivity criterion for fitting domain extension in the initial SCF iterations (default 1).
RDOMSCF_FINAL	Distance criterion for fitting domain extension in the final SCF iterations (default RDOMAUX_SCF).
IDOMSCF_FINAL	Connectivity criterion for fitting domain extension in the final SCF iterations (default IDOMAUX_SCF).
RDOMAUX_MP2	Distance criterion for fitting domain extension in LMP2. The default value depends on FITDOM_MP2
IDOMAUX_MP2	Connectivity criterion for fitting domain extension in LMP2. The default value depends on FITDOM_MP2
RDOMAUX_CCSD	Distance criterion for fitting domain extension in LCCSD. The default value depends on FITDOM_CCSD).
IDOMAUX_CCSD	Connectivity criterion for fitting domain extension in LCCSD. The default value depends on FITDOM_CCSD.
RDOMAUX_CPHF	Distance criterion for fitting domain extension in CPHF (default 3.0).
RDOMAUX_SCFGRD	Distance criterion for fitting domain extension in gradients (default 5.0).
SCSGRD	Switches the DF-LMP2 analytic gradient to Grimmes SCS scaled MP2 energy functional (default 0).

15.1.5 Miscellaneous control options

There is a rather large number of parameters. Many of these should normally not be changed, and therefore only a subset is described here. A full list can be obtained using

```
HELP,CFIT
```

16 THE SCF PROGRAM

The Hartree-Fock self-consistent field program is invoked by one of the following commands:

HF or RHF	calls the spin-restricted Hartree-Fock program
UHF or UHF-SCF,*options*	calls the spin-unrestricted Hartree-Fock program

In contrast to older versions of MOLPRO, the HF and RHF directives have identical functionality and can both be used for closed-shell or open-shell calculations. Other aliases are HF-SCF or RHF-SCF.

Often, no further input is necessary. By default, the number of electrons is equal to the nuclear charge, the wavefunction is assumed to be totally symmetric (symmetry 1), and the spin multiplicity is 1 (singlet) for an even number of electrons and 2 (doublet) otherwise. The Aufbau principle is used to determine the occupation numbers in each symmetry. Normally, this works well in closed-shell cases, but sometimes wrong occupations are obtained or the wavefunction alternates between different orbital spaces. In such cases, the OCC directive must be used to force convergence to the desired state. The default behaviour can be modified either by options on the command line, or by directives.

In open-shell cases, we recommend to use the WF, OCC, CLOSED, or OPEN cards to define the wavefunction uniquely. Other commands frequently used are START and ORBITAL (or SAVE) to modify the default records for starting and optimized orbitals, respectively. The SHIFT option or directive allows to modify the level shift in the RHF program, and EXPEC to calculate expectation values of one-electron operators (see section 6.13).

Density fitting can be used for closed and open-shell spin-restricted HF and is involked by a prefix DF- (DF-HF or DF-RHF, see section 15). For UHF, only Coulomb fitting is possible (CF-UHF). Density fitting very much speeds up calculations for large molecules. The greatest savings are seen for large basis sets with high angular momentum functions. For details see R. Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, *Fast Hartree-Fock theory using local density fitting approximations*, Mol. Phys. **102**, 2311 (2004). **All publications resulting from DF-HF or DF-KS calculations should cite this work.**

16.1 Options

In this section the options for HF|RHF|UHF are described. For further options affecting Kohn-Sham caluculations see section 17. For compatibility with previous *MOLPRO* versions, options can also be given on subsequent directives, as described in later sections.

16.1.1 Options to control HF convergence

ACCU[RACY]=*accu*	Convergence threshold for the density matrix (square sum of the density matrix element changes). Tf *accu*> 1, a threshold of 10^{-}*accu*) is used. The default depends on the global ENERGY threshold.
ENERGY=*thrden*	The convergence threshold for the energy. The default depends on the global ENERGY threshold.
START=*record*	Record holding start orbitals.
SAVE|ORBITAL=*record*	Dump record for orbitals.

`MAXIT`=*maxit*	Maximum number of iterations (default 60)
`SHIFTA|SHIFTC`=*shifta*	Level shift for closed-shell orbitals in RHF (default −0.3) and α-spin orbitals in UHF (default 0).
`SHIFTB|SHIFTO`=*shiftb*	Level shift for open-shell orbitals in RHF and β-spin orbitals in UHF (default 0)
`NITORD|NITORDER]`=*nitord*	In open-shell calculations, the orbitals are reordered after each iteration to obtain maximum overlap with the orbitals from the previous iteration. This takes only effect after *nitord* iterations. The default is depends on the quality of the starting guess.
`NITSH|NITSHIFT`=*nitsh*	If the iteration count is smaller than *nitsh*, the shifts are set to zero. The default depends on the quality of the starting guess. TORT—
`NITCL|NITCLOSED`=*nitcl*	If the iteration count is smaller than *nitcl*, only the closed-shell part of the Fock matrix is used (default *nitcl*= 0).
`NITOCC`=*nitocc*	Starting with iteration *nitocc* the occupation pattern is kept fixed. The default depends on the quality of the starting guess.
`NITORT|NITORTH`=*nitort*	The orbitals are reorthonormalized after every *nitort* iterations. The default is *nitort*= 10.
`POTFAC`=*potfac*	Scale factor for potential energy in first iteration (default 1.0).

Note that in case of a restart the iteration count starts with 3.

16.1.2 Options for the diagonalization method

In calculations with very large basis sets, the diagonalization time becomes a significant fraction of the total CPU time. This can be reduced using the orbital rotation method as described in R. Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, Mol. Phys. **102**, 2311 (2004))

`MINROT`=*minrot*	If *minrot*$\geq$ 0, the orbital rotation method is employed. Explicit diagonalization of the full Fock matrix is performed in the first *minrot* iterations and in the last iteration. If *minrot*=0, a default is used which depends on the starting guess.
`NEXPR`=*nexpr*	Number of terms used in the exponential expansion of the unitary orbital transformation matrix (default 4).
`DEROT`=*nexpr*	Energy gap used in the orbital rotation method. For orbitals within $\pm$*derot* hartree of the HOMO orbital energy the Fock matrix is constructed and diagonalized (default 1.0)
`JACOBI`=*jacobi*	If nonzero, use Jacobi diagonalization.

16.1.3 Options for convergence acceleration methods (DIIS)

For more details, see `IPOL` directive.

`IPTYP`=*iptyp*	Interpolation type (default `DIIS`, see `IPOL` directive).
`IPNIT|DIIS_START`=*ipnit*	First iteration for DIIS interpolation.
`IPSTEP|DIIS_STEP`=*ipstep*	Iteration increment for DIIS interpolation.
`MAXDIS|MAXDIIS`=*maxdis*	Max number of Fock matrices used in DIIS interpolation (default 10).

16.1.4 Options for integral direct calculations

`DIRECT`	(logical). If given, do integral-direct HF.
`THRMIN|THRDSCF_MIN`=*value*	Final integral screening threshold for DSCF.
`THRMAX|THRDSCF_MAX`=*value*	Initial integral screening threshold for DSCF.
`THRINT|THRDSCF`=*value*	Same as `THRDSCF_MIN`.
`PRESCREEN`=*value*	If nonzero, use density screening (default).
`DISKSIZE]`=*value*	Max disk size in Byte for semi-direct calculations (currently disabled).
`BUFSIZE`=*value*	Max memory buffer size for semi-direct calculations (currently disabled).
`THRDISK`=*value*	Threshold for writing integrals to disk (currently disabled).
`PRINT_DFOCK`=*value*	Print option for direct Fock matrix calculation.

16.1.5 Special options for UHF calculations

`NATORB`=*record*	Save natural charge orbitals in given record.
`UNOMIN`=*unomin*	Minimum occpation number for UNO-CAS (default 0.02)
`UNOMAX`=*unomax*	Maximum occupation number for UNO-CAS (default 1.98)

16.1.6 Options for local density-fitting calculations

Please refer section 15 for more options regarding density fitting. The following options affect local density fitting, as described in H.-J. Werner, F. R. Manby, and P. J. Knowles, J. Chem. Phys. **118**, 8149 (2003), and R. Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, Mol. Phys. **102**, 2311 (2004)). Note that local fitting affects the accuracy.

`LOCFIT`=*locfit*	If nonzero, use local fitting for exchange. If > 1, also use local fitting for Coulomb (not recommended).
`RDOM`=*locfit*	Radius for fitting domain selection in local fitting (default 5 bohr).
`RDOMC`=*locfit*	Radius for fitting domain selection for core orbitas in local fitting (default `RDOM`).
`DOMSEL`=*domesel*	Criterion for selecting orbital domains in local fitting (default 0.1).

16.1.7 Options for polarizabilities

`POLARI`=*value*	If nonzero, compute analytical dipole polarizabilities. See also the `POLARI` directive (section 16.8), which allows to specify various one-electron operators (by default, the dipole operator is used).
`THRCPHF`=*thresh*	Threshold for CPHF if polarizabilities are computed (default 1.d-6).

16.1.8 Printing options

`PRINT|ORBPRINT`=*value*	Number of virtual orbitals to be printed. If *value*=0, the occupied orbitals are printed.
`DEBUG`=*value*	Option for debug print.

16.2 Defining the wavefunction

The number of electrons and the total symmetry of the wavefunction are specified on the `WF` card:

`WF`,*elec,sym,spin*

where

elec	is the number of electrons
sym	is the number of the irreducible representation
spin	defines the spin symmetry, $spin = 2 * S$ (singlet=0, doublet=1, triplet=2 etc.)

Note that these values take sensible defaults if any or all are not specified (see section 10.2).

16.2.1 Defining the number of occupied orbitals in each symmetry

`OCC`,$n_1, n_2, \ldots, n_8$

To avoid convergence problems in cases with high symmetry, this card should be included whenever the occupation pattern is known in advance. n_i is the number of occupied orbitals in the irreducible representation i. The total number of orbitals must be equal to (*elec*+*spin*)/2 (see `WF` card).

16.2.2 Specifying closed-shell orbitals

`CLOSED`,$n_1, n_2, \ldots, n_8$

This optional card can be used in open-shell calculations to specify the number of closed-shell orbitals in each symmetry. This makes possible to force specific states in the absence of an `OPEN` card.

16.2.3 Specifying open-shell orbitals

`OPEN`,$orb_1.sym_1, orb_2.sym_2, \ldots, orb_n.sym_n$

This optional card can be used to specify the singly occupied orbitals. The number of singly occupied orbitals must be equal to *spin*, and their symmetry product must be equal to *sym* (see `WF` card). If the `OPEN` card is not present, the open shell orbitals are selected automatically. The algorithm tries to find the ground state, but it might happen that a wrong state is obtained if there are several possibilities for distributing the open shell electrons among the available orbitals. This can also be avoided using the `CLOSED` card. If $orb_i.sym$ is negative, this orbital will be occupied with negative spin (only allowed in UHF).

16.3 Saving the final orbitals

ORBITAL,*record.file*
SAVE,*record.file*

The optimized orbitals, and the corresponding density matrix, fock matrix, and orbital energies are saved on *record.file*. SAVE is an alias for ORBITAL. If this card is not present, the defaults for *record* are:

RHF	2100	
UHF	2200	(holds both α and β-spin orbitals and related quantities)

These numbers are incremented by one for each subsequent calculation of the same type in the same input. Note that this holds for the sequence number in the input, independently in which order they are executed (see section 4.3).

The default for *file* is 2.

16.4 Starting orbitals

The START directive can be used to specify the initial orbitals used in the SCF iteration. It is either possible to generate an initial orbital guess, or to start with previously optimized orbitals. Alternatively, one can also use a previous density matrix to construct the first fock operator.

If the START card is absent, the program tries to find suitable starting orbitals as follows:

First:	Try to read orbitals from *record* specified on the ORBITAL or SAVE card or the corresponding default (see ORBITAL). All files are searched.
Second:	Try to find orbitals from a previous SCF or MCSCF calculation. All files are searched.
Third:	If no orbitals are found, the starting orbitals are generated using approximate atomic densities or eigenvectors of *h* (see below).

Since these defaults are usually appropriate, the START card is not required in most cases.

16.4.1 Initial orbital guess

An initial orbital guess can be requested as follows:

START,[TYPE=]*option*

The *option* keyword can be:

H0	Use eigenvectors of *h* as starting guess.
ATDEN	Use natural orbitals of a diagonal density matrix constructed using atomic occupation numbers.

The atomic density guess works very well with minimal or generally contracted basis sets for which the first contracted basis functions correspond to the atomic 1*s*, 2*s*, 2*p* ... orbitals, e.g., Dunning's cc-pVnZ sets, the STO-3G, or the 6-31G bases. For such basis sets ATDEN is used by default. If a segmented basis set with several contractions for each shell is used, ATDEN should not be specified and H0 is used by default. Since eigenvectors of *h* are often a very poor starting guess, it is recommended to generate the starting orbitals using a small basis like STO-3G (see section 16.4.2 below).

Example:

```
r=1.85,theta=104          !set geometry parameters
geometry={O;              !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
basis=STO-3G              !first basis set
hf                        !scf using STO-3G basis
basis=6-311G              !second basis set
hf                        !scf using 6-311G basis set
```

http://www.molpro.net/info/current/examples/h2o_sto3gstart1.com

The second calculation uses the optimized orbitals of the STO-3G calculation as starting guess. This is done by default and no START card is necessary. The explicit use of START and SAVE cards is demonstrated in the example in the next section.

The following input is entirely equivalent to the one in the previous section:

```
r=1.85,theta=104          !set geometry parameters
geometry={O;              !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
basis=STO-3G              !first basis set
hf                        !scf using STO-3G basis
start,atdens              !use atomic density guess
save,2100.2               !save orbitals to record 2100.2
basis=6-311G              !second basis set
hf                        !scf using 6-311G basis set
start,2100.2              !start with orbitals from the previous STO-3G calculation.
save,2101.2               !save optimized orbitals to record 2101.2
```

http://www.molpro.net/info/current/examples/h2o_sto3gstart2.com

16.4.2 Starting with previous orbitals

START,[RECORD=]*record.file*,[*specifications*]

reads previously optimized orbitals from record *record* on file *file*. Optionally, a specific orbital set can be specified as described in section 4.11.

The specified dump record may correspond to a different geometry, basis set, and/or symmetry than used in the present calculation. Using starting orbitals from a different basis set can be useful if no previous orbitals are available and the ATDENS option cannot be used (see above).

The following example shows how to change the symmetry between scf calculations. Of course, this example is quite useless, but sometimes it might be easier first to obtain a solution in higher symmetry and then convert this to lower symmetry for further calculations.

```
r1=1.85,r2=1.85,theta=104          !set geometry parameters
geometry={O;                       !z-matrix geometry input
          H1,O,r1;
          H2,O,r2,H1,theta}
basis=vdz
hf                                 !scf using c2v symmetry
orbital,2100.2                     !save on record 2100.2

symmetry,x

hf
start,2100.2                       !start with previous orbitals from c2v symmetry
orbital,2101.2                     !save new orbitals

geometry={O;                       ! geometry has to be respecified so that
          H1,O,r1;                     ! H1 and H2 can be retagged as symmetry related
          H2,O,r2,H1,theta}
symmetry,x,y
hf
start,2101.2                       !start with orbitals from cs symmetry
orbital,2102.2                     save new orbitals
```

http://www.molpro.net/info/current/examples/h2o_c2v_cs_start.com

Note, however, that this only works well if the orientation of the molecule does not change. Sometimes it might be helpful to use the `noorient` option.

Note also that a single dump record cannot hold orbitals for different basis dimensions. Using `save=2100.2` in the second calculation would therefore produce an error.

If orbitals from a corresponding SCF calculation at a neighbouring geometry are available, these should be used as starting guess.

16.4.3 Starting with a previous density matrix

START,DENSITY=*record.file*,[*specifications*]

A density matrix is read from the given dump record and used for constructing the first fock matrix. A specific density matrix can be specified as described in section 4.11. It is normally not recommended to use the DENSITY option.

16.5 Rotating pairs of orbitals

ROTATE,$orb_1.sym, orb_2.sym, angle$

Performs a 2×2 rotation of the initial orbitals orb_1 and orb_2 in symmetry *sym* by *angle* degrees. With *angle*$= 0$ the orbitals are exchanged. See MERGE for other possibilities to manipulate orbitals. In UHF, by default only the β-spin orbitals are rotated. The initial α-spin orbitals can be rotated using

ROTATEA,$orb_1.sym, orb_2.sym, angle$

In this case ROTATEB is an alias for ROTATE.

16.6 Using additional point-group symmetry

Since *MOLPRO* can handle only Abelian point-groups, there may be more symmetry than explicitly used. For instance, if linear molecules are treated in C_{2v} instead of $C_{\infty v}$, the $\delta_{(x^2-y^2)}$-orbitals appear in symmetry 1 (A_1). In other cases, a linear geometry may occur as a special case of calculations in C_S symmetry, and then one component of the π-orbitals occurs in symmetry 1 (A'). The program is able to detect such hidden "extra" symmetries by blockings in the one-electron hamiltonian h and the overlap matrix S. Within each irreducible representation, an "extra" symmetry number is then assigned to each basis function. These numbers are printed at the end of the integral output. Usually, the extra symmetries are ordered with increasing l-quantum number of the basis functions. This information can be used to determine and fix the extra symmetries of the molecular orbitals by means of the `SYM` command.

`SYM`,$irrep, sym(1), sym(2),,, sym(n)$

$sym(i)$ are the extra symmetries for the first n orbitals in the irreducible representation *irrep*. For instance, if you want that in a linear molecule the orbitals 1.1 to 3.1 are σ and 4.1, 5.1 δ, the `SYM` card would read (calculation done with X,Y as symmetry generators):

```
SYM,1,1,1,1,2,2
```

If necessary, the program will reorder the orbitals in each iteration to force this occupation. The symmetries of occupied and virtual orbitals may be specified. By default, symmetry contaminations are not removed. If *irrep* is set negative, however, symmetry contaminations are removed. Note that this may prevent convergence if degenerate orbitals are present.

16.7 Expectation values

`EXPEC`,$oper_1, oper_2, \ldots, oper_n$

Calculates expectation values for one-electron operators $oper_1$, $oper_2$, ..., $oper_n$. See section 6.13 for the available operators. By default, the dipole moments are computed. Normally, it is recommended to use the `GEXPEC` directive if expectation values for other operators are of interest. See section 6.13 for details.

16.8 Polarizabilities

`POLARIZABILITY`[,$oper_1, oper_2, \ldots, oper_n$]

Calculates polarizabilities for the given operators $oper_1$, $oper_2$, ..., $oper_n$.. See section 6.13 for the available operators. If no operators are specified, the dipole polarizabilities are computed.

Presently, this is working only for closed-shell without direct option.

The polarizabilities are stored in the variables `POLXX, POLXY, POLXZ, POLYY, POLYZ, POLZZ`.

16.9 Miscellaneous directives

All commands described in this section are optional. Appropriate default values are normally used.

16.9.1 Level shifts

`SHIFT`,*shifta,shiftb*

A level shift of *shifta* and *shiftb* hartree for α- and β-spin orbitals, respectively, is applied. This can improve convergence, but has no effect on the solution. *shifta*$= -0.2$ to -0.3 are typical values. The defaults are *shifta*$= 0$ and *shifta*$= -0.3$ in closed and open-shell calculations, respectively, and *shiftb*$= 0$.

16.9.2 Maximum number of iterations

`MAXIT`,*maxit*

sets the maximum number of iterations to *maxit*. The default is *maxit*$= 30$.

16.9.3 Convergence threshold

`ACCU`,*accu*

The convergence threshold is set to 10**(-*accu*). This applies to the square sum of the density matrix element changes. The default is *accu*$= 10$.

16.9.4 Sanity check on the energy

`NOENEST`

This disables the sanity check on the energy even if the energy value is unreasonable. Otherwise, the energy will be automatically checked by default.

16.9.5 Print options

`ORBPRINT`,*print*,*test*

This determines the number of virtual orbitals printed at the end of the calculation. By default, *print*$= 0$, i.e., only the occupied orbitals are printed. *print*$= -1$ suppresses printing of orbitals entirely. *test*$= 1$ has the additional effect of printing the orbitals after each iteration.

16.9.6 Interpolation

`IPOL`,*iptyp,ipnit,ipstep,maxdis*

This command controls DIIS interpolation. *iptyp* can be:

`DIIS`	direct inversion of the iterative subspace. This is the default and yields mostly fastest convergence.
`DM`	obsolete. No effect in MOLPRO98
`HFM`	obsolete. No effect in MOLPRO98
`NONE`	No interpolation.

ipnit is the number of the iteration in which the interpolation starts. *ipstep* is the iteration increment between interpolations. *maxdis* is the maximum dimension of the DIIS matrix (default 10).

16.9.7 Reorthonormalization of the orbitals

`ORTH`,*nitort*

The orbitals are reorthonormalized after every *nitort* iterations. The default is *nitort*= 10.

16.9.8 Direct SCF

`DIRECT`,*options*

If this card is present, the calculation is done in direct mode. See section 14.2 for options. Normally, it is recommended to use the global `GDIRECT` command to request the direct mode. See section 14.2 for details.

17 THE DENSITY FUNCTIONAL PROGRAM

Density-functional theory calculations may be performed using one of the following commands:

DFT	calculate functional of a previously computed density.
RKS or RKS-SCF	calls the spin-restricted Kohn-Sham program. KS and KS-SCF are aliases for RKS.
UKS or UKS-SCF	calls the spin-unrestricted Kohn-Sham program

Each of these commands may be qualified with the key-names of the functional(s) which are to be used, and further options:

command, key1, key2, key3, ..., options

If no functional keyname is given, the default is LDA (see below). Following this command may appear directives specifying options for the density-functional modules (see section 17.2) or the Hartree-Fock program (see section 16.1).

On completion of the functional evaluation, or self-consistent Kohn-Sham calculation, the values of the individual functionals are stored in the MOLPRO vector variable DFTFUNS; the total is in DFTFUN, and the corresponding individual functional names in DFTNAME.

Energy gradients are available for self-consistent Kohn-Sham calculations.

Density fitting can be used for closed and open-shell spin-restricted KF and is involked by a prefix DF- (DF-KS or DF-RKS, see section 15). For UKS, only Coulomb fitting is possible (CF-UKS). Density fitting very much speeds up calculations for large molecules. The greatest savings are seen for large basis sets with high angular momentum functions. For details see R. Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, *Fast Hartree-Fock theory using local density fitting approximations*, Mol. Phys. **102**, 2311 (2004). **All publications resulting from DF-HF or DF-KS calculations should cite this work.**

Normally, sensible defaults are used to define the integration grid. The accuracy can be controlled using options as described in section 17.1 or directives as described in section 17.2). More control is provided by the GRID command, as described in section 17.3.

17.1 Options

The following options may be specified on the KS or UKS command lines:

GRID=*target*	Specifies the grid target accuracy (per atom). The default is 1.d-6 unless this has been modeified using a global THRESH, GRID option.
COARSE (logical)	If *true*, perform initial iterations with a coarser grid. Default is *false*.
GRIDMAX=*gridmax*	In the initial iterations, the grid accuracy is min(*gridmax, target*coarsefac*) (only if COARSE is set).
COARSEFAC=*coarsefac*	Factor for initial grid accuracy (see above). The default is 1000.
DFTFAC=[*fac1,fac2,..*]	Factors for each functional. The number of given values must agree with the number of functionals.
EXFAC=*factor*	Fraction of exact exchange added to the functional. The default depends on the functional.

`TOLORB=`*value*	Threshold for orbital screening (current default 1.d-15).
`MATRIX=`*matrix*	Option to select integrator. *matrix=0*: use old (slow) integrator; *matrix=1*: Use new matrix-driven integrator (default).

In addition, all options valid for HF (see section 16.1) can be given.

17.2 Directives

The following options may be used to control the operation of the DFT modules. In the Kohn-Sham case, these may come in any order before or after directives for the SCF program as described in Section 16.

17.2.1 Density source (`DENSITY`, `ODENSITY`)

`DENSITY`,*orbc.filec*,... `ODENSITY`,*orbo.fileo*,...

For non-self-consistent `DFT` calculations, specifies the source of the density matrix. The total density is read from *orbc.filec*, with further options specifying density sets in the standard way as described in Section 4.11. `ODENSITY` can be used to specify the spin density. The defaults are the densities last written by an SCF or MCSCF program.

17.2.2 Thresholds (`DFTTHRESH`)

`DFTTHRESH`,*key1=value1,key2=value2*...

Sets various truncation thresholds. *key* can be one of the following.

`TOTAL`	Overall target accuracy (per atom) of density functional. Defaults to the value of the global threshold `GRID` or the value specified by option `GRID`. For proper use of this threshold, other thresholds should be left at their default value of zero.
`ORBITAL`	Orbital truncation threshold.
`DENSITY`	Density truncation threshold.
`FOCK`	Fock matrix truncation threshold.

17.2.3 Exact exchange computation (`EXCHANGE`)

`EXCHANGE`,*factor*

For Kohn-Sham calculations, compute exchange energy according to Hartree-Fock formalism and add the contribution scaled by *factor* to the fock matrix and the energy functional. Otherwise, the default is *factor=0*, i.e., the exchange is assumed to be contained in the functional, and only the Coulomb interaction is calculated explicitly.

`DFTFACTOR`,*fac1, fac2, ...*

Provide a factor for each functional specified. The functionals will be combined accordingly. By default, all factors are one.

17.2.4 Rangehybrid methods (RANGEHYBRID)

For coupling of short-range (sr-)DFT with long-range (lr-)ab-initio methods, one first has to specify the coupling parameter μ in the sr interelectronic interaction $\sum_{i<j} \mathrm{erf}(\mu r_{ij})/r_{ij}$; this can be done by setting a variable (e.g. `mu=0.5`). As a next step, long-range ERIs have to be calculated by calling the integral program (e.g. `int;erf,mu;`).

Then sr-DFT/lr-HF calculations can be performed by calling the RKS program with the additional subcommand `rangehybrid`. Available short-range functionals are `exerf` and `ecerf` for sr-LDA, and `exerfpbe` and `ecerfpbe` for sr-PBE; as usual, the functionals have to be specified after the `rks` command (e.g. `rks,exerf,ecerf;`). The underlying short-range LDA correlation functional is that of S. Paziani, S. Moroni, P. Gori-Giorgi, G.B. Bachelet, Phys. Rev. B 73, 155111 (2006).

Finally, sr-DFT/lr-post-HF calculations can be done by adding, within a call of the chosen post-HF program, two subcommands: `srxcdft` followed by the desired short-range functionals (e.g. `srxcdft,exerf,ecerf;`), and `dftden` followed by the record number from which the density for the sr functionals is taken. Implementations are available for `ci`, `mp2`, `ccsd`, `ccsd(t)`, and the corresponding local MP2 and CC methods w/wo density-fitting.

17.2.5 Exchange-correlation potential (POTENTIAL)

`POTENTIAL`,*rec.fil*

For stand-alone DFT calculations, compute exchange-correlation potential pseudo-matrix elements, defined formally as the differential of the sum of all specified functionals with respect to elements of the atomic orbital density matrix. The matrix is written to record *rec* on file *fil*.

17.2.6 Grid blocking factor (DFTBLOCK)

`DFTBLOCK`,*nblock*

Respecify the number of spatial integration points treated together as a block in the DFT integration routines (default 128). Increasing *nblock* may enhance efficiency on, e.g., vector architectures, but leads to increased memory usage.

17.2.7 Dump integrand values(DFTDUMP)

`DFTDUMP`,*file,status*

Write out values of the integrand at grid points to the file *file*. The first line of *file* contains the number of functional components; there then follows a line for each functional giving the input key of the functional. Subsequent lines give the functional number, cartesian coordinates, integrand value and integration weight with Fortran format `(I2,3F15.10,F23.15).`

17.3 Numerical integration grid control (GRID)

Density functionals are evaluated through numerical quadrature on a grid in three-dimensional space. Although the sensible defaults will usually suffice, the parameters that define the grid can be specified by using the `GRID` top-level command, which should be presented *before* the the `DFT` or `KS` commands that will use the grid. Alternatively, `GRID` and its subcommands can be presented as directives within the `KS` program.

`GRID`,*orb.file*,*status*

The integration grid is stored on record *orb.file* (default 1800.2). The information on disk consists of two parts: the parameters necessary to define the grid, and a cache of the evaluated grid points and weights. The latter is flagged as 'dirty' whenever any parameters are changed, and whenever the geometry changes; if the cache is dirty, then when an attempt is made to use the grid, it will be recalculated, otherwise the cached values are used.

If *status* is `OLD`, an attempt to restore the grid from a previous calculation is performed; effectively, the old grid provides a template of parameters which can be adjusted using the parameter commands described below. If *status* is `NEW`, the grid is always created with default parameters. If *status* is `UNKNOWN` (the default), a new grid is created either if record *orb.file* does not exist; otherwise the old grid is used.

The `GRID` command may be followed by a number of parameter-modifying subcommands. The currently implemented default parameters are equivalent to the following input commands.

```
GRIDTHRESH,1e-5,0,0
RADIAL,LOG,3,1.0,20,25,25,30
ANGULAR,LEBEDEV,0.0,0.0
LMIN,3,5,5,7
LMAX,53,53,53,53
VORONOI,10
GRIDSAVE
GRIDSYM
```

17.3.1 Target quadrature accuracy (`GRIDTHRESH`)

`GRIDTHRESH`,*acc*,*accr*,*acca*

Specify the target accuracy of integration. Radial and angular grids are generated adaptively, with the aim of integrating the Slater-Dirac functional to the specified accuracy. *acc* is an overall target accuracy, and is the one that should normally be used; radial and angular grid target accuracies are generated algorithmically from it. However, they can be adjusted individually by specifying *accr* and *acca* respectively.

17.3.2 Radial integration grid (`RADIAL`)

`RADIAL`,*method*,m_r,*scale*,n_0,n_1,n_2,n_3

Specify the details of the radial quadrature scheme. Four different radial schemes are available, specified by *method* = `EM`, `BECKE`, `AHLRICHS` or `LOG`, with the latter being the default.

`EM` is the Euler-Maclaurin scheme defined by C. W. Murray, N. C. Handy and G. J. Laming, Mol. Phys. 78 (1993) 997. m_r, for which the default value is 2, is defined in equation (6) of the above as

$$r = \alpha \frac{x^{m_r}}{(1-x)^{m_r}} \tag{1}$$

whilst *scale* (default value 1) multiplied by the Bragg-Slater radius of the atom gives the scaling parameter α.

LOG is the scheme described by M. E. Mura and P. J. Knowles, J. Chem. Phys. 104 (1996) 9848. It is based on the transformation

$$r = -\alpha \log_e(1 - x^{m_r}) \,, \tag{2}$$

with $0 \leq x \leq 1$ and simple Gauss quadrature in x-space. The recommended value of m_r is 3 for molecular systems, giving rise to the Log3 grid; m_r=4 is more efficient for atoms. α is taken to be *scale* times the recommended value for α given by Mura and Knowles, and *scale* defaults to 1.

BECKE is as defined by A. D. Becke, J. Chem. Phys. 88 (1988) 2547. It is based on the transformation

$$r = \alpha \frac{(1+x)}{(1-x)} \,, \tag{3}$$

using points in $-1 \leq x \leq +1$ and standard Gauss-Chebyshev quadrature of the second kind for the x-space quadrature. Becke chose his scaling parameters to be half the Bragg-Slater radius except for hydrogen, for which the whole Bragg-Slater radius was used, and setting *scale* to a value other than 1 allows a different α to be used. m_r is not necessary for this radial scheme.

AHLRICHS is the radial scheme defined by O. Treutler and R. Ahlrichs, J. Chem. Phys. 102 (1995) 346. It is based on the transformation their M4 mapping

$$r = \frac{\alpha}{\log_e 2}(1+x)^{0.6} \log_e \left(\frac{2}{1-x} \right) \,, \tag{4}$$

with using standard Gauss-Chebyshev quadrature of the second kind for the x-space integration. m_r is not necessary for this radial scheme.

n_0, n_1, n_2, n_3 are the degrees of quadrature n_r (see equation (3) of Murray et al.), for hydrogen/helium, first row, second row, and other elements respectively.

accr as given by the THR command specifies a target accuracy; the number of radial points is chosen according to a model, instead of using an explicit n_i. The stricter of n_i, accr is used, unless either is zero, in which case it is ignored.

17.3.3 Angular integration grid (ANGULAR)

ANGULAR,*method*,*acca*,*crowd*
LMIN,$l_0^{\min}, l_1^{\min}, l_2^{\min}, l_3^{\min}$
LMAX,$l_0^{\max}, l_1^{\max}, l_2^{\max}, l_3^{\max}$

Specify the details of the angular quadrature scheme. The default choice for *method* is LEBEDEV (ie. as in A. D. Becke, J. Chem. Phys. 88 (1988) 2547) which provides angular grids of octahedral symmetry. The alternative choice for *method* is LEGENDRE which gives Gauss-Legendre quadrature in θ and simple quadrature in ϕ, as defined by C. W. Murray, N. C. Handy and G. J. Laming, Mol. Phys. 78 (1993) 997.

Each type of grid specifies a family of which the various members are characterized by a single quantum number l; spherical harmonics up to degree l are integrated exactly. $l\min_i$ and $l\max_i, i = 0,1,2,3$ specify allowed ranges of l for hydrogen/helium, first row, second row, and other elements respectively. For the Lebedev grids, if the value of l is not one of the set implemented in MOLPRO (3, 5, 7, 9, 11, 13, 15, 17, 19, 23, 29, 41, 47, 53), then l is increased to give the next largest angular grid available. In general, different radial points will have different l, and in the absence of any moderation described below, will be taken from $l_i^{\max}$.

crowd is a parameter to control the reduction of the degree of quadrature close to the nucleus, where points would otherwise be unnecessarily close together; larger values of crowd mean less reduction thus larger grids. A very large value of this parameter, or, conventionally, setting it c;to zero, will switch off this feature.

acca is a target energy accuracy. It is used to reduce l for a given radial point as far as possible below $l_i^{\max}$ but not lower than $l_i^{\max}$. The implementation uses the error in the angular integral of the kernel of the Slater-Dirac exchange functional using a sum of approximate atomic densities. If *acca* is zero, the global threshold is used instead, or else it is ignored.

17.3.4 Atom partitioning of integration grid (`VORONOI`)

`VORONOI`,m_μ

Controls Becke-Voronoi partitioning of space. The algorithm of C. W. Murray, N. C. Handy and G. J. Laming, Mol. Phys. 78 (1993) 997 is used, with m_μ defined by equation (24). The default value is 10.

17.3.5 Grid caching (`GRIDSAVE, NOGRIDSAVE`)

`NOGRIDSAVE`

disables the disk caching of the grid, i.e, forces the recalculation of the grid each time it is needed.

`GRIDSAVE`

forces the use of a grid cache where possible.

17.3.6 Grid symmetry (`GRIDSYM,NOGRIDSYM`)

`NOGRIDSYM`

switches off the use of symmetry in generating the integration grid, whereas

`GRIDSYM`

forces the use of any point-group symmetry.

17.3.7 Grid printing (`GRIDPRINT`)

`GRIDPRINT`,*key*=*value*,...

controls printing of the grid, which by default is not done. At present, the only possible value for *key* is `GRID`, and *value* should be specified as an integer. `GRID=0` causes the total number of integration points to be evaluated and reported; `GRID=1` additionally shows the number of points on each atom; `GRID=2` causes the complete set of grid points and weights to be printed.

17.4 Density Functionals

In the following, ρ_α and ρ_β are the α and β spin densities; the total spin density is ρ;

The gradients of the density enter through

$$\sigma_{\alpha\alpha} = \nabla\rho_\alpha \cdot \nabla\rho_\alpha \, , \sigma_{\beta\beta} = \nabla\rho_\beta \cdot \nabla\rho_\beta \, , \sigma_{\alpha\beta} = \sigma_{\beta\alpha} = \nabla\rho_\alpha \cdot \nabla\rho_\beta \, , \sigma = \sigma_{\alpha\alpha} + \sigma_{\beta\beta} + 2\sigma_{\alpha\beta} \quad (5)$$

$$\chi_\alpha = \frac{\sqrt{\sigma_{\alpha\alpha}}}{\rho_\alpha^{4/3}} \, , \chi_\beta = \frac{\sqrt{\sigma_{\beta\beta}}}{\rho_\beta^{4/3}} \, . \quad (6)$$

$$\upsilon_\alpha = \nabla^2\rho_\alpha \, , \upsilon_\beta = \nabla^2\rho_\beta \, , \upsilon = \upsilon_\alpha + \upsilon_\beta \, . \quad (7)$$

Additionally, the kinetic energy density for a set of (Kohn-Sham) orbitals generating the density can be introduced through

$$\tau_\alpha = \sum_i^\alpha |\nabla\phi_i|^2 \, , \tau_\beta = \sum_i^\beta |\nabla\phi_i|^2 \, , \tau = \tau_\alpha + \tau_\beta \, . \quad (8)$$

All of the available functionals are of the general form

$$F\left[\rho_s, \rho_{\bar{s}}, \sigma_{ss}, \sigma_{\bar{s}\bar{s}}, \sigma_{s\bar{s}}, \tau_s, \tau_{\bar{s}}, \upsilon_s, \upsilon_{\bar{s}}\right] = \int d^3\mathbf{r} K\left(\rho_s, \rho_{\bar{s}}, \sigma_{ss}, \sigma_{\bar{s}\bar{s}}, \sigma_{s\bar{s}}, \tau_s, \tau_{\bar{s}}, \upsilon_s, \upsilon_{\bar{s}}\right) \quad (9)$$

where $\bar{s}$ is the conjugate spin to s.

Below is a list of keywords for the functionals supported by MOLPRO. Additionally there are a list of alias keywords deatailed in the next section for various combinations of the primary functionals listed below.

`PW86`	.. GGA Exchange Functional. doi:10.1103/PhysRevB.33.8800
`PW92C`	Perdew-Wang 1992 GGA Correlation Functional. Electron-gas correlation energy. doi:10.1103/PhysRevB.45.13244
`B95`	Becke 1995 Correlation Functional. *tau* dependent Dynamical correlation functional. doi:10.1063/1.470829
`TH4`	.. Density an gradient dependent first and second row exchange-correlation functional. doi:TH3/4
`P86`	.. Gradient correction to VWN. doi:10.1103/PhysRevB.33.8822
`THGFCFO`	.. Density and gradient dependent first row exchange-correlation functional. FCFO = FC + open shell fitting. doi:10.1016/S0009-2614(97)00586-1
`LYP`	Lee, Yang and Parr Correlation Functional. C. Lee, W. Yang and R. G. Parr, Phys. Rev. B 37, 785(1988); B. Miehlich, A. Savin, H. Stoll and H. Preuss, Chem. Phys. Letters 157, 200 (1989)

$$K = 4\frac{A\rho_\alpha\rho_\beta Z}{\rho} + AB\omega\sigma\left(\rho_\alpha\rho_\beta\,(47-7\,\delta)/18 - 2\rho^2/3\right) + \sum_s AB\omega\left(\rho_s\rho_{\bar{s}}\left(8\,2^{2/3}e\rho_s^{8/3} - (5/2-\delta/18)\,\sigma_{ss} - \frac{(\delta-11)\,\rho_s\sigma_{ss}}{9\rho}\right). + \left(2\rho^2/3-\rho_s^2\right)\sigma_{\bar{s}\bar{s}}\right), \tag{10}$$

where

$$\omega = e^{-\frac{c}{\rho^{1/3}}} Z\rho^{-11/3}, \tag{11}$$

$$\delta = \frac{c+dZ}{\rho^{1/3}}, \tag{12}$$

$$B = 0.04918, \tag{13}$$

$$A = 0.132, \tag{14}$$

$$c = 0.2533, \tag{15}$$

$$d = 0.349, \tag{16}$$

$$e = \frac{3}{10}\left(3\pi^2\right)^{2/3} \tag{17}$$

and

$$Z = \left(1+\frac{d}{\rho^{1/3}}\right)^{-1}. \tag{18}$$

EXERF

Short-range LDA correlation functional. Local-density approximation of exchange energy
for short-range interelectronic interaction $\mathrm{erf}(\mu r_{12})/r_{12}$,
A. Savin, in Recent Developments and Applications of Modern Density Functional Theory, edited by J.M. Seminario (Elsevier, Amsterdam, 1996).

$$\varepsilon_x^{\mathrm{SR}}(r_s,\zeta,\mu) = \frac{3}{4\pi}\frac{\phi_4(\zeta)}{\alpha\, r_s} - \frac{1}{2}(1+\zeta)^{4/3} f_x\left(r_s,\mu(1+\zeta)^{-1/3}\right) + \frac{1}{2}(1-\zeta)^{4/3} f_x\left(r_s,\right.$$

with

$$\phi_n(\zeta) = \frac{1}{2}\left[(1+\zeta)^{n/3}+(1-\zeta)^{n/3}\right], \tag{20}$$

$$f_x(r_s,\mu) = -\frac{\mu}{\pi}\left[(2y-4y^3)\,e^{-1/4y^2} - 3y + 4y^3 + \sqrt{\pi}\,\mathrm{erf}\left(\frac{1}{2y}\right)\right], \qquad y = \frac{\mu\,\alpha\, r_s}{2}$$

and $\alpha = (4/9\pi)^{1/3}$.

`PBEX`	PBE Exchange Functional doi:10.1103/PhysRevLett.77.3865
`B86`	$X\alpha\beta\gamma$. Divergence free semiempirical gradient-corrected exchange energy functional. $\lambda = \gamma$ in ref. doi:10.1063/1.450025
`BR`	Becke-Roussel Exchange Functional. A. D. Becke and M. R. Roussel,Phys. Rev. A 39, 3761 (1989)

$$K = \frac{1}{2}\sum_s \rho_s U_s, \tag{22}$$

where

$$U_s = -(1 - e^{-x} - xe^{-x}/2)/b, \tag{23}$$

$$b = \frac{x^3 e^{-x}}{8\pi\rho_s} \tag{24}$$

and x is defined by the nonlinear equation

$$\frac{xe^{-2x/3}}{x-2} = \frac{2\pi^{2/3}\rho_s^{5/3}}{3Q_s}, \tag{25}$$

where

$$Q_s = (\upsilon_s - 2\gamma D_s)/6, \tag{26}$$

$$D_s = \tau_s - \frac{\sigma_{ss}}{4\rho_s} \tag{27}$$

and

$$\gamma = 1. \tag{28}$$

`BW`	Becke-Wigner Exchange-Correlation Functional. Hybrid exchange-correlation functional comprimising Becke's 1998 exchange and Wigner's spin-polarised correlation functionals. doi:10.1039/FT9959104337
`HCTH147`	Handy least squares fitted functional doi:10.1063/1.480732
`CS2`	Colle-Salvetti correlation functional. R. Colle and O. Salvetti, Theor. Chim. Acta 37, 329 (1974); C. Lee, W. Yang and R. G. Parr, Phys. Rev. B 37, 785(1988)

`CS2` is defined through

$$K = -a\left(\frac{\rho + 2b\rho^{-5/3}\left[\rho_\alpha t_\alpha + \rho_\beta t_\beta - \rho t_W\right]e^{-c\rho^{-1/3}}}{1 + d\rho^{-1/3}}\right) \tag{29}$$

where

$$t_\alpha = \frac{\tau_\alpha}{2} - \frac{\upsilon_\alpha}{8} \qquad (30)$$

$$t_\beta = \frac{\tau_\beta}{2} - \frac{\upsilon_\beta}{8} \qquad (31)$$

$$t_W = \frac{1}{8}\frac{\sigma}{\rho} - \frac{1}{2}\upsilon \qquad (32)$$

and the constants are $a = 0.04918, b = 0.132, c = 0.2533, d = 0.349$.

`B86MGC` — $X\alpha\beta\gamma$ with Modified Gradient Correction. B86 with modified gradient correction for large density gradients.
doi:10.1063/1.451353

`MK00` — Exchange Functional for Accurate Virtual Orbital Energies
doi:10.1063/1.481298

`PW91C` — Perdew-Wang 1991 GGA Correlation Functional
doi:10.1103/PhysRevB.46.6671

`HCTH120` — Handy least squares fitted functional
doi:10.1063/1.480732

`PW91X` — Perdew-Wang 1991 GGA Exchange Functional
doi:10.1103/PhysRevB.46.6671

`THGFC` — .. Density and gradient dependent first row exchange-correlation functional for closed shell systems. Total energies are improved by adding DN, where N is the number of electrons and $D = 0.1863$.
doi:10.1016/S0009-2614(97)00586-1

`TH1` — Tozer and Handy 1998. Density and gradient dependent first row exchange-correlation functional.
doi:10.1063/1.475638

`ECERF` — Short-range LDA correlation functional. Local-density approximation of correlation energy
for short-range interelectronic interaction $\mathrm{erf}(\mu r_{21})/r_{12}$,
S. Paziani, S. Moroni, P. Gori-Giorgi, and G. B. Bachelet, Phys. Rev. B 73, 155111 (2006).

$$\varepsilon_c^{\mathrm{SR}}(r_s,\zeta,\mu) = \varepsilon_c^{\mathrm{PW92}}(r_s,\zeta) - \frac{[\phi_2(\zeta)]^3 Q\left(\frac{\mu\sqrt{r_s}}{\phi_2(\zeta)}\right) + a_1\mu^3 + a_2\mu^4 + a_3\mu^5 + a_4\mu}{(1+b_0^2\mu^2)^4}$$

where

$$Q(x) = \frac{2\ln(2)-2}{\pi^2}\ln\left(\frac{1+ax+bx^2+cx^3}{1+ax+dx^2}\right), \qquad (34)$$

with $a = 5.84605$, $c = 3.91744$, $d = 3.44851$, and $b = d - 3\pi\alpha/(4\ln(2)-4)$. The parameters $a_i(r_s,\zeta)$ are given by

$$a_1 = 4b_0^6C_3 + b_0^8C_5,$$

$$a_2 = 4b_0^6C_2 + b_0^8C_4 + 6b_0^4\varepsilon_c^{\mathrm{PW92}},$$

$$a_3 = b_0^8C_3,$$

$$a_4 = b_0^8C_2 + 4b_0^6\varepsilon_c^{\mathrm{PW92}},$$

$$a_5 = b_0^8\varepsilon_c^{\mathrm{PW92}},$$

with

$$
\begin{aligned}
C_2 &= -\frac{3(1-\zeta^2)\, g_c(0,r_s,\zeta=0)}{8\, r_s^3} \\
C_3 &= -(1-\zeta^2)\frac{g(0,r_s,\zeta=0)}{\sqrt{2\pi}\, r_s^3} \\
C_4 &= -\frac{9\, c_4(r_s,\zeta)}{64 r_s^3} \\
C_5 &= -\frac{9\, c_5(r_s,\zeta)}{40\sqrt{2\pi} r_s^3} \\
c_4(r_s,\zeta) &= \left(\frac{1+\zeta}{2}\right)^2 g''\left(0, r_s\left(\frac{2}{1+\zeta}\right)^{1/3}, \ \zeta=1\right) + \left(\frac{1-\zeta}{2}\right)^2 \times \\
& \quad g''\left(0, r_s\left(\frac{2}{1-\zeta}\right)^{1/3}, \ \zeta=1\right) + (1-\zeta^2) D_2(r_s) - \frac{\phi_8(\zeta)}{5\,\alpha^2\, r_s^2} \\
c_5(r_s,\zeta) &= \left(\frac{1+\zeta}{2}\right)^2 g''\left(0, r_s\left(\frac{2}{1+\zeta}\right)^{1/3}, \ \zeta=1\right) + \left(\frac{1-\zeta}{2}\right)^2 \times \\
& \quad g''\left(0, r_s\left(\frac{2}{1-\zeta}\right)^{1/3}, \ \zeta=1\right) + (1-\zeta^2) D_3(r_s),
\end{aligned}
\tag{35}
$$

and

$$b_0(r_s) = 0.784949\, r_s \tag{36}$$

$$g''(0,r_s,\zeta=1) = \frac{2^{5/3}}{5\,\alpha^2\, r_s^2}\,\frac{1\,-\,0.02267 r_s}{(1\,+\,0.4319 r_s\,+\,0.04 r_s^2)} \tag{37}$$

$$D_2(r_s) = \frac{e^{-0.547 r_s}}{r_s^2}\left(-0.388 r_s + 0.676 r_s^2\right) \tag{38}$$

$$D_3(r_s) = \frac{e^{-0.31 r_s}}{r_s^3}\left(-4.95 r_s + r_s^2\right). \tag{39}$$

Finally, $\varepsilon_c^{\rm PW92}(r_s,\zeta)$ is the Perdew-Wang parametrization of the correlation energy of the standard uniform electron gas [J.P. Perdew and Y. Wang, Phys. Rev. B 45, 13244 (1992)], and

$$g(0,r_s,\zeta=0) = \frac{1}{2}(1 - B r_s + C r_s^2 + D r_s^3 + E r_s^4)\, \mathrm{e}^{-d r_s}, \tag{40}$$

is the on-top pair-distribution function of the standard jellium model [P. Gori-Giorgi and J.P. Perdew, Phys. Rev. B 64, 155102 (2001)], where $B = -0.0207$, $C = 0.08193$, $D = -0.01277$, $E = 0.001859$, $d = 0.7524$. The correlation part of the on-top pair-distribution function is $g_c(0,r_s,\zeta=0) = g(0,r_s,\zeta=0) - \frac{1}{2}$.

`BRUEG` Becke-Roussel Exchange Functional — Uniform Electron Gas Limit. A. D. Becke and M. R. Roussel,Phys. Rev. A 39, 3761 (1989)

As for `BR` but with $\gamma = 0.8$.

`VWN5` Vosko-Wilk-Nusair (1980) V local correlation energy. VWN 1980(V) functional. The fitting parameters for $\Delta\varepsilon_c(r_s,\zeta)_V$ appear in the caption of table 7 in the reference.
doi:VWN80

`PBEXREV` Revised PBE Exchange Functional. Changes the value of the constant R from the original PBEX functional
doi:10.1103/PhysRevLett.80.890

`TH2` .. Density and gradient dependent first row exchange-correlation functional.
doi:10.1021/jp980259s

`THGFCO` .. Density and gradient dependent first row exchange-correlation functional.
doi:10.1016/S0009-2614(97)00586-1

`LTA` Local τ Approximation. LSDA exchange functional with density represented as a function of τ.
doi:10.1063/1.479374

`G96` Gill's 1996 Gradient Corrected Exchange Functional
doi:G96

`B86R` X$\alpha\beta\gamma$ Re-optimised. Re-optimised β of B86 used in part 3 of Becke's 1997 paper.
doi:10.1063/1.475007

`DIRAC` Slater-Dirac Exchange Energy. Automatically generated Slater-Dirac exchange.
doi:10.1103/PhysRev.81.385

`B88C` Becke 1988 Correlation Functional. Correlation functional depending on B86MGC exchange functional with empirical atomic parameters, t and u. The exchange functional that is used in conjunction with B88C should replace B88MGC here.
doi:10.1063/1.454274

`B97RDF` Density functional part of B97 Re-parameterized by Hamprecht et al. Re-parameterization of the B97 functional in a self-consistent procedure by Hamprecht et al. This functional needs to be mixed with 0.21*exact exchange.
doi:10.1063/1.477267

`B88` Becke 1988 Exchange Functional
doi:10.1103/PhysRevA.38.3098

`TH3` .. Density and gradient dependent first and second row exchange-correlation functional.
doi:TH3/4

`VSXC` .
doi:10.1063/1.476577

`CS1` Colle-Salvetti correlation functional. R. Colle and O. Salvetti, Theor. Chim. Acta 37, 329 (1974); C. Lee, W. Yang and R. G. Parr, Phys. Rev. B 37, 785(1988)

`CS1` is formally identical to `CS2`, except for a reformulation in which the terms involving υ are eliminated by integration by parts. This makes the functional more economical to evaluate.

	In the limit of exact quadrature, `CS1` and `CS2` are identical, but small numerical differences appear with finite integration grids.
`MK00B`	Exchange Functional for Accurate Virtual Orbital Energies. MK00 with gradient correction of the form of B88X but with different empirical parameter. doi:10.1063/1.481298
`THGFL`	.. Density dependent first row exchange-correlation functional for closed shell systems. doi:10.1016/S0009-2614(97)00586-1
`HCTH93`	Handy least squares fitted functional doi:10.1063/1.477267
`B97DF`	Density functional part of B97. This functional needs to be mixed with 0.1943*exact exchange. doi:10.1063/1.475007
`STEST`	Test for number of electrons
`VWN3`	Vosko-Wilk-Nusair (1980) III local correlation energy. VWN 1980(III) functional doi:VWN80
`PBEC`	PBE Correlation Functional doi:10.1103/PhysRevLett.77.3865

17.4.1 Alias density functionals

Additional functional keywords are also defined as convenient aliases. The following table gives the translations.

alias	functionals	factors	Ref
B	B88	1	doi:10.1103/PhysRevA.38.3098
B-LYP	B88:LYP	1:1	
B-P	B88:P86	1:1	
B-VWN	B88:VWN5	1:1	
B3LYP	EXACT:B88:DIRAC:LYP:VWN5	0.2:0.72:0.08:0.81:0.19	
B3LYP3	EXACT:B88:DIRAC:LYP:VWN3	0.2:0.72:0.08:0.81:0.19	
B3LYP5	EXACT:B88:DIRAC:LYP:VWN5	0.2:0.72:0.08:0.81:0.19	
B88X	B88	1	doi:10.1103/PhysRevA.38.3098
B97	EXACT:B97DF	0.1943:1	
B97R	EXACT:B97RDF	0.21:1	
BECKE	B88	1	doi:10.1103/PhysRevA.38.3098
BH-LYP	EXACT:B88:LYP	0.5:0.5:1	
CS	CS1	1	
D	DIRAC	1	
HFB	B88	1	doi:10.1103/PhysRevA.38.3098
HFS	DIRAC	1	
LDA	DIRAC:VWN5	1:1	
LSDAC	PW92C	1	doi:10.1103/PhysRevB.45.13244
LSDC	PW92C	1	doi:10.1103/PhysRevB.45.13244
LYP88	LYP	1	
PBE	PBEX:PBEC	1:1	doi:10.1103/PhysRevLett.77.3865
PBE0	EXACT:PBEX:PBEC	0.25:0.75:1	doi:10.1063/1.478522
PBE0MOL	EXACT:PBEX:PW91C	0.25:0.75:1	
PBEREV	PBEXREV:PBEC	1:1	
PW91	PW91X:PW91C	1:1	doi:10.1103/PhysRevB.46.6671
S	DIRAC	1	
S-VWN	DIRAC:VWN5	1:1	
SLATER	DIRAC	1	
VS99	VSXC	1	
VWN	VWN5	1	
VWN80	VWN5	1	

17.4.2 Implementing new functionals

New functionals are implemented based upon the automatic code generation (ACG) program (doi:10.1016/S0010-4655(01)00148-5). In order to work the program requires the maple mathematics program and an XSLT parser, defined by the variable XSLT in CONFIG.

The format of the input file is an XML file containing all of the information about the new functional. All density functional XML files are placed in the directory `lib/df` and are automatically activated on the next instance of the `make` command in the MOLPRO base directory.

The root element of the XML document is `content`. At the next level the element, `functional` is expected, 1 per file.

The `functional` element has an `id` attribute which is used as the keyword for the functional in MOLPRO, and optional `doi` attribute for specifying a reference. The allowed elements are defined in table 8. The final element is `maple` for which multiple cases are allowed. A typical maple expression such as

```
A:=1.2:
```

title	Text to appear as a heading for the functional documentation
tex	Text to document the functional

Table 8: Elements allowed for defining functionals

is written as

```
<maple lhs="A">1.2</maple>.
```

To input a Maple procedure such as

```
add_together:=proc(a,b) a+b end:
```

one should write

```
<maple lhs="add_together" proc="a,b">a+b</maple>.
```

As an example the Perdew-Wang 1991 GGA exchange functional is given below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<content>
 <functional id="PW91X">
  <title>Perdew-Wang 1991 GGA Exchange Functional</title>
  <maple lhs="g">1/2*E(2*rho(s))</maple>
  <maple lhs="G">1/2*E(2*rho(s))</maple>
  <maple lhs="E" proc="n"> -3/(4*Pi)*(3*Pi^2)^(1/3)*n^(4/3)*F(S)</maple>
  <maple lhs="S">chi(s)/(2*(6*Pi^2)^(1/3))</maple>
  <maple lhs="F" proc="S">
   (1+0.19645*S*arcsinh(7.7956*S) + (0.2743-0.1508*exp(-100*S^2))*S^2)/
   (1+0.19645*S*arcsinh(7.7956*S)+0.004*S^4)
  </maple>
 </functional>
</content>
```

17.4.3 Implementing new hybrid-functionals

Hybrid functionals are defined in the file `lib/dalias.registry`. Entries can be added to this file as required, and then run `make` to update the other registry files.

17.5 Empirical damped dispersion correction

Empirical damped dispersion corrections can be calculated in addition to Kohn-Sham calculations. This is particularly important in cases where long-range correlation effects become dominant. The dispersion correction uses the van-der-Waals radii and atomic dispersion coefficients from Refs. [1-2] and its functional form is given by:

$$E_{\mathrm{disp}} = -s_6 \sum_{i,j<i}^{N_{\mathrm{atoms}}} f_{\mathrm{damp}}(R_{ij}) \frac{C_6^{ij}}{R_{ij}^6} \,. \tag{41}$$

In the above equation, N_{atoms} is the total number of atoms, R_{ij} is the interatomic distance of atoms i and j, s_6 is a global scaling parameter depending on the choice of the functional used (see below), and the C_6^{ij} values are calculated from atomic dispersion coefficients C_6^i and C_6^j in the following way:

$$C_6^{ij} = 2\frac{C_6^i C_6^j}{C_6^i + C_6^j} \quad \text{(Ref. [1])} \tag{42}$$

$$C_6^{ij} = \sqrt{C_6^i C_6^j} \quad \text{(Ref. [2])} \tag{43}$$

The function f_{damp} damps the dispersion correction for shorter interatomic distances and is given by:

$$f_{\text{damp}}(R_{ij}) = \frac{1}{1+\exp\left[-\alpha(R_{ij}/(R^i_{\text{vdW}}+R^j_{\text{vdW}})-1)\right]} \tag{44}$$

whith R^i_{vdW} being the van-der-Waals radius for atom i and α is a parameter that is usually set to 23 (Ref. [1]) or 20 (Ref. [2]).

Currently the following functionals can be used in conjunction with the empirical dispersion correction:

Table 9: Optimised scaling parameters s_6 (see Eq. (41)) for density functionals

functional	Ref. [1]	Ref. [2]
BLYP	1.40	1.20
PBE	0.70	0.75
BP86	1.30	1.05
B3LYP	-	1.05

The dispersion correction can be calculated using `dispcorr` in the Molpro input and it can be added to DFT energies by using the following template:

```
ks,<func>
eks=energy
dispcorr
eks_plus_disp=eks+edisp
```

i.e., the dispersion corrections (Eq. (41)) are stored in variables `edisp`. Note that `dispcorr` notices which type of functional has previously been used by reading in the internal Molpro variable `DFTNAME`.

The `dispcorr` program can have the following options:

`MODE` Adjusts the parametrisation used: `MODE=1` uses parameters from Ref. [1] and `MODE=2` uses parameters from Ref. [2] (default: `MODE=1`)

`SCALE` Overall scaling parameter s_6 (see Eq. (41) and table 9 for optimised values).

`ALPHA` Damping function parameter (see Eq. (44)). Smaller values lead to larger corrections for intermediate distances.

The third dispersion correction from Grimme *et al.* [3] takes into account also eith-order dispersion coefficients and uses different damping functions. It can be invoked in the same way as shown above, but by replacing `dispcorr` with `dispcorr3` in the template.

The `dispcorr3` program can have the following options:

`FUNC`	Functional name (default: `FUNC='pbe'`).
`VERSION`	Can have values 2 and 3 according to parametrisations from Refs. [2] and [3] (default: `VERSION=3`)
`ANAL`	Performs a detailed analysis of pair contributions.
`GRAD`	Cartesian gradients are computed. (note that geometry optimisations with `DFT+dispcorr3` are currently not yet possible).
`TZ`	Use special parameters for calculations with triple-zeta basis sets. Preliminary results in the SI of Ref. [3] indicate that results are slightly worse than with the default parameters and QZVP type basis sets. This option should be carfully tested for future use in very large computations.

(see also `http://toc.uni-muenster.de/DFTD3/index.html` for further documentation).

References:

[1] S. Grimme, J. Comp. Chem. **25**, 1463 (2004).
[2] S. Grimme, J. Comp. Chem. **27**, 1787 (2006).
[3] S. Grimme, J. Antony, S. Ehrlich and H. Krieg, J. Chem. Phys. **132**, 154104 (2010)

17.6 Examples

The following shows the use of both non-self-consistent and self-consistent DFT.

```
geometry={c;n,c,r}
r=1.1 angstrom
df=[b,lyp]
rhf;method(1)=program
dft;edf(1)=dftfun
uhf;method(2)=program
dft;edf(2)=dftfun
uks;method(3)=program,edf(3)=dftfun
dft;method(4)=program,edf(4)=dftfun
table,dftname,dftfuns
table,method,edf
```

`http://www.molpro.net/info/current/examples/cndft.com`

18 ORBITAL LOCALIZATION

Localized orbitals are calculated according to the Boys, Pipek-Mezey or NLMO criteria. Localization takes place within each symmetry species separately. If complete localization is desired, no symmetry should be used. All subcommands can be abbreviated by three characters.

The localization program is invoked by the `LOCALI` command

LOCALI [,*method*],[LOCMETHOD=*locmethod*],[REFORB=*record*]

The keyword *method* can be either BOYS, PIPEK or NATURAL. By default, the valence orbitals from the last energy calculation are localized using the Boys criterion. Only orbital subsets which leave the energy invariant are transformed. These defaults can be modified using the optional commands described in the following sections.

The option LOCMETHOD only applies to Pipek-Mezey localization. The value *locmethod* can take the following values:

locmethod=1: Standard iterative localization method

locmethod=2: Use second-order localization method. Redundant rotations will be eliminated.

locmethod=3: First do a first iterations using the standard method, then invoke locmethod=2. This option is recommended in cases with redundant rotations, e.g., benzene.

The option REFORB has the same effect as the directive REFORB described further below.

18.1 Defining the input orbitals (ORBITAL)

ORBITAL,*record.file*,*specifications*

The orbitals to be localized are read from dump record *record.file*. A state specific orbital set can be selected using *specifications*, as explained in section 4.11. Default are the orbitals calculated last.

18.2 Saving the localized orbitals (SAVE)

SAVE,*record.file*

This specifies the dump record where the localized orbitals are stored. If the dump record already exists, the localized orbitals are added to it. Default is the input record (cf. ORBITAL).

18.3 Choosing the localization method (METHOD)

METHOD,*method*

The localization method *method* can be either BOYS, PIPEK or NATURAL. This can also be specified as argument on the LOCALI card (see above).

18.4 Delocalization of orbitals (DELOCAL)

DELOCAL

If this card is present, the orbitals are delocalized.

18.5 Localizing AOs(LOCAO)

LOCAO

If this card is present, the number of AOs contributing to each MO is minimized. This can be useful to rotate degenerate orbitals (e.g., px, py, pz in an atom) so that pure orbitals (in this case px, py, pz) result.

This implies Pipek-Mezey localization.

18.6 Selecting the orbital space

By default, only the valence orbitals are localized, in order to ensure invariance of subsequent electron correlation treatments. This behaviour can be modified using the OCC and CORE directives.

18.6.1 Defining the occupied space (OCC)

OCC, $o_1, o_2 \ldots$

defines the highest orbital o_i in each symmetry i to be localized.

18.6.2 Defining the core orbitals (CORE)

CORE, $c_1, c_2 \ldots$

The first c_i orbitals in each symmetry are treated as core orbitals and not localized. Thus, orbitals $c_i + 1$ to o_i are localized in symmetry i.

18.6.3 Defining groups of orbitals (GROUP, OFFDIAG)

GROUP,*orb1,orb2,orb3,...*

This card defines groups of orbitals to be localized as follows:

GROUP,1.1,2.1,3.1	a group of orbitals 1-3 in symmetry 1
GROUP,1.1,-3.1	equivalent to previous example
GROUP,3.1,5.1,-8.1	this group includes orbitals 3,5,6,7,8 in symmetry 1

Orbitals in different groups are localized independently. Orbitals not included in any group are unchanged.

18.6.4 Localization between groups (OFFDIAG)

OFFDIAG

If this card is present, localize between groups instead of within groups.

18.7 Ordering of localized orbitals

ORDER,*type*

If *type*=CHARGE, the orbitals are ordered according to their charge centroids (default).

If *type*=FOCK, the orbitals are ordered according to increasing diagonal elements of the fock operator (PIPEK) or increasing Coulson-additive orbital energies (BOYS). This requires a Fock operator from the preceding energy calculation. For localization of Hartree-Fock orbitals, this operator is stored in the dump record and automatically found. For localization of MCSCF orbitals, an effective fock operator is computed from the MCSCF density matrix (see DENSITY option). Alternatively, a dump record of a previous SCF calculation can be specified on the FOCK card, and then the fock operator is read from this record. For degenerate orbitals, further ordering according to the the coordinates of charge centres is attempted (first according to largest z-coordinates, then according to x, then y).

This card does not apply to NLMO localization.

18.7.1 No reordering (NOORDER)

NOORDER

If this card is present, the localized orbitals are not reordered. This is useful if localized orbitals are used as starting guess, and it is intended that their order remains unchanged.

18.7.2 Ordering using domains (SORT)

SORT,[THRCHCHG=*charge*][THREIG=*eps*],GROUP=*igrp*],[REVERT],*centrelist*

This directive only works for Pipek-Mezey localization. The orbitals are ordered according to domains and the given centrelist. The contributions of the centres to domains are determined by Löwdin charges. Only centres with charges greater than THRCHCHG (default 0.4) are included in these domains. The orbitals are reordered according to the following criteria:

1.) The primary centre in a domain is the one with largest charge, the secondary centre the one with the next largest charge. Orbitals are reordered separately within each localization group. First all orbitals are sorted so that the primary centres are in the order of the given *centrelist*. Orbitals with primary centres which are not in *centrelist* come last.

2.) Within each group of orbitals found for a given primary centre, those containing only one centre (lone pairs) are included first. The remaining ones are ordered so that the secondary atoms are in the order of *centrelist*. Orbitals with secondary centres which are not in *centrelist* come last.

3.) If REVERT is given, the order in each localization group is reverted.

4.) If GROUP is given, only the orbitals in the given group are reordered. *igrp* is 2 for closed shells and inactive orbitals, 1 for open-shells in single reference methods, and 3 for active orbitals in CASSCF calculations.

5.) If THREIG is given, only orbitals with energies larger than the given value are reordered. *eps* must be negative. The remaining orbitals come last (first if REVERT is given).

Note that core orbitals are neither localized nor reordered.

18.7.3 Defining reference orbitals (REFORB)

REFORB,*record.file*,*specifications*

The localized orbitals are reordered such that the overlap with the reference orbitals read from *record.file* is maximized. This is useful for local correlation treatments for keeping the order of the localized constant for different geometries. A state specific orbital set can be selected using *specifications*, as explained in section 4.11.

18.7.4 Selecting the fock matrix (FOCK)

FOCK,*record.file*

This specifies a record holding a Fock operator to be used for ordering the orbitals. Note that only SCF dump records hold fock operators. Default is the Fock operator from the energy calculation which produced the input orbitals.

18.7.5 Selecting a density matrix (DENSITY)

DENSITY,*record.file*,*specifications*

This specifies a record holding a density matrix for construction of a fock operator used for ordering the orbitals. This can be used if no fock operator is available, and has only an effect for MCSCF localizations. By default, the (state averaged) MCSCF density is used. A state specific density matrix can be selected using *specifications* as described in section 4.11.

18.8 Localization thresholds (THRESH)

THRESH,*thresh,eorder*

thresh is a threshold for localization (default 1.d-12). If *eorder* is nonzero (default 1.d-4), the orbitals whose energy difference is smaller then *eorder* are considered to be degenerate and reordered according to the position of their charge centres (see section 18.7).

18.9 Options for PM localization (PIPEK)

Some special options exist for Pipek-Mezey localization (all optional):

PIPEK,METHOD=*method*,DELETE=*ndel*,MAXDL=*maxdl*,THRESH=*thresh*,ORDER=*iorder*,STEP=*step*

METHOD: *method=1*: use 2x2 rotation method (default);
method=2: use Newton-Raphson method;
method=3: Initial iterations using 2x2 rotation method , final convergence using NR method.

DELETE: Delete the last *ndel* basis functions of each angular momentum type for each atom in PM localization. This can be useful to achieve proper localization with diffuse (augmented) basis sets.

MAXDL: If *ndel*¿0 delete functions only up to angular momentum *maxdl*.

ORDER:	If *iorder=1*, order final orbitals according to increasing diagonal fock matrix elements; If *iorder=2*, order final orbitals according charge centres (default).
THRESH:	Localization threshold (same as on THRESH directive).
STEP:	Max step size in NR method (default 0.1d0).

18.10 Printing options (PRINT)

PRINT,[ORBITAL=]*pri* [,CHARGE] [,CENTRES] [,TEST] [,TRAN];

If ORB[ITAL] is given, the localized orbitals are printed.
If CHA[RGE] or CEN[TRES] is given, the charge centres of the localized orbitals are printed.
If TRAN is given, the transformation matrix is printed (Boys only).
If TEST is given, intermediate information is printed.

19 THE MCSCF PROGRAM MULTI

MULTI is a general MCSCF/CASSCF program written by
P. J. Knowles and H.-J. Werner (1984).

Bibliography:

H.-J. Werner and P. J. Knowles, J. Chem. Phys. 82, 5053 (1985).
P. J. Knowles and H.-J. Werner, Chem. Phys. Lett. 115, 259 (1985).

All publications resulting from use of this program must acknowledge the above. See also:

H.-J. Werner and W. Meyer, J. Chem. Phys. 73, 2342 (1980).
H.-J. Werner and W. Meyer, J. Chem. Phys. 74, 5794 (1981).
H.-J. Werner, Adv. Chem. Phys. LXIX, 1 (1987).

This program allows one to perform CASSCF as well as general MCSCF calculations. For CASSCF calculations, one can optionally use Slater determinants or CSFs as a N-electron basis. In most cases, the use of Slater determinants is more efficient. General MCSCF calculations must use CSFs as a basis.

A quite sophisticated optimization method is used. The algorithm is second-order in the orbital and CI coefficient changes and is therefore quadratically convergent. Since important higher order terms in the independent orbital parameters are included, almost cubic convergence is often observed. For simple cases, convergence is usually achieved in 2-3 iterations. However, convergence problems can still occur in certain applications, and usually indicate that the active space is not adequately chosen. For instance, if two weakly occupied orbitals are of similar importance to the energy, but only one of them is included in the active set, the program might alternate between them. In such cases either reduction or enlargement of the active orbital space can solve the problem. In other cases difficulties can occur if two electronic states in the same symmetry are almost or exactly degenerate, since then the program can switch from one state to the other. This might happen near avoided crossings or near an asymptote. Problems of this sort can be avoided by optimizing the energy average of the particular states. It is also possible to force convergence to specific states by choosing a subset of configurations as primary space (`PSPACE`). The hamiltonian is constructed and diagonalized explicitly in this space; the coefficients of the remaining configurations are optimized iteratively using the P-space wavefunction as zeroth order approximation. For linear molecules, another possibility is to use the `LQUANT` option, which makes it possible to force convergence to states with definite Λ quantum number, i.e., Σ, Π, Δ, etc. states.

19.1 Structure of the input

All sub-commands known to *MULTI* may be abbreviated by four letters. The input commands fall into several logical groups; within each group commands may appear in any order, but the groups must come in correct order.

a) The program is invoked by the command `MULTI` or `MCSCF`

b) cards defining partitioning of orbitals spaces – `OCC,FROZEN,CLOSED`

c) general options (most commands not otherwise specified here)

d) a `WF` card defining a state symmetry

e) options pertaining to that state symmetry – `WEIGHT,STATE,LQUANT`

f) configuration specification for that state symmetry – `SELECT,CON,RESTRICT`

g) definition of the primary configurations for that state symmetry - `PSPACE`

h) further general options

Stages d) through to h) may be repeated several times; this is the way in which you specify an average energy of several states of different symmetry.

Many options can be specified on the `MULTI` command line:

`MULTI`,*options*

Selected options:

`MAXIT`	Max. number of iterations (default 10)
`ENERGY`	Convergence threshold for energy
`GRADIENT`	Convergence threshold for gradient
`STEP`	Convergence threshold for steplength
`FAILSAFE`	(logical) Use options for more robust convergence

Many further options and thresholds, which can also be given on the command line, are described in section 19.8.5.

19.2 Defining the orbital subspaces

19.2.1 Occupied orbitals

`OCC`,$n_1, n_2, \ldots, n_8$;

n_i specifies numbers of occupied orbitals (including FROZEN and CLOSED) in irreducible representation number *i*. In the absence of an `OCC` card, the information from the most recent MCSCF calculation is used, or, if there is none, those orbitals corresponding to a minimal valence set, i.e., full valence space, are used.

19.2.2 Frozen-core orbitals

`FROZEN`,$n_1, n_2, \ldots,$*record.file*;

n_i is the number of frozen-core orbitals in irrep number *i*. These orbitals are doubly occupied in all configurations and not optimized. Note that in earlier MOLPRO versions this directive was called `CORE` and has now been renamed to avoid confusion with `CORE` orbitals in the MRCI and CCSD programs.

record.file is the record name for frozen core orbitals; if not supplied, taken from *orb* on `START` card. *record.file* can be specified in any field after the last nonzero n_i. It should always be given if the orbital guess is from a neighbouring geometry and should then specify the SCF orbitals calculated at the present geometry. If a subsequent gradient calculation is performed with this wavefunction, *record.file* is mandatory and must specify closed-shell SCF orbitals at the present geometry. Note that *record* must be larger than 2000.

If the `FROZEN` card is omitted, then the numbers of core orbitals are taken from the most recent MCSCF calculation, or otherwise no orbitals are frozen. If the `FROZEN` card is given as

FROZEN,record.file, then the orbitals corresponding to atomic inner shells are taken, i.e., 1*s* for Li–Ne, 1*s*2*s*2*p* for Na–Ar, etc. A FROZEN card without any specification resets the number of frozen core orbitals to zero.

19.2.3 Closed-shell orbitals

CLOSED,$n_1,n_2,\ldots,n_8$

n_i is the number of closed-shell orbitals in irrep number i, inclusive of any FROZEN orbitals. These orbitals do not form part of the active space, i.e., they are doubly occupied in all CSFs. In contrast to the core orbitals (see FROZEN), these orbitals are fully optimized.

If the CLOSED card is omitted, then the data defaults to that of the most recent MCSCF calculation, or else the atomic inner shells as described above for FROZEN.

19.2.4 Freezing orbitals

FREEZE,*orb.sym*;

The specified orbital will not be optimized and will remain identical to the starting guess. *orb.sym* should be an active or closed-shell orbital. If *orb.sym* is a frozen core orbital, this card has no effect.

19.3 Defining the optimized states

Each state symmetry to be optimized is specified by one WF card, which may optionally be followed by STATE, WEIGHT, RESTRICT, SELECT, CON, and/or PSPACE cards. All cards belonging to a particular state symmetry as defined on the WF card must form a block which comes directly after the WF card. The cards can be in any order, however.

19.3.1 Defining the state symmetry

The number of electrons and the total symmetry of the wavefunction are specified on the WF card:

WF,*elec,sym,spin*

where

elec	is the number of electrons
sym	is the number of the irreducible representation
spin	defines the spin symmetry, *spin*= 2*S* (singlet=0, doublet=1, triplet=2, etc.)

Note that these values take sensible defaults if any or all are not specified (see section 10.2).

The input directives STATE, WEIGHT, LQUANT, SELECT, PUNCSF always refer to the state symmetry as defined on the previous WF card. If such a directive is found before a WF card has been given, the current state symmetry is assumed, either from a previous calculation or from variables [MC]SYMMETRY(1) and [MC]SPIN(1) (if these are defined). If any of these cards or a WF card is given, the variables STATE, WEIGHT, LQUANT, SELECT are *not* used, and the number of state symmetries defaults to one, regardless of how many symmetries are specified in variable [MC]SYMMETRY.

19.3.2 Defining the number of states in the present symmetry

STATE,*nstate*;

nstate is the number of states in the present symmetry. By default, all states are optimized with weight 1 (see WEIGHT card).

19.3.3 Specifying weights in state-averaged calculations

WEIGHT,$w(1), w(2), \ldots, w(nstate)$;

$w(i)$ is the weight for the state i in the present symmetry. By default, all weights are 1.0. See also STATE card. If you want to optimize the second state of a particular state symmetry alone, specify

STATE,2;WEIGHT,0,1;

Note, however, that this might lead to root-flipping problems.

19.3.4 Dynamical weighting

Dynamical weighting, as described in J. Chem. Phys. **120**, 7281 (2004), can be activated using

DYNW,*dynfac*

The weights for each state are then computed as

$$w = 1/\cosh(dynfac * \Delta E)^2 \tag{45}$$

where ΔE is the energy difference (in Hartree) between the state under consideration and the ground state. This is dynamically adjusted during the optimization process.

19.4 Defining the configuration space

By default, the program generates a complete configuration set (CAS) in the active space. The full space may be restricted to a certain occupation pattern using the RESTRICT option. Alternatively, configurations may be selected from the wavefunction of a previous calculation using SELECT, or explicitly specified on CON cards. Note that this program only allows to select or specify orbital configurations. For each orbital configuration, all spin couplings are always included. Possible RESTRICT, SELECT and CON cards must immediately follow the WF card which defines the corresponding state symmetry.

19.4.1 Occupation restrictions

RESTRICT,*nmin,nmax*,orb_1,orb_2,...orb_n;

This card can be used to restrict the occupation patterns. Only configurations containing between *nmin* and *nmax* electrons in the specified orbitals orb_1, orb_2,...,orb_n are included in the wavefunction. If *nmin* and *nmax* are negative, configurations with exactly abs(*nmin*) and abs(*nmax*) electrons in the specified orbitals are deleted. This can be used, for instance, to omit singly excited configurations. The orbitals are specified in the form *number.sym*, where *number* is the number of the orbital in irrep *sym*. Several RESTRICT cards may follow each other. RESTRICT only works if a CONFIG card is specified before the first WF card.

RESTRICT cards given before the first WF cards are global, i.e., are active for all state symmetries. If such a global restrict card is given, variable [MC]RESTRICT is *not* used.

Additional state-specific RESTRICT cards may be given after a WF card. These are used in addition to the global orbital restrictions.

If neither state-specific nor global RESTRICT cards are found, the values from the variable [MC]RESTRICT are used.

19.4.2 Selecting configurations

SELECT,*ref1,ref2,refthr,refstat,mxshrf*;

This card is used to specify a configuration set other than a CAS, which is the default. This option automatically triggers the CONFIG option, which selects CSFs rather than determinants. Configurations can be defined using CON cards, which must follow immediately the SELECT card. Alternatively, if *ref1* is an existing MOLPRO record name, the configurations are read in from that record and may be selected according to a given threshold.

ref1=rec1.file	(*rec1*$>$ 2000) The configurations are read in from the specified record. If *ref1* is not specified, the program assumes that the configurations are read from subsequent CON cards (see CON).
ref2=rec2.file	(*rec2*$>$ 2000) Additional configurations are read from the specified record. If *rec2* is negative, all records between *rec1* and abs(*rec2*) are read. All configurations found in this way are merged.
refthr	Selection threshold for configurations read from disc (records *rec1*–*rec2*). This applies to the norm of all CSFs for each orbital configuration.
refstat	Specifies from which state vector the configurations are selected. This only applies to the case that the configurations were saved in a state-averaged calculation. If *refstat* is not specified, the configurations are selected from all states.
mxshrf	max. number of open shells in the selected or generated configurations.

19.4.3 Specifying orbital configurations

CON,$n_1,n_2,n_3,n_4,\ldots$

Specifies an orbital configuration to be included in the present symmetry. The first CON card must be preceded by a SELECT card. n_1, n_2 etc. are the occupation numbers of the active orbitals (0,1,or 2). For example, for
OCC,5,2,2;CLOSED,2,1,1;
n_1 is the occupation of orbital 3.1 (number.sym), n_2 is the occupation of orbital 4.1, n_3 of 5.1, n_4 of 2.2, and n_5 of 2.3 Any number of CON cards may follow each other.

Example for the BH molecule:

```
OCC,4,1,1;                ! four sigma, one pi orbitals are occupied
FROZEN,1;                 ! first sigma orbital is doubly occupied and frozen
```

```
WF,6,1;                ! 6 electrons, singlet Sigma+ state
SELECT                 ! triggers configuration input
CON,2,2                ! 2sigma**2, 3sigma**2
CON,2,1,1              ! 2sigma**2, 3sigma, 4sigma
CON,2,0,2              ! 2sigma**2, 4sigma**2
CON,2,0,0,2            ! 2sigma**2, 1pi_x**2
CON,2,0,0,0,2          ! 2sigma**2, 1pi_y**2
```

19.4.4 Selecting the primary configuration set

PSPACE,*thresh*

The hamiltonian is constructed and diagonalized explicitly in the primary configuration space, which can be selected with the PSPACE card. The coefficients of the remaining configurations (Q-space) are optimized iteratively using the P-space wavefunction as zeroth order approximation.

If *thresh* is nonzero, it is a threshold for automatically selecting all configurations as P-space configurations which have energies less then $emin + thresh$, where *emin* is the lowest energy of all configurations. Further P-space configurations can be specified using CON cards, which must follow immediately after the PSPACE card. These are merged with the ones selected according to the threshold. Automatic selection can be avoided by specifying a very small threshold. There is a sensible default value for thresh (0.4), so you usually don't need a pspace card in your input. Furthermore, if the number of configurations in the MCSCF is less than 20, all configurations go into the P-space unless you give a PSPACE card in the input.

A P-space threshold defined on a PSPACE card before the first WF (or STATE, WEIGHT, SELECT, PUNCSF if WF is not given) card is global, i.e., valid for all state symmetries. State-specific thresholds can be defined by placing a PSPACE card after the corresponding WF card. In the latter case the PSPACE card can be followed by CON cards, which define state-specific P-space configurations.

19.4.5 Projection to specific Λ states in linear molecules

Since MOLPRO can only use Abelian point groups (e.g. C_{2v} instead of $C_{\infty v}$ for linear molecules), $\Delta_{x^2-y^2}$ states as well as Σ^+ states occur in the irreducible representation number 1, for example. Sometimes it is not possible to predict in advance to which state(s) the program will converge. In such cases the LQUANT option can be used to specify which states are desired.

LQUANT,*lam(1),lam(2),...,lam(nstate)*;

lam(i) is the Λ quantum number of state *i*, i.e., 0 for Σ states, 1 for Π states, 2 for Δ states, etc. The matrix over Λ^2 will be constructed and diagonalized in the P-space configuration basis. The eigenvectors are used to transform the P-space hamiltonian into a symmetry adapted basis, and the program then selects the eigenvectors of the correct symmetry. The states will be ordered by symmetry as specified on the LQUANT card; within each symmetry, the states will be ordered according to increasing energy.

19.5 Restoring and saving the orbitals and CI vectors

MULTI normally requires a starting orbital guess. In this section we describe how to define these orbitals, and how to save the optimized orbitals. In a CASSCF calculation, one has the choice of transforming the final orbitals to natural orbitals (the first order density matrix is diagonalized), to pseudo-canonical orbitals (an effective Fock-operator is diagonalized), or of localizing the orbitals.

19.5.1 Defining the starting guess

`START`,*record,[options]*;

record: dump record containing starting orbitals. As usual, *record* has the form *irec.ifil*, where *irec* is the record number (e.g., 2140), and *ifil* the file number (usually 2). The *options* can be used to select orbitals of a specific type; for details, see section 4.11.

If this card is missing, the program tries to find suitable starting orbitals as follows:

First:	Try to read orbitals from the record specified on the `ORBITAL` card (or the corresponding default, see `ORBITAL`). All files are searched.
Second:	Try to find orbitals from the most recent MCSCF calculation. All files are searched.
Third:	Try to find orbitals from the most recent SCF calculation. All files are searched.

If no orbitals are found, a starting orbital guess is generated.

It is often useful to employ MCSCF orbitals from a neighbouring geometry as starting guess (this will happen automatically if orbitals are found, see the above defaults). Note, however, that frozen-core orbitals should always be taken from an SCF or MCSCF calculation at the present geometry and must be specified separately on the `FROZEN` card. Otherwise the program is likely to stop with error “non-orthogonal core orbitals”. The program remembers where to take the core orbitals from if these have been specified on a `FROZEN` card in a previous MCSCF calculation.

19.5.2 Rotating pairs of initial orbitals

`ROTATE`,*orb1.sym,orb2.sym,angle*

Performs a 2×2 rotation of the initial orbitals *orb1* and *orb2* in symmetry *sym* by *angle* degrees. With *angle*=0 the orbitals are exchanged. `ROTATE` is meaningful only after the `START` card. See `MERGE` for other possibilities to manipulate orbitals.

19.5.3 Saving the final orbitals

`ORBITAL`,*record.file*

The orbitals are dumped to record *record.file*. Default for *record* is 2140 and *file=2*. This default record number is incremented by one for each subsequent MCSCF calculation in the same job (see section 4.11). Therefore, if several different MCSCF calculations at several geometries are performed in one job, each MCSCF will normally start with appropriate orbitals even if no `ORBITAL` or `START` card is present.

The `ORBITAL` card can be omitted if a `NATORB, CANORB` or `LOCORB` card is present, since *orb* can also be specified on these cards (the same defaults for *orb* as above apply in these cases).

19.5.4 Saving the CI vectors and information for a gradient calculation

Old form (obsolete):

SAVE,*cidump,refsav,grdsav*;

New form:

SAVE,[CI=*cidump*,] [REF=*refsav*,] [GRD=*grdsav*];

This directive must be placed before any WF or STATE cards. The options can be given in any order.

cidump: record name for saving the CI vectors. By default, the vectors are only written to a scratch file. If NATORB, CANORB or LOCORB cards are present, *cidump* should be specified on these cards. At present, there is hardly any use of saved CI vectors, and therefore this option is rarely needed.

refsav: record name for saving the orbital configurations and their weights for use in subsequent MULTI or CI calculations using the SELECT directive. If wavefunctions for more than one state symmetry are optimized in a state-averaged calculation, the weights for each state symmetry are saved separately on records $refsav + (istsym - 1) * 100$, where *istsym* is the sequence number of the WF card in the input. If several NATORB, CANORB, or LOCORB cards are present, the record number is increased by 1000 for each subsequent orbital set. Note that this option implies the use of CSFs, even of no CONFIG card (see section 19.6.1) is present.

grdsav: record name for saving the information which is needed in a subsequent gradient calculation. This save is done automatically to record 5000.1 if the input contains a FORCE or OPTG card, and therefore the GRD option is normally not required.

19.5.5 Natural orbitals

NATORB,[*record*,] [*options*]

Request to calculate final natural orbitals and write them to record *record*. The default for *record* is 2140.2, or what else has been specified on an ORBITAL card, if present. By default, the orbitals are not printed and the hamiltonian is not diagonalized for the new orbitals The following *options* can be specified (in any order):

CI	Diagonalize the hamiltonian in the basis of the computed natural orbitals and print the configurations and their associated coefficients. This has the same effect as the GPRINT,CIVECTOR directive (see section 6.12. By default, only configurations with coefficients larger than 0.05 are printed. This threshold can be modified using the THRESH (see section 19.8.2) or GTHRESH (see section 6.11) options.
STATE=*state*	Compute natural orbitals for the specified state. *state* has the form *istate.isym*, e.g., 3.2 for the third state in symmetry 2. In contrast to earlier versions, *isym* refers to the number of the irreducible representation, and not the sequence number of the state symmetry. It is therefore independent of the order in which WF cards are given. The specified state must have been optimized. If STATE is not given and two or more states are averaged, the natural orbitals are calculated with the state-averaged density matrix (default).

`SPIN`=*ms2*	Compute natural orbitals for states with the specified spin. *ms2* equals $2 * S$, i.e., 0 for singlet, 1 for doublet etc. This can be used to together with `STATE` to select a specific state in case that states of different spin are averaged. If `STATE` is not specified, the state-averaged density for all states of the given spin is used.
`SAVE`=*record*	Request to save the civector(s) to the specified record.
`ORBITAL`=*record*	Request to save the orbitals to the specified record (same effect as specifying *record* as first agrument (see above).
`PRINT`=*nvirt*	Request to print *nvirt* virtual orbitals in each symmetry. By default, the orbitals are not printed unless the `ORBPRINT` option (see section 19.8.1 is present or the global `GPRINT,ORBITALS` (see section 6.12) directive has been given before. The `PRINT` option on this card applies only to the current orbitals.

Several `NATORB`, `CANORB`, and `LOCORB` cards (for different states) may follow each other. In contrast to earlier versions of MOLPRO the different orbital sets can all be stored in one dump record (but different records still work). See section 4.11 for information about dump records and how specific orbital sets can be requested in a later calculation.

19.5.6 Pseudo-canonical orbitals

`CANORB`,[*record*,] [*options*]

or

`CANONICAL`,[*record*,] [*options*]

Request to canonicalize the final orbitals, and writing them to record *record*. All options have the same effect as described for `NATORB`.

19.5.7 Localized orbitals

`LOCORB`,[*record*,] [*options*]

or

`LOCAL`,[*record*,] [*options*]

Request to localize the final orbitals, and writing them to record *record*. All options have the same effect as described for `NATORB`.

Note: `LOCAL` is interpreted by MULTI, but `LOCALI` is a separate command which calls the localization program and not recognized by MULTI. In order to avoid confusion, it is recommended to use `LOCORB` rather then `LOCAL` as subcommand within MULTI.

19.5.8 Diabatic orbitals

In order to construct diabatic states, it is necessary to determine the mixing of the diabatic states in the adiabatic wavefunctions. In principle, this mixing can be obtained by integration of the non-adiabatic coupling matrix elements. Often, it is much easier to use an approximate method, in which the mixing is determined by inspection of the CI coefficients of the MCSCF or CI wavefunctions. This method is applicable only if the orbital mixing is negligible. For CASSCF wavefunctions this can be achieved by maximizing the overlap of the active orbitals with those of a reference geometry, at which the wavefunctions are assumed to be diabatic (e.g. for symmetry reasons). The orbital overlap is maximized using using the new `DIAB` command in the `MCSCF` program. Only the active orbitals are transformed.

This procedure works as follows: first, the orbitals are determined at the reference geometry. Then, the calculations are performed at displaced geometries, and the "diabatic" active orbitals, which have maximum overlap with the active orbitals at the reference geometry, are obtained by adding a `DIAB` directive to the input:

Old form (`Molpro96`, obsolete):

`DIAB`,*orbref, orbsav, orb1,orb2,pri*

New form:

`DIAB`,*orbref* [,`TYPE`=*orbtype*] [,`STATE`=*state*] [,`SPIN`=*spin*] [,*MS2=ms2*] [,`SAVE`=*orbsav*] [,`ORB1`=*orb1*, `ORB2`=*orb2*] [,*PRINT=pri*] [,*METHOD=method*]

Here *orbref* is the record holding the orbitals of the reference geometry, and *orbsav* is the record on which the new orbitals are stored. If *orbsav* is not given (recommended!) the new orbitals are stored in the default dump record (2140.2) or the one given on the `ORBITAL` directive (see section 19.5.3). In contrast to earlier versions of MOLPRO it is possible that *orbref* and *orbsav* are the same. The specifications `TYPE`, `STATE`, `SPIN` can be used to select specific sets of reference orbitals, as described in section 4.11. *orb1, orb2* is a pair of orbitals for which the overlap is to be maximized. These orbitals are specified in the form *number.sym*, e.g. 3.1 means the third orbital in symmetry 1. If *orb1, orb2* are not given, the overlap of all active orbitals is maximized. *pri* is a print parameter. If this is set to 1, the transformation angles for each orbital are printed for each Jacobi iteration. *method* determines the diabatization method. *method=1* (default): use Jacobi rotations; *method=2*: use block diagonalization. Both methods yield very similar results. *method=2* must only be used for CASSCF wavefunctions. *method=-1* and *method=-2*: as the positive values, but AO overlap matrix of the current geometry is used. This minimizes the change of the MO coefficients, rather than maximizing the overlap to the neighbouring orbitals.

Using the defaults described above, the following input is sufficient in most cases:

`DIAB`,*orbref*

Using `Molpro98` is is not necessary any more to give any `GEOM` and `DISPL` cards. The displacements and overlap matrices are computed automatically (the geometries are stored in the dump records, along with the orbitals).

The diabatic orbitals have the property that the sum of orbital and overlap contributions in the non-adiabatic coupling matrix elements become approximately zero, such that the adiabatic mixing occurs only through changes of the CI coefficients. This allows to determine the mixing angle directly from the CI coefficients, either in a simple way as described for instance in J. Chem. Phys. **89**, 3139 (1988), or in a more advanced manner as described by Pacher, Cederbaum, and Köppel in J. Chem. Phys. **89**, 7367 (1988). Recently, an automatic procedure, as described in J. Chem. Phys. **102**, 0000, (1999) has been implemented into MOLPRO. This is available in Version 99.1 and later and is described in section 38.

Below we present an example for the first two excited states of H_2S, which have B_1 and A_2 symmetry in C_{2v}, and A'' symmetry in C_S. We first perform a reference calculation in C_{2v} symmetry, and then determine the diabatic orbitals for displaced geometries in C_S symmetry. Each subsequent calculation uses the previous orbitals as reference. One could also use the orbitals of the C_{2v} calculation as reference for all other calculations. In this case one would have to take out the second-last input card, which sets `reforb=2141.2`.

```
***,H2S diabatic A" states

basis=VDZ                              !use cc-pVDZ basis set
symmetry,x,planeyz                     !use Cs symmetry & fix orientation of the molecule
orient,noorient                        !dont allow automatic reorientation
geometry={s;h1,s,r1;h2,s,r2,h1,theta}  !Z-matrix geometry input

gprint,orbitals,civector               !global print options

text,reference calculation for C2V
theta=92.12,r1=2.3,r2=2.3              !reference geometry

{hf;occ,7,2;wf,18,1}                   !scf calculation for ground state

{multi;occ,9,2;closed,4,1;             !define active and inactive spaces
wf,18,2;state,2;                       !two A" states (1B1 and 1A2 in C2v)
orbital,2140.2}                        !save orbitals to 2140.2
reforb=2140.2

text,calculations at displaced geometries

rd=[2.4,2.5,2.6]                       !define a range of bond distances

do i=1,#rd                             !loop over displaced geometries

r2=rd(i)                               !set r2 to current distance

{multi;occ,9,2;closed,4,1;             !same wavefunction definition as at reference geom.
wf,18,2;state,2;
orbital,2141.2                         !save new orbitals to record
diab,reforb}                           !compute diabatic orbitals using reference orbitals
                                       !stored on record reforb
reforb=2141.2                          !set variable reforb to the new orbitals.
enddo
```

http://www.molpro.net/info/current/examples/h2s_diab.com

See section 38 for the automatic generation of diabatic energies.

19.6 Selecting the optimization methods

By default, MULTI uses the non-linear optimization method developed by Werner, Meyer, and Knowles. Other methods, such as the Newton-Raphson procedure or the Augmented Hessian procedure, are also implemented and can be selected using the `ITERATIONS` directive (for state-averaged calculations, only the non-linear optimization method can be used). For CASSCF calculations, the CI problem is solved in a basis of Slater determinants, unless a `CONFIG` card is given. Some procedures may be disabled using the `DONT` directive.

19.6.1 Selecting the CI method

`CONFIG`,*key*;

key may be `DET` or `CSF`, and defaults to `CSF`. If no `CONFIG` or `SELECT` card is given, the default is determinants (CASSCF).

19.6.2 Selecting the orbital optimization method

The `ITERATIONS` directive can be use to modify the defaults for the optimization method. It consists of a sequence of several cards, which should be enclosed in a pair of curly brackets.

{ `ITERATIONS`;
`DO`,*method1,iter1*[,`TO`,*iter2*];
`DONT`,*method2,iter3*[,`TO`,*iter4*];
...
}

method can be one of the following:

`DIAGCI`	Diagonalize hamiltonian in the beginning of the specified iterations. This is the default for iteration 1.
`INTERNAL`	Optimize internal orbitals at the beginning of the specified iterations. This is default for second and subsequent iterations.
`WERNER`	use Werner-Meyer-Knowles non-linear optimization method for the specified iterations. This is the default for all iterations.
`AUGMENT`	Use step-restricted Augmented Hessian method for the specified iterations.
`NEWTON`	Use Newton-Raphson method for specified iterations.
`UNCOUPLE`	Do not optimize orbitals and CI coefficients simultaneously in the specified iterations. This option will set DIAGCI for these iterations.
`NULL`	No orbital optimization.

19.6.3 Disabling the optimization

In addition to the `ITERATIONS` directive described above, some procedures can be be disabled more simply using the `DONT` directive. `DONT`,*code*

code may be

`ORBITAL`	Do initial CI but don't optimize orbitals.
`WAVEFUNC`	Do not optimize the orbitals and CI coefficients (i.e. do only wavefunction analysis, provided the orbitals and CI coefficients are supplied (see `START card`)).
`WVFN`	Alias for WAVEFUNC.
`ANAL`	Do no wavefunction analysis.

19.6.4 Disabling the extra symmetry mechanism

`NOEXTRA`

This card disables the search for extra symmetries. By default, if extra symmetries are present, each orbital is assigned to such an extra symmetry and rotations between orbitals of different extra symmetry are not performed.

19.7 Calculating expectation values

By default, the program calculates the dipole expectation and transition moments. Further expectation values or transition properties can be computed using the `TRAN, TRAN2` and `EXPEC, EXPEC2` directives.

19.7.1 Matrix elements over one-electron operators

`EXPEC`,$oper_1, oper_2, \ldots, oper_n$
`TRAN`,$oper_1, oper_2, \ldots, oper_n$

Calculate expectation values and transition matrix elements for the given one-electron operators. With `EXPEC` only expectation values are calculated. $oper_i$ is a codeword for the operator. The available operators and their associated keywords are given in section 6.13.

19.7.2 Matrix elements over two-electron operators

`EXPEC2`,$oper_1, oper_2, \ldots, oper_n$
`TRAN2`,$oper_1, oper_2, \ldots, oper_n$

Calculate transition matrix elements for two-electron operators. This is presently only useful for angular momentum operators. With `EXPEC2` only diagonal matrix elements will be computed. For instance

`TRAN2,LXX`	calculates matrix elements for L_x^2
`TRAN2,LYY`	calculates matrix elements for L_y^2
`TRAN2,LXZ`	calculates matrix elements for $\frac{1}{2}(L_xL_z + L_zL_x)$
`TRAN2,LXX,LYY,LZZ`	calculates matrix elements for L_x^2, L_y^2, and L_z^2. The matrix elements for the sum L^2 are also printed.

19.7.3 Saving the density matrix

`DM`,[*spindens*]

If the `DM` directive is given, the first order density matrix in AO basis is written to the dump record specified on the `ORBITAL` card (default 2140.2). If no `ORBITAL` card is present, but a record is specified on a `NATORB`, `CANORB`, or `LOCORB` card, the densities are saved to the first record occurring in the input. In a state-averaged calculation the SA-density, as well the individual state densities, are saved. See section 4.11 for information about how to recover any of these densities for use in later programs.

Of *spindens* is a number greater than zero, the spin density matrices are also saved. Note that a maximum of 50 density matrices can be saved in one dump record.

If no `DM` directive is given), the first order density matrix is saved in single-state calculations, and only the stage-averaged density matrix in state-averaged calculations.

19.8 Miscellaneous options

All commands described in this section are optional. Appropriate default values are normally used. Note that printing of the orbitals and civectors can also be requested using the global `GPRINT` command, or by giving `NATORB` or `CANORB` options.

19.8.1 Print options

`ORBPRINT`[,*nvirt*]

requests the occupied and *nvirt* virtual orbitals in each symmetry to be printed (default *nvirt*=0). By default, the program does not print the orbitals, unless the `ORBPRINT` directive or a global `GPRINT,ORBITALS` (see section 6.12) command is present. Specific orbital sets can be printed using the `PRINT` option on a `NATORB`, `CANORB`, or `LOCORB` card (see section 19.5.5). To print additional information at the end of the calculation, use

`PRINT`,*key1*,*key2*,...;

Printing is switched on for *key1*, *key2*,.... To print information in each iteration, use

`IPRINT`,*key1*,*key2*,...;

Possible print keys are:

`MICRO`	print details of "microiterations" — useful for finding out what's going wrong if no convergence
`REF`	print summary of configuration set (CSFs only)
`REF1`	print list of configuration set (CSFs only)
`COR`	print summary of intermediate spaces used in CSF calculation
`COR1`	print list of intermediate configuration sets (CSFs only)
`PSPACE`	print list of configurations making up the "primary" space
`ORBITALS`	print orbitals (see also `ORBPRINT`)
`NATORB`	print natural orbitals (see also `ORBPRINT`)
`VIRTUALS`	print virtual orbitals (see also `ORBPRINT`)
`CIVECTOR`	print CI vector (better use `CANORB` or `NATORB`)
`INTEGRAL`	print transformed integrals (for testing only!)
`DENSITY`	print density matrices
`HESSIAN`	print hessian
`DIAGONAL`	print diagonal elements of hessian
`GRADIENT`	print gradient
`LAGRANGI`	print Lagrangian
`STEP`	print update vector
`ADDRESS`	print addressing information (for testing only!)
`DEBUG`	print debugging information
`CI2`	print debugging information in routine ci2 (Warning: may be long!!)
`IO`	print debugging information in I/O routines

19.8.2 Convergence thresholds

Convergence thresholds can be modified using

`ACCURACY`,[`GRADIENT`=*conv*] [,`STEP`=*sconv*] [,`ENERGY`=*econv*]

where

conv	Threshold for orbital gradient (default 10^{-2}).)
econv	Threshold for change of total energy (default 10^{-6}).
sconv	Threshold for size of step (default 10^{-3}).

The default values can be modified using the global GTHRESH command (see section 6.11). Normally, the above default values are appropriate.

19.8.3 Maximum number of iterations

MAXITER,*maxit*;

maxit is maximum number of iterations (default 20). If the calculation does not converge in the default number of iterations, you should first think about the reason before increasing the limit. In most cases the choice of active orbitals or of the optimized states is not appropriate (see introduction of MULTI). The maximum allowed value of *maxit* is 40. If the calculation has not converged in this number of iterations, it is likely that the active space is not reasonable. Note: slow convergence can occur in RASSCF calculations if single excitations into weakly occupied orbitals are present. These should be eliminated using

RESTRICT,-1,-1,*orbital list*

19.8.4 Test options

TEST,$i1,i2,i3,\ldots$;

Activate testing options numbered $i1$, $i2$, ... Please do not use unless you know what you are doing!

19.8.5 Special optimization parameters

The following parameters can also be given as options on the MULTI command line.

STEP,*radius,trust1,tfac1,trust2,tfac2*;

Special parameters for augmented hessian method. For experts only!

GOPER,*igop*;

Use G-operator technique in microiterations (Default). If igop.lt.0 do not use G-operators.

COPT,*ciacc,copvar,maxci,cishft,icimax,icimx1,icimx2,icstrt,icstep*;

Special parameters for the direct CI method. For experts only!

ciacc	grad threshold for CI diagonalization
copvar	start threshold for CI-optimization
maxci	max. number of CI-optimizations per microiteration
cishft	denominator shift for q-space
icimax	max. number of CI-optimizations in first macroiteration
icimx1	max. number of CI-optimizations in second and subsequent iterations
icimx2	max. number of CI-optimizations in internal absorption step

icstrt	first microiteration with CI-optimization
icstep	microiteration increment between CI-optimizations

INTOPT,*maxito,maxitc,maxrep,nitrep,iuprod*;

Special parameters for internal optimization scheme. For experts only!

NONLINEAR,*itmaxr,ipri,drmax,drdamp,gfak1,gfak2,gfak3,irdamp,ntexp*

Special parameters for non-linear optimization scheme. For experts only!

Old form (obsolete):

THRESH,*thrpri,thrpun,varmin,varmax,thrdiv,thrdoub*

New form:

THRESH [,THRPRI=*thrpri*] [,THRPUN=*thrpun*] [,VARMIN=*varmin*]
[,VARMAX=*varmax*] [,THRDIV=*thrdiv*] [,THRDOUB=*thrdoub*]

thrpri	threshold for printing CI coefficients (default 0.04)
thrpun	threshold for writing CI coefficients to the punch file. Default is no write to the punch file
varmin,varmax	thresholds for non-linear optimization scheme. For experts only!
thrdoub	threshold for detecting almost doubly occupied orbitals for inclusion into the pseudo canonical set (default 0, i.e. the feature is disabled).

DIIS,*disvar,augvar,maxdis,maxaug,idsci,igwgt,igvec,idstrt,idstep*;

Special parameters for DIIS convergence acceleration. For experts only!

19.8.6 Saving wavefunction information for CASVB

VBDUMP[,*vbdump*];

For users of the valence bond program *CASVB*, all wavefunction information that may subsequently be required is saved to the record *vbdump*. The default is not to write this information. If the keyword is specified without a value for *vbdump*, then record 4299.2 is used. This keyword is not needed prior to variational *CASVB* calculations.

19.8.7 Saving transformed integrals

TRNINT,*trnint*;

trnint specifies the record name for integrals in the basis of active CASSCF MOs. These are used for example by *CASVB* (see section 39.5). The default value for *trnint* is 1900.1.

19.9 Coupled-perturbed MCSCF

The coupled-perturbed MCSCF is required for computing gradients with state-averaged orbitals, non-adiabatic couplings, difference gradients or polarizabilities. We note that the present implementation is somewhat preliminary and not very efficient.

19.9.1 Gradients for SA-MCSCF

For computing state-averaged gradients, use

CPMCSCF,GRAD,*state*,[SPIN=*spin*],[MS2=*ms2*],[ACCU=*thresh*],[RECORD=*record*]

where *state* specifies the state (e.g., 2.1 for the second state in symmetry 1) for which the gradients will computed. *spin* specifies the spin of the state: this is half the value used in the corresponding WF card (e.g., 0=Singlet, 0.5=Doublet, 1=Triplet). Alternatively, MS2 can be used, where *ms2* = 2*spin, i.e., the same as specified on WF cards. The specification of SPIN or MS2 is only necessary if states with different spin are state-averaged. *record* specifies a record on which the gradient information is stored (the default is 5101.1). *thresh* is a threshold for the accuracy of the CP-MCSCF solution. The default is 1.d-7.

The gradients are computed by a subsequent call to FORCES or OPTG.

Note: if for some reason the gradients are to be computed numerically from finite energy differences, it is in state-averaged calculations necessary to give, instead of the CPMCSCF input, the following:

SAVE,GRAD=-1

Otherwise the program will stop with an error message.

19.9.2 Difference gradients for SA-MCSCF

For computing difference gradients, use

CPMCSCF,DGRAD,*state1*,*state2*,[ACCU=*thresh*],[RECORD=*record*]

where *state1* and *state2* specify the two states considered. (e.g., 2.1,3.1 for the second and third states in symmetry 1) The gradient of the energy difference will be computed. Both states must have the same symmetry. *record* specifies a record on which the gradient information is stored (the default is 5101.1). *thresh* is a threshold for the accuracy of the CP-MCSCF solution. The default is 1.d-7.

The gradients are computed by a subsequent call to FORCES or OPTG.

19.9.3 Non-adiabatic coupling matrix elements for SA-MCSCF

For computing non-adiabatic coupling matrix elements analytically, use

CPMCSCF,NACM,*state1*,*state2*,[ACCU=*thresh*],[RECORD=*record*]

where *state1* and *state2* specify the two states considered. (e.g., 2.1,3.1 for the second and third states in symmetry 1) Both states must have the same symmetry. *record* specifies a record on which the gradient information is stored (the default is 5101.1). This will be read in the subsequent gradient calculation. *thresh* is a threshold for the accuracy of the CP-MCSCF solution. The default is 1.d-7.

NADC and NADK are an aliases for NADC, and SAVE is an alias for RECORD.

The matrix elements for each atom are computed by a subsequent call to FORCES.

Note: this program is not yet extensively tested and should be used with care!

19.9.4 MCSCF hessians

The MCSCF/CASSCF hessian can be computed analytically (only without symmetry) using

`CPMCSCF,HESS,[ACCU=`*value*`]`

where the `ACCU` option specifies the convergence threshold in the CPMCSCF calculation (default 1.d-4). The hessian is stored on record 5300.2 and can be used in a subsequent frequency calculation.

Example:

```
{multi;cpmcscf,hess,accu=1.d-5}
frequencies
```

Note that the `NOEXTRA` option is used when computing a hessian.

19.10 Optimizing valence bond wavefunctions

`VB={...}`

Using this keyword, the optimization of the CI coefficients is carried out by *CASVB*. The `VB` keyword can be followed by any of the directives described in section 39. Energy-based optimization of the VB parameters is the default, and the output level for the main *CASVB* iterations is reduced to -1.

19.11 Hints and strategies

MCSCF is not a "black box" procedure like SCF! For simple cases, for example a simple CASSCF with no `CLOSED` orbitals, this program will converge in two or three iterations. For more complicated cases, you may have more trouble. In that case, consider the following:

- Always start from neighbouring geometry orbitals when available (this is the default).
- The convergence algorithm is more stable when there are no `CLOSED` orbitals, i.e., orbitals doubly occupied in all configurations, but fully optimized. Thus a reasonable approach is to make an initial calculation with `CLOSED` replaced by `FROZEN` (all doubly occ. frozen).
- If still no success, you can switch off the coupling between CI coefficients and orbital rotations for a few iterations, e.g.:

  { `ITERATIONS;DO,UNCOUPLE,1,TO,2;` }

  and/or disable the simultaneous optimization of internal orbitals & CI, e.g.:

  { `ITERATIONS;DONT,INTERNAL,1,TO,2;` }

You can often get a clue about where the program starts to diverge if you include:

`IPRINT,MICRO;`

in the data. Also consider the general remarks at the beginning of this chapter. For the details of the algorithms used, see J. Chem. Phys 82, 5053 (1985); Chem. Phys. Letters 115, 259 (1985); Advan. Chem. Phys. 59, 1 (1987);

19.12 Examples

The simplest input for a CASSCF calculation for H_2O, C_{2v} symmetry, is simply:

```
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
hf                                    !do scf calculation
multi                                 !do full valence casscf
```

http://www.molpro.net/info/current/examples/h2o_casscf.com

This could be extended, for instance, by the following input cards

```
OCC,4,1,2;        !  specify occupied space
CLOSED,2          !  specify closed-shell (inactive) orbitals
FROZEN,1;         !  specify frozen core orbitals
WF,10,1;          !  define wavefunction symmetry
START,2100.2;     !  read guess orbitals from record 2100, file 2
ORBITAL,2140.2;   !  save final orbitals to record 2140, file 2
NATORB,PRINT,CI   !  print natural orbitals and diagonalize the hamiltonian
                  !  for the natural orbitals. The largest CI coefficients
                  !  are printed.
```

Example for a state-averaged calculation for CN, X and B $^2\Sigma^+$ states, and A $^2\Pi_x$, $^2\Pi_y$ states averaged. A full valence CASSCF calculation is performed

```
***,cn
r=2.2                                !define bond length
geometry={c;n,c,r}
{rhf;occ,5,1,1;wf,13,1,1;             !RHF calculation for sigma state
orbital,2100.2}                      !save orbitals to record 2100.2 (default)

{multi;occ,6,2,2;closed,2;            !Define active and inactive orbitals
start,2100.2;                        !Start with RHF orbitals from above
save,ref=4000.2                      !Save configuration weights for CI in record 4000.2
wf,13,1,1;state,2;wf,13,2,1;wf,13,3,1;!Define the four states
natorb,ci,print;                     !Print natural orbitals and associated ci-coefficients
tran,lz                              !Compute matrix elements over LZ
expec2,lzz}                          !compute expectation values for LZZ
```

http://www.molpro.net/info/current/examples/cn_sa_casscf.com

Example for an RASSCF (restricted active space) calculation for N_2, including SCF determinant plus all double excitations into valence orbitals. The single excitations are excluded. D_{2h} symmetry, CSF method used:

```
***,N2
geometry={N1;N2,N1,r}                  !geometry input
r=2.2                                  !bond length
{hf;occ,3,1,1,,2;wf,14,1;save,2100.2} !scf calculation

{multi;occ,3,1,1,,3,1,1;               !Define occupied orbitals
frozen,1,,,,1,2100.2;                    !Define frozen core scf orbitals
config;                                !Use CSF method
wf,14,1;                               !Define state symmetry
restrict,0,2,3.5,1.6,1.7;              !Restriction to singles and doubles
restrict,-1,-1,3.5,1.6,1.7;            !Take out singles
print,ref1                             !Print configurations
natorb,ci,print}                       !Print natural orbitals and CI coeffs
```

http://www.molpro.net/info/current/examples/n2_rasscf.com

20 THE CI PROGRAM

Multiconfiguration reference internally contracted configuration interaction

Bibliography:

H.-J. Werner and P.J. Knowles, J. Chem. Phys. 89, 5803 (1988).
P.J. Knowles and H.-J. Werner, Chem. Phys. Lett. 145, 514 (1988).

All publications resulting from use of this program must acknowledge the above. See also:

H.-J. Werner and E.A. Reinsch, J. Chem. Phys. 76, 3144 (1982).
H.-J. Werner, Adv. Chem. Phys. 59, 1 (1987).

The command `CI` or `CI-PRO` calls the program. The command `CISD` calls fast closed-shell CISD program. The command `QCI` calls closed-shell quadratic CI program. The command `CCSD` calls closed-shell coupled-cluster program.

The following options may be specified on the command line:

`NOCHECK`	Do not stop if no convergence.
`DIRECT`	Do calculation integral direct.
`NOSING`	Do not include singly external configurations.
`NOPAIR`	Do not include doubly external configurations (not valid for single reference methods).
`MAXIT=`*value*	Maximum number of iterations.
`MAXITI=`*value*	Maximum number of microiterations (for internals).
`SHIFTI=`*value*	Denominator shift for update of internal configurations.
`SHIFTS=`*value*	Denominator shift for update of singles.
`SHIFTP=`*value*	Denominator shift for update of doubles.
`THRDEN=`*value*	Convergence threshold for the energy.
`THRVAR=`*value*	Convergence threshold for the CI-vector. This applies to the square sum of the changes of the CI-coefficients.
`SWAP|NOSWAP`	If `SWAP` is given, the MRCI wavefunctions are reordered according to maximum overlap with the reference functions (this only applies in multi-state calculations). The default is `NOSWAP`, i.e. the states are ordered according to increasing MRCI energy.

ROTREF=*value*	If *value*=0 the cluster corrections are not printed for the rotated reference energies (cf. Section 20.7). If *value*=1 all corrections are printed. If *value*=-1 the 2009.1 behaviour is recovered.

20.1 Introduction

The internally contracted MRCI program is called by the CI command. This includes as special cases single reference CI, CEPA, ACPF, MR-ACPF and MR-AQCC. For closed-shell reference functions, a special faster code exists, which can be called using the CISD, QCI, or CCSD commands. This also allows to calculate Brueckner orbitals for all three cases (QCI and CCSD are identical in this case).

With no further input cards, the wavefunction definition (core, closed, and active orbital spaces, symmetry) corresponds to the one used in the most recently done SCF or MCSCF calculation. By default, a CASSCF reference space is generated. Other choices can be made using the OCC, CORE, CLOSED, WF, SELECT, CON, and RESTRICT cards. The orbitals are taken from the corresponding SCF or MCSCF calculation unless an ORBITAL directive is given. The wavefunction may be saved using the SAVE directive, and restarted using START. The EXPEC directive allows to compute expectation values over one-electron operators, and the TRAN directive can be used to compute transition matrix elements for one-electron properties. Natural orbitals can be printed and saved using the NATORB directive.

For excited state calculations see STATE, REFSTATE, and PROJECT.

20.2 Specifying the wavefunction

Note: All occupations must be given before WF, PAIRSS, DOMAIN, REGION or other directives that need the occupations.

20.2.1 Occupied orbitals

OCC,$n_1, n_2, \ldots, n_8$;

n_i specifies numbers of occupied orbitals (including CORE and CLOSED) in irreducible representation number i. If not given, the information defaults to that from the most recent SCF, MCSCF or CI calculation.

20.2.2 Frozen-core orbitals

CORE,$n_1, n_2, \ldots, n_8$;

n_i is the number of frozen-core orbitals in irrep number i. These orbitals are doubly occupied in all configurations, i.e., not correlated. If no CORE card is given, the program uses the same core orbitals as the last CI calculation; if there was none, then the atomic inner shells are taken as core. To avoid this behaviour and correlate all electrons, specify

CORE

20.2.3 Closed-shell orbitals

CLOSED,$n_1,n_2,\ldots,n_8$

n_i is the number of closed-shell orbitals in irrep number i, inclusive of any core orbitals. These orbitals do not form part of the active space, i.e., they are doubly occupied in all reference CSFs; however, in contrast to the core orbitals (see CORE), these orbitals are correlated through single and double excitations. If not given, the information defaults to that from the most recent SCF, MCSCF or CI calculation. For calculations with closed-shell reference function (closed=occ), see CISD, QCI, and CCSD.

20.2.4 Defining the orbitals

ORBIT,*name.file*,[*specifications*];

name.file specifies the record from which orbitals are read. Optionally, various *specifications* can be given to select specific orbitals if *name.file* contains more than one orbital set. For details see section 4.11. Note that the IGNORE_ERROR option can be used to force MPn or triples calculations with non-canonical orbitals.

The default is the set of orbitals from the last SCF, MCSCF or CI calculation.

20.2.5 Defining the state symmetry

The number of electrons and the total symmetry of the wavefunction are specified on the WF card:

WF,*elec,sym,spin*

where

elec:	is the number of electrons
sym:	is the number of the irreducible representation
spin:	defines the spin symmetry, *spin*$=2S$ (singlet=0, doublet=1, triplet=2, etc.)

The WF card must be placed after any cards defining the orbital spaces OCC, CORE, CLOSED.

The REF card can be used to define further reference symmetries used for generating the configuration space, see REF.

20.2.6 Additional reference symmetries

REF,*sym*;

This card, which must come after the WF directive, defines an additional reference symmetry used for generating the uncontracted internal and singly external configuration spaces. This is sometimes useful in order to obtain the same configuration spaces when different point group symmetries are used. For instance, if a calculation is done in C_s symmetry, it may happen that the two components of a Π state, one of which appears in A' and the other in A'', come out not exactly degenerate. This problem can be avoided as in the following example:

for a doublet A' state:

```
WF,15,1,1;       !define wavefunction symmetry (1)
REF,2;           !define additional reference symmetry (2)
```

and for the doublet A” state:

```
WF,15,2,1;       !define wavefunction symmetry (2)
REF,1;           !define additional reference symmetry (1)
```

For linear geometries the same results can be obtained more cheaply using C_{2v} symmetry,

```
WF,15,2,1;       !define wavefunction symmetry (2)
REF,1;           !define additional reference symmetry (1)
REF,3;           !define additional reference symmetry (3)
```

or

```
WF,15,3,1;       !define wavefunction symmetry (2)
REF,1;           !define additional reference symmetry (1)
REF,2;           !define additional reference symmetry (2)
```

Each `REF` card may be followed by `RESTRICT`, `SELECT`, and `CON` cards, in the given order.

20.2.7 Selecting configurations

`SELECT`,*ref1,ref2,refthr,refstat,mxshrf*;

This card is used to specify a reference configuration set other than a CAS, which is the default. Configurations can be defined using `CON` cards, which must appear after the *SELECT* card. Alternatively, if *ref1* is an existing MOLPRO record name, the configurations are read in from that record and may be selected according to a given threshold.

The select card must be placed directly after the *WF* or *REF* card(s), or, if present, the `RESTRICT` cards. The general order of these cards is

`WF` (or `REF`)
`RESTRICT` (optional)
`SELECT` (optional)
`CON` (optional)

ref1=rec1.file: (*rec1>2000*) The configurations are read in from the specified record. See section 19.5.4 about how to save the configurations in the MCSCF calculation. If *ref1* is not specified, the program assumes that the configurations are read from subsequent `CON` cards (see `CON`).

ref2=rec2.file: (*rec2>2000*) additional configurations are read from the specified record. If *rec2* is negative, all records between *rec1* and abs(*rec2*) are read. All configurations found in this way are merged.

refthr: Selection threshold for configurations read from disc (records *rec1–rec2*). This applies to the norm of all CSFs for each orbital configuration.

refstat: Specifies from which state vector the configurations are selected. This only applies to the case that the configurations were saved in a state-averaged calculation. If *refstat* is zero or not specified, the configurations are selected from all states. If *refstat* is greater than zero, then the specified reference state is used. If *refstat* is less than zero, then all appropriate reference states are used. Lastly, if *refstat* is of the form *istat1.istat2*, states *istat1* through *istat2* are used.

mxshrf: maximum number of open shells in the selected or generated configurations.

20.2.8 Occupation restrictions

`RESTRICT`,*nmin,nmax,*orb_1,orb_2,$\ldots orb_n$;

This card can be used to restrict the occupation patterns in the reference configurations. Only configurations containing between *nmin* and *nmax* electrons in the specified orbitals orb_1, orb_2, $\ldots$, orb_n are included in the reference function. If *nmin* and *nmax* are negative, configurations with exactly abs(*nmin*) and abs(*nmax*) electrons in the specified orbitals are deleted. This can be used, for instance, to omit singly excited configurations. The orbitals are specified in the form *number.sym*, where *number* is the number of the orbital in irrep *sym*. Several `RESTRICT` cards may follow each other.

The `RESTRICT` cards must follow the `WF` or `REF` cards to which they apply. The general order of these cards is

`WF` (or `REF`)
`RESTRICT` (optional)
`SELECT` (optional)
`CON` (optional)

If a `RESTRICT` cards precedes the `WF` card, it applies to all reference symmetries. Note that `RESTRICT` also affects the spaces generated by `SELECT` and/or `CON` cards.

20.2.9 Explicitly specifying reference configurations

`CON`,$n_1, n_2, n_3, n_4, \ldots$

Specifies an orbital configuration to be included in the reference function. n_1, n_2 etc. are the occupation numbers of the active orbitals (0,1,or 2). Any number of `CON` cards may follow each other, but they must all appear directly after a `SELECT` card.

20.2.10 Defining state numbers

`STATE`,*nstate,nroot(1),nroot(2),...,nroot(nstate)*;

nstate is the number of states treated simultaneously; *nroot(i)* are the root numbers to be calculated. These apply to the order of the states in the initial internal CI. If not specified, *nroot(i)=i*. Note that it is possible to leave out states, i.e.,

```
STATE,1,2;     ! calculates second state
STATE,2,1,3;   ! calculates first and third state
```

All states specified must be reasonably described by the internal configuration space. It is possible to have different convergence thresholds for each state (see `ACCU` card). It is also possible not to converge some lower roots which are included in the list *nroot(i)* (see `REFSTATE` card). For examples, see `REFSTATE` card.

20.2.11 Defining reference state numbers

`REFSTATE`,*nstatr,nrootr(1),nrootr(2),...,nrootr(nstatr)*;

nstatr is the number of reference states for generating contracted pairs. This may be larger or smaller than *nstate*. If this card is not present, *nstatr*=*nstate* and *nrootr(i)*=*nroot(i)*. Roots for which no reference states are specified but which are specified on the `STATE` card (or included by default if the *nroot(i)* are not specified explicitly on the `STATE` card) will not be converged, since the result will be bad anyway. However, it is often useful to include these states in the list *nroot(i)*, since it helps to avoid root flipping problems. Examples:

```
state,2;
```

will calculate two states with two reference states.

```
state,2;refstate,1,2;
```

will optimize second state with one reference state. One external expansion vector will be generated for the ground state in order to avoid root flipping. The results printed for state 1 are bad and should not be used (unless the pair space is complete, which might happen in very small calculations).

```
state,1,2;refstate,1,2;
```

As the second example, but no external expansion vectors will be generated for the ground state. This should give exactly the same energy for state 2 as before if there is no root flipping (which, however, frequently occurs).

```
state,2;accu,1,1,1;
```

Will calculate second state with two reference states. The ground state will not be converged (only one iteration is done for state 1) This should give exactly the same energy for state 2 as the first example.

20.2.12 Specifying correlation of orbital pairs

`PAIR`,*iorb1.isy1,iorb2.isy2,np*;

is a request to correlate a given orbital pair.

np=1:	singlet pair
np=-1:	triplet pair

np=0: singlet and triplet pair (if possible)

Default is to correlate all electron pairs in active and closed orbitals. See also `PAIRS` card.

`PAIRS`,*iorb1.isy,iorb2.isy,np*;

Correlate all pairs which can be formed from orbitals *iorb1.isy1* through *iorb2.isy2*. Core orbitals are excluded. Either *iorb2* must be larger than *iorb1* or *isy2* larger than *isy1*. If *iorb1.isy1=iorb2.isy2* the `PAIRS` card has the same effect as a `PAIR` card. `PAIR` and `PAIRS` cards may be combined.

If no `PAIR` and no `PAIRS` card is specified, all valence orbitals are correlated. The created pair list restricts not only the doubly external configurations, but also the all internal and semi internals.

20.2.13 Restriction of classes of excitations

`NOPAIR;`

No doubly external configurations are included.

`NOSINGLE;`

No singly external configurations are included.

`NOEXC;`

Perform CI with the reference configurations only.

20.3 Options

20.3.1 Coupled Electron Pair Approximation

`CEPA`(*ncepa*);

($0 \leq ncepa \leq 3$). Instead of diagonalizing the hamiltonian, perform CEPA calculation, CEPA type *ncepa*. This is currently available only for single configuration reference functions.

20.3.2 Coupled Pair Functional (ACPF, AQCC)

`ACPF`,*options*
`AQCC`,*options*

where options can be

`GACPFI=`*gacpfi*

`GACPFE=`*gacpfe*

Instead of diagonalizing the hamiltonian, perform ACPF calculation or AQCC calculation. Using the options `GACPFI` and `GAPCPE` The internal and external normalization factors *gacpfi*, *gacpfe* may be reset from their default values of 1, 2/*nelec* and 1, 1-(*nelec*-2)(*nelec*-3)/*nelec*(*nelec*-1), respectively.

The ACPF and related methods are currently not robustly working for excited states. Even though it sometimes works, we do not currently recommend and support these methods for excited state calculations.

20.3.3 Projected excited state calculations

PROJECT,*record,nprojc*;

Initiate or continue a projected excited state calculation, with information stored on *record*. If *nprojc*>0, the internal CI vectors of *nprojc* previous calculations are used to make a projection operator. If *nprojc*$=-1$, this calculation is forced to be the first, i.e. ground state, with no projection. If *nprojc*$=0$, then if *record* does not exist, the effect is the same as *nprojc*$=-1$; otherwise *nprojc* is recovered from the dump in *record*. Thus for the start up calculation, it is best to use project,*record*,-1; for the following excited calculations, use project,*record*; At the end of the calculation, the wavefunction is saved, and the information in the dump *record* updated. The project card also sets the tranh option, so by default, transition hamiltonian matrices are calculated.

For example, to do successive calculations for three states, use

```
ci;...;project,3000.3,-1;
ci;...;project,3000.3;
ci;...;project,3000.3;
```

20.3.4 Transition matrix element options

TRANH,*option*;

If *option*>-1, this forces calculation of transition hamiltonian matrix elements in a TRANS or PROJECT calculation. If *option*<1, this forces calculation of one electron transition properties.

20.3.5 Convergence thresholds

ACCU,*istate,energy,coeff*;

Convergence thresholds for state *istate*. The actual thresholds for the energy and the CI coefficients are 10**(-*energy*) and 10**-(*coeff*). If this card is not present, the thresholds for all states are the default values or those specified on the THRESH card.

20.3.6 Level shifts

SHIFT,*shiftp,shifts,shifti*;

Denominator shifts for pairs, singles, and internals, respectively.

20.3.7 Maximum number of iterations

MAXITER,*maxit,maxiti*;

maxit:	maximum number of macroiterations;
maxiti:	maximum number of microiterations (internal CI).

20.3.8 Restricting numbers of expansion vectors

MAXDAV,*maxdav,maxvi*;

maxdav:	maximum number of external expansion vectors in macroiterations;
maxvi:	maximum number of internal expansion vectors in internal CI.

20.3.9 Selecting the primary configuration set

PSPACE,*select,npspac*;

select:	energy criterion for selecting p-space configurations. If negative, a test for p-space H is performed.
npspac:	minimum number of p-space configurations. Further configurations are added if either required by select or if configurations are found which are degenerate to the last p-space configuration. A minimum number of npspace is automatically determined from the state specifications.

20.3.10 Canonicalizing external orbitals

FOCK,$n_1, n_2, \ldots$;

External orbitals are obtained as eigenfunctions of a Fock operator with the specified occupation numbers n_i. Occupation numbers must be provided for all valence orbitals.

20.3.11 Saving the wavefunction

SAVE,*savecp,saveco,idelcg*;

or

SAVE [,CIVEC=*savecp*] [,CONFIG=*saveco*] [,DENSITY=*dumprec*] [,NATORB=*dumprec*] [,FILES]

savecp:	record name for save of wavefunction. If negative the wavefunction is saved after each iteration, else at the end of the job. In case of coupled cluster methods (CCSD, QCISD, BCCD), the wavefunction is saved in each iteration in any case (presently only implemented for the closed-shell case).
saveco:	record name for save of internal configurations and their maximum weight over all states for subsequent use as reference input (see SELECT card). If the record already exists, the record name is incremented by one until a new record is created.
idelcg:	if nonzero or FILES is specified, don't erase icfil and igfil (holding CI and residual vectors) at the end of the calculation.
dumprec:	Dump record for saving density matrix and natural orbitals. Only one dump record must be given. In any case the density matrix and the natural orbitals are saved. See also DM or NATORB cards.

20.3.12 Starting wavefunction

START,*readc1,irest*;

readc1: record name from which the wavefunction is restored for a restart. In the case of coupled cluster methods (CCSD, QCISD, BCCD), the amplitudes are read from record *readc1* and used for restart (presently only implemented for closed-shell methods)

irest: If nonzero, the CI coefficients are read and used for the restart; otherwise, only the wavefunction definition is read in.

20.3.13 One electron properties

EXPEC,$oper_1$,$oper_2$,$oper_3$,... ;

After the wavefunction determination, calculate expectation values for one-electron operators $oper_i$. See section 6.13 for the available operators and their keywords. In multi-state calculations or in projected calculations, also the transition matrix elements are calculated.

20.3.14 Transition moment calculations

TRANS,*readc1,readc2*,[BIORTH],[$oper_1$,$oper_2$,$oper_3$,...];

Instead of performing an energy calculation, only calculate transition matrix elements between wavefunctions saved on records *readc1* and *readc2*. See section 6.13 for a list of available operators and their corresponding keywords. If no operator names are specified, the dipole transition moments are calculated.

If option BIORTH is given, the two wavefunctions may use different orbitals. However, the number of active and inactive orbitals must be the same in each case. Note that BIORTH is not working for spin-orbit matrix elements. Under certain conditions it may happen that biorthogonalization is not possible, and then an error message will be printed.

20.3.15 Saving the density matrix

DM,*record.ifil*,[*idip*];

The first order density matrices for all computed states are stored in record *record* on file *ifil*. If *idip* is not zero, the dipole moments are printed starting at iteration *idip*. See also NATORB. In case of transition moment calculation, the transition densities are also stored, provided both states involved have the same symmetry.

20.3.16 Natural orbitals

NATORB,[RECORD=]*record.ifil*,[PRINT=*nprint*],[CORE[=*natcor*]];

Calculate natural orbitals. The number of printed external orbitals in any given symmetry is *nprint*) (default 2). *nprint=-1* suppressed the printing. If *record* is nonzero, the natural orbitals and density matrices for all states are saved in a dump record *record* on file *ifil*. If *record.ifil* is specified on a DM card (see above), this record is used. If different records are specified on the

DM and NATORB cards, an error will result. The record can also be given on the SAVE card. If CORE is specified, core orbitals are not printed.

Note: The dump record must not be the same as *savecp* or *saveco* on the SAVE card, or the record given on the PROJECT.

20.3.17 Miscellaneous options

OPTION,*code1=value,code2=value,...*

Can be used to specify program parameters and options. If no codes and values are specified, active values are displayed. The equal signs may be omitted. The following codes are allowed (max 7 per card):

NSTATE:	see state card
NSTATI:	number of states calculated in internal CI
NSTATR:	see refstat card
NCEPA:	see CEPA card
NOKOP:	if nonzero, skip integral transformation
ITRDM:	if .ge. 0 transition moments are calculated
ITRANS:	if nonzero, perform full integral transformation (not yet implemented)
IDIP:	Print dipole moments from iteration number *value*
REFOPT:	if nonzero, optimize reference coefficients; otherwise extract reference coefficients from internal CI
IAVDEN:	average HII and HSS denominators over spin couplings if nonzero
IDELCG:	if.ne.0 then destroy files icfil,igfil at end
IREST:	if nonzero, restart
NATORB:	if nonzero, natural orbitals are calculated and printed. The number of printed external orbitals per symmetry is min(*natorb*,2)
WFNAT:	if nonzero, natural orbitals are saved to this record
IPUNRF:	if nonzero, punch coefficients of reference configurations
NPUPD:	if nonzero, update pairs in nonorthogonal basis, otherwise in orthogonal basis.
MAXIT:	see maxiter card
MAXITI:	see maxiter card
MAXDAV:	see maxdav card
MAXVI:	see maxdav card
NOSING:	see nosing card
NOPAIR:	see nopair card
MXSHRF:	see select card
IKCPS=0:	In CIKEXT, only K(CP) is calculated; this option taken when and only when no singles.
IKCPS=1:	only K(CP') is calculated. Implies that modified coupling coefficients are used.

`IKCPS=2:`	K(CP) and K(CP') are calculated. Default is IKCPS=2 except when single reference configuration, when IKCPS=1.
`IOPTGM:`	Option for density matrix routines.
`IOPTGM=0:`	all quantities in density matrix routines are recalculated for each intermediate symmetry (max. CPU, min. core).
`IOPTGM=1:`	quantities precalculated and stored on disk (max. I/O, min. core).
`IOPTGM=2:`	quantities precalculated and kept in core (min. CPU, max. core).
`IOPTOR:`	If nonzero, calculate intermediate orbitals for each pair. Might improve convergence in some cases, in particular if localized orbitals are used.

20.3.18 Miscellaneous parameters

`PARAM`,*code1=value,code2=value*...

Redefine system parameters. If no codes are specified, the default values are displayed. The following codes are allowed:

`LSEG:`	disc sector length
`INTREL:`	number of integers per REAL*8 word
`IVECT=0:`	scalar machine
`IVECT=1:`	vector machine
`MINVEC:`	call MXMB in coupling coefficient routines if vector length larger than this value.
`IBANK:`	number of memory banks for vector machines. If IBANK>1, vector strides which are multiples of IBANK are avoided where appropriate.
`LTRACK:`	number of REAL*8 words per track or block (for file allocation)
`LTR:`	determines how matrices are stored on disc. If LTR=LSEG, all matrices start at sector boundaries (which optimizes I/O), but unused space is between matrices (both on disc and in core). With LTR=1 all matrices are stored dense. This might increase I/O if much paging is necessary, but reduce I/O if everything fits in core.
`NCPUS:`	Maximum number of CPUs to be used in multitasking.

20.4 Miscellaneous thresholds

`THRESH`,*code1=value,code2=value*...

If *value*=0, the corresponding threshold is set to zero, otherwise 10**(-*value*). The equal signs may be omitted. If no codes are specified, the default values are printed. The following codes are allowed (max 7 per card):

`ZERO:`	numerical zero
`THRDLP:`	delete pairs if eigenvalue of overlap matrix is smaller than this threshold.
`PNORM:`	delete pair if its norm is smaller than this threshold (all pairs are normalized to one for a closed shell case).
`PRINTCI:`	print CI coefficients which are larger than this value.
`INTEG:`	omit two-electron integrals which are smaller than this value.
`ENERGY:`	convergence threshold for energy; see also: `ACCU` card.
`COEFF:`	convergence threshold for coefficients; see also: `ACCU` card.
`SPARSE:`	omit coefficient changes which are smaller than this value.
`EQUAL:`	set values in the internal vector and the diagonal elements equal if they differ by less than this value. Useful for keeping track of symmetry.

20.5 Print options

`PRINT`,*code1=value,code2=value,...*

Print options. Generally, the value determines how much intermediate information is printed. *value*=-1 means no print (default for all codes). In some of the cases listed below the specification of higher values will generate even more output than described. The equal signs and zeros may be omitted. All codes may be truncated to three characters. The following codes are allowed (max 7 per card):

`ORBITALS:`	print orbitals
`JOP=0:`	print operator list
`JOP=1:`	print coulomb operators in MO basis
`JOP=2:`	print coulomb operators in AO and MO basis
`KOP:`	as `JOP` for internal exchange operators
`KCP=0:`	print paging information for CIKEXT
`KCP=1:`	print external exchange operators in MO basis
`KCP=2:`	print operators in AO and MO basis
`DM=0:`	print paging information for CIDIMA
`DM=1:`	print density matrix in MO basis
`DM=2:`	print density matrix in AO and MO basis
`FPP=0:`	print energy denominators for pairs
`FPP=1:`	in addition, print diagonal coupling coefficients in orthogonal basis.
`FPP=2:`	print operators FPP
`CP=0:`	print update information for pairs in each iteration
`CP=1:`	print pair matrix updates (MO basis)
`CP=2:`	in addition print pair matrices (MO basis)
`CP=3:`	print CP in AO basis (in CIKEXT)

`CI=0:`	print convergence information for internal CI
`CI=1:`	print internal CI coefficients and external expansion coefficients
`CS:`	as CP for singles
`CPS=0:`	print paging information for CICPS
`CPS=1:`	print matrices CPS in MO basis
`GPP=0:`	print paging information for CIGPQ
`GPP=1:`	print matrices GP at exit of CIGPQ
`GPS=0:`	print paging information for CIGPS
`GPS=1:`	print vectors GS at exit CIGPS
`GSP=1:`	print matrices GP at exit CIGPS
`GPI=0:`	print paging information for CIGPI
`GPI=1:`	print total GP in orthogonal basis
`GPI=2:`	print matrices GP and TP
`GIP=0:`	print paging information for CIGIP
`GIP=1:`	print GI at exit CIGIP
`GSS=0:`	print paging information for CIGSS
`GSS=1:`	print vectors GS at exit CIGSS
`GSI=0:`	print paging information for CIGSI
`GSI=1:`	print GS at exit CIGSI
`GIS=0:`	print paging information for CIGIS
`GIS=1:`	print GI at exit CIGIS
`GII:`	print intermediate information in internal CI
`DPQ:`	print coupling coefficients $\alpha(P,Q)$
`EPQ:`	print coupling coefficients $\beta(P,Q)$
`HPQ:`	print coupling coefficients $\gamma(P,Q)$
`DPI:`	print coupling coefficients for pair-internal interactions
`DSS:`	not yet used
`DSI:`	not yet used
`LOG:`	At end of each iteration, write summary to log file. Delete at end of job if `LOG=0`
`CC=0:`	print address lists for coupling coefficients
`CC=1:`	print coupling coefficients
`DEN=1:`	print internal first order density
`DEN=2:`	print internal second order density
`DEN=3:`	print internal third order density
`DEN=4:`	print first, second and third order densities
`GAM=1:`	print first order transition densities
`GAM=2:`	print second order transition densities
`GAM=3:`	print first and second order transition densities
`PAIRS=0:`	print list of non redundant pairs

PAIRS=1:	print list of all pairs
CORE=0:	print summary of internal configurations ($N, N-1$ and $N-2$ electron)
CORE=1:	print internal configurations ($N, N-1, N-2$)
REF=0:	print summary of reference configurations
REF=1:	print reference configurations and their coefficients
PSPACE:	print p-space configurations
HII:	print diagonal elements for internals
HSS:	print diagonal elements for singles
SPQ:	various levels of intermediate information in pair orthogonalization routine.
TEST=0:	print information at each subroutine call
TEST=1:	print in addition information about I/O in LESW, SREIBW
TEST=2:	print also information about I/O in FREAD, FWRITE
CPU:	print analysis of CPU and I/O times
ALL:	print everything at given level (be careful!)

20.6 Examples

```
***,Single reference CISD and CEPA-1 for water
r=0.957,angstrom
theta=104.6,degree;
geometry={O;              !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
{hf;wf,10,1;}                           !TOTAL SCF ENERGY     -76.02680642
{ci;occ,3,1,1;core,1;wf,10,1;}          !TOTAL CI(SD) ENERGY  -76.22994348
{cepa(1);occ,3,1,1;core,1;wf,10,1;}     !TOTAL CEPA(1) ENERGY -76.23799334
```

http://www.molpro.net/info/current/examples/h2o_cepa1.com

```
***,Valence multireference CI for X and A states of H2O
gthresh,energy=1.d-8
r=0.957,angstrom,theta=104.6,degree;
geometry={O;              !z-matrix geometry input
          H1,O,r;
          H2,O,r,H1,theta}
{hf;wf,10,1;}          !TOTAL SCF ENERGY               -76.02680642
{multi;occ,4,1,2;closed,2;frozen,1;wf,9,2,1;wf,9,1,1;tran,ly}
                     !MCSCF ENERGY                    -75.66755631
                     !MCSCF ENERGY                    -75.56605896
{ci;occ,4,1,2;closed,2;core,1;wf,9,2,1;save,7300.1}
                     !TOTAL MRCI ENERGY               -75.79831209
{ci;occ,4,1,2;closed,2;core,1;wf,9,1,1;save,7100.1}
                     !TOTAL MRCI ENERGY               -75.71309879
{ci;trans,7300.1,7100.1,ly}
                     !Transition moment <1.3|X|1.1> = -0.14659810  a.u.
                     !Transition moment <1.3|LY|1.1> = 0.96200488i a.u.
```

http://www.molpro.net/info/current/examples/h2op_mrci_trans.com

```
***,BH singlet Sigma and Delta states
r=2.1
geometry={b;h,b,r}
{hf;occ,3;wf,6,1;}
{multi;
occ,3,1,1;frozen,1;wf,6,1;state,3;lquant,0,2,0;wf,6,4;lquant,2;
tran,lz;
expec2,lzlz;}
! Sigma states:- energies -25.20509620 -24.94085861
{ci;occ,3,1,1;core,1;wf,6,1;state,2,1,3;}
! Delta states:- energies -24.98625171
{ci;occ,3,1,1;core,1;wf,6,1;state,1,2;}
! Delta state:- xy component
{ci;occ,3,1,1;core,1;wf,6,4;}
```

http://www.molpro.net/info/current/examples/bh_mrci_sigma_delta.com

20.7 Cluster corrections for multi-state MRCI

In the following, we assume that

$$\Psi^{(n)}_{\text{ref}} = \sum_R C^{(0)}_{Rn}\Phi_R \tag{46}$$

$$\Psi^{(n)}_{\text{mrci}} = \sum_R C_{Rn}\Phi_R + \Psi_c \tag{47}$$

are the normalized reference and MRCI wave functions for state n, respectively. $C^{(0)}_R$ are the coefficients of the reference configurations in the initial reference functions and C_{Rn} are the relaxed coefficients of these configurations in the final MRCI wave function. Ψ_c is the remainder of the MRCI wave function, which is orthogonal to all reference configurations Φ_R.

The corresponding energies are defined as

$$E^{(n)}_{\text{ref}} = \langle\Psi^{(n)}_{\text{ref}}|\hat{H}|\Psi^{(n)}_{\text{ref}}\rangle, \tag{48}$$

$$E^{(n)}_{\text{mrci}} = \langle\Psi^{(n)}_{\text{mrci}}|\hat{H}|\Psi^{(n)}_{\text{mrci}}\rangle. \tag{49}$$

The standard Davidson corrected correlation energies are defined as

$$E^n_{\text{D}} = E^{(n)}_{\text{corr}} \cdot \frac{1-c_n^2}{c_n^2} \tag{50}$$

where c_n is the coefficient of the (fixed) reference function in the MRCI wave function:

$$c_n = \langle\Psi^{(n)}_{\text{ref}}|\Psi^{(n)}_{\text{mrci}}\rangle = \sum_R C^{(0)}_{Rn}C_{Rn}, \tag{51}$$

and the correlation energies are

$$E^{(n)}_{\text{corr}} = E^{(n)}_{\text{mrci}} - E^{(n)}_{\text{ref}}. \tag{52}$$

In the vicinity of avoided crossings this correction may give unreasonable results since the reference function may get a small overlap with the MRCI wave function. One way to avoid this problem is to replace the reference wave function $\Psi_{\rm ref}^{(n)}$ by the the relaxed reference functions

$$\Psi_{\rm rlx}^{(n)} = \frac{\sum_R C_{Rn}\Phi_R}{\sqrt{\sum_R C_{Rn}^2}}, \tag{53}$$

which simply leads to

$$c_n^2 = \sum_R C_{Rn}^2. \tag{54}$$

Alternatively, one can linearly combine the fixed reference functions to maximize the overlap with the MRCI wave functions. This yields projected functions

$$\Psi_{\rm prj}^{(n)} = \sum_m |\Psi_{\rm ref}^{(m)}\rangle\langle\Psi_{\rm ref}^{(m)}|\Psi_{\rm mrci}^{(n)}\rangle = \sum_m |\Psi_{\rm ref}^{(m)}\rangle d_{mn} \tag{55}$$

with

$$d_{mn} = \langle\Psi_{\rm ref}^{(m)}|\Psi_{\rm mrci}^{(n)}\rangle = \sum_R C_{Rm}^{(0)} C_{Rn}. \tag{56}$$

These projected functions are not orthonormal. The overlap is

$$\langle\Psi_{\rm prj}^{(m)}|\Psi_{\rm prj}^{(n)}\rangle = (\mathbf{d}^\dagger\mathbf{d})_{mn}. \tag{57}$$

Symmetrical orthonormalization, which changes the functions as little as possible, yields

$$\Psi_{\rm rot}^{(n)} = \sum_m |\Psi_{\rm ref}^{(m)}\rangle u_{mn}, \tag{58}$$

$$\mathbf{u} = \mathbf{d}(\mathbf{d}^\dagger\mathbf{d})^{-1/2}. \tag{59}$$

The overlap of these functions with the MRCI wave functions is

$$\langle\Psi_{\rm rot}^{(m)}|\Psi_{\rm mrci}^{(n)}\rangle = [(\mathbf{d}^\dagger\mathbf{d})(\mathbf{d}^\dagger\mathbf{d})^{-1/2}]_{mn} = [(\mathbf{d}^\dagger\mathbf{d})^{1/2}]_{mn}. \tag{60}$$

Thus, in this case we use for the Davidson correction

$$c_n = [(\mathbf{d}^\dagger\mathbf{d})^{1/2}]_{nn}. \tag{61}$$

The final question is which reference energy to use to compute the correlation energy used in eq. (50). In older MOLPRO version (to 2009.1) the reference wave function which has the largest overlap with the MRCI wave function was used to compute the reference energy for the corresponding state. But this can lead to steps of the Davidson corrected energies if the order of the states swaps along potential energy functions. In this version there are two options: the default is to use for state n the reference energy n, cf. eq. (52) (assuming the states are ordered according to increasing energy). The second option is to recompute the correlation energies using the rotated reference functions

$$E_{corr}^{(n)} = E_{\rm MRCI}^{(n)} - \langle\Psi_{\rm rot}^{(n)}|\hat{H}|\Psi_{\rm rot}^{(n)}\rangle \tag{62}$$

Both should give smooth potentials (unless at conical intersections or crossings of states with different symmetries), but there is no guarantee that the Davidson corrected energies of different states don't cross. This problem is unavoidable for non-variational energies. The relaxed and rotated Davidson corrections give rather similar results; the rotated one yields somewhat larger cluster corrections and was found to give better results in the case of the F + H_2 potential [see J. Chem. Phys. **128**, 034305 (2008)].

By default, the different cluster corrections listed in Table 10 are computed in multi-state MRCI calculations. and stored in variables. By default, `ENERGD(n)=ENERGD0(n)`. This can be

Table 10: Cluster corrections computed in multi-state MRCI calculations. By default, the energies are in increasing order of the MRCI total energy. In single-state calculations only the fixed and relaxed values are available.

Name	c_n (Eq.)	$E^{(n)}_{corr}$ (Eq.)	Variable
Using standard reference energies:			
Fixed	(51)	(52)	`ENERGD1(n)`
Relaxed	(54)	(52)	`ENERGD0(n)`
Rotated	(61)	(52)	`ENERGD2(n)`
Using rotated reference energies:			
Relaxed	(54)	(62)	`ENERGD3(n)`
Rotated	(61)	(62)	`ENERGD4(n)`

changed by setting `OPTION,CLUSTER=`*x*; then `ENERGD(n)=ENERGD`*x*`(n)` (default $x = 0$). The behaviour of Molpro 2009.1 and older can be retrieved using

`MRCI,SWAP,ROTREF=-1.`

21 MULTIREFERENCE RAYLEIGH SCHRÖDINGER PERTURBATION THEORY

Bibliography:

Original RS2/RS3:

H.-J. Werner, Mol. Phys. 89, 645-661 (1996)

New internally contracted RS2C:

P. Celani and H.-J. Werner, J. Chem. Phys. 112, 5546 (2000)

All publications resulting from use of this program must acknowledge the above.

The commands

`RS2`,*options*
`RS2C`,*options*
`RS3`,*options*

are used to perform second or third-order perturbation calculations. `RS3` always includes `RS2` as a first step. For closed-shell single-reference cases, this is equivalent to `MP2` or `MP3` (but a different program is used). `RS2C` calls a new more efficient second-order program (see below), which should normally be used if third-order is not required (note that `RS3C` is not available).

Options can be the following:

`Gn`	Use modified zeroth order Hamiltonian, see section 21.4
`SHIFT=`*value*	Level shift, see section 21.5
`IPEA=`*value*	IPEA shift proposed by G. Ghigo, B. O. Roos, and P.A. Malmqvist, Chem. Phys. Lett. **396**, 142 (2004), see section 21.5.
`MIX=`*nstates*	Invokes multi-state (MS-CASPT2) treatment using *nstates* states. See section 21.3 for more details.
`ROOT=`*ioptroot*	Root number to be optimized in geometry optimization. This refers to the *nstates* included in the MS-CASPT2. See section 21.7 for more details.
`SAVEH=`*record*	Record for saving the effective Hamiltonian in MS-CASPT2 calculations. If this is not given, a default record will be used (recommended).
`INIT`	(logical) Initializes a MS-CASPT2 with single state reference functions, see section 21.3
`IGNORE`	(logical) Flags an approximate gradient calculation without CP-CASPT2; see section 21.7 for details.

In addition, all valid options for `MRCI` can be given (see Sect. 20).

21.1 Introduction

Multireference perturbation calculations are performed by the MRCI program as a special case. For RS2 (CASPT2,RASPT2) only matrix elements over a one-electron operator need to be computed, and therefore the computational effort is much smaller than for a corresponding MRCI. For RS3 (CASPT3) the energy expectation value for the first-order wavefunction must be computed and the computational effort is about the same as for one MRCI iteration. The RS2 and RS3 programs use the same configuration spaces as the MRCI, i.e., only the doubly external configurations are internally contracted.

A new version of the program has been implemented in which also subspaces of the singly external and internal configuration spaces are internally contracted (see reference given above). This program, which is called using the keyword `RS2C`, is more efficient than `RS2`, in particular for large molecules with many closed-shell (inactive) orbitals. It is recommended to use this program for normal applications of second-order multireference perturbation theory (CASPT2, RASPT2). Note that it gives slightly different results than `RS2` due to the different contraction scheme. It should also be noted that neither `RS2` or `RS2C` are identical with the CASPT2 of Roos et al. [J. Chem. Phys. **96**, 1218 (1992)], since certain configuration subspaces are left uncontracted. However, the differences are normally very small. The last point that should be mentioned is that the calculation of CASPT2/RASPT2 density matrices (and therefore molecular properties) is presently possible only with the RS2 command and *not* with RS2C.

The results of multireference perturbation theory may be sensitive to the choice of the zeroth-order Hamiltonian. This dependence is more pronounced in second-order than in third-order. Several options are available, which will be described in the following sections. It may also happen that $(\hat{H}^{(0)} - E^{(0)})$ in the basis of the configuration state functions becomes (nearly) singular. This is known as ”intruder state problem” and can cause convergence problems or lead to a blow-up of the wavefunction. Often, such problems can be eliminated by including more orbitals into the reference wavefunction, but of course this leads to an increase of the CPU time. The use of modified Fock operators (see below) or level shifts, as proposed by Roos and

Andersson [Chem. Phys. Lett. **245**, 215 (1995)] may also be helpful. Presently, only ”real” level shifts have been implemented.

With no further input cards, the wavefunction definition (core, closed, and active orbital spaces, symmetry) corresponds to the one used in the most recently done SCF or MCSCF calculation. By default, a CASSCF reference space is generated. Other choices can be made using the `OCC`, `CORE`, `CLOSED`, `WF`, `SELECT`, `CON`, and `RESTRICT` cards, as described for the `CI` program. The orbitals are taken from the corresponding SCF or MCSCF calculation unless an `ORBITAL` directive is given.

For a CASPT2 calculation, the zeroth-order Hamiltonian can be brought to a block-diagonal form when (pseudo)canonical orbitals are used. This leads to fastest convergence. It is therefore recommended that in the preceding `MULTI` calculation the orbitals are saved using the `CANONICAL` directive (note that the default is `NATORB`).

Most options for MRCI calculations (like `STATE`, `REFSTATE` etc.) apply also for RS2(C) and RS3 and are not described here again. Some additional options which specific for CASPT2/3 and are described below.

21.2 Excited state calculations

There are two possibilities to perform excited state calculations:

1) One can calculate each state separately. This is done using the card

`STATE`,1,*root*

where *root* is the desired root (i.e., 2 for the first excited state). In this case the Fock operator used in the zeroth-order Hamiltonian is computed using the density for the given state.

2) Alternatively, two or more states can be computed simultaneously, using

`STATE`, *n* [,*root1, root2, . . . , rootn*]

where *n* is the number of states to be computed. The default is to compute the lowest *n* roots. Optionally, this default can be modified by specifying the desired roots *rooti* as shown. One should note that this *does not* correspond to the multi-state CASPT2 as described in section 21.3.

In the case that several states are computed simultaneously, the fock operator employed in the zeroth-order Hamiltonian is computed from a state-averaged density matrix, and the zeroth-order Hamiltonians for all states are constructed from the same fock operator. By default, equal weights for all states are used. This default can be modified using the `WEIGHT` directive

`WEIGHT`,*w1, w2,. . . ,wn*.

If a `REFSTATE` card is given (see section 20.2.11), the state-averaged fock operator is made for all reference states, and the `WEIGHT` card refers to the corresponding states.

21.3 Multi-State CASPT2

Multi-state CASPT2 is implemented as described by Finley et al. CPL **288**, 299 (1998). Currently this can only be used with the `RS2` program (i.e., not with `RS2C`). There are two different modes in which MS-CASPT2 calculations can be performed:

(i) Each of the states to be mixed is computed independently, and finally all states are mixed. In the following, such calculations will be denoted SS-SR-CASPT2 (single-state, single reference

CASPT2). There is one contracted reference state for each CASPT2 calculation that is specific for the state under consideration. This is the cheapest method, but there are no gradients available in this case. It is the users responsibility to make sure that no state is computed twice.

(ii) All *nstates* states are treated together, with *nstates* contracted reference states. This is more expensive, but should give a more balanced description since the different reference states can mix in the CASPT2. It is required that *nstates* equals the number of states specified on the `STATE` directive. For this case, denoted ”MS-MR-CASPT2” (multi-state multi reference CASPT2), analytical energy gradients are available, see section 21.7

21.3.1 Performing SS-SR-CASPT2 calculations

If one wants to mix together *nstates* CASPT2 wavefunctions, a *nstates* single-state, single-reference CASPT2 calculations must be run.

The first calculation must use

{`RS2,MIX=`*nstates*, `INIT`, *options*
`STATE,1,1;`}

and the subsequent ones

{`RS2,MIX=`*nstates*, *options*
`STATE,1,`*istate*;}

for $istate = 2,\ldots,nstates$. Further *options* can be given, for instance a level shift.

At the end of each calculation, the CASPT2 wavefunction is stored, and at the end of the last CASPT2 calculation the Bloch Hamiltonian and the corresponding overlap matrix are automatically assembled and printed. The Hamiltonian is diagonalized after symmetrization following Brandow IJQC 15, 207 (1979), as well as with simple half-sum (averaging). The MS-CASPT2 energy and mixing coefficients printed in each case.

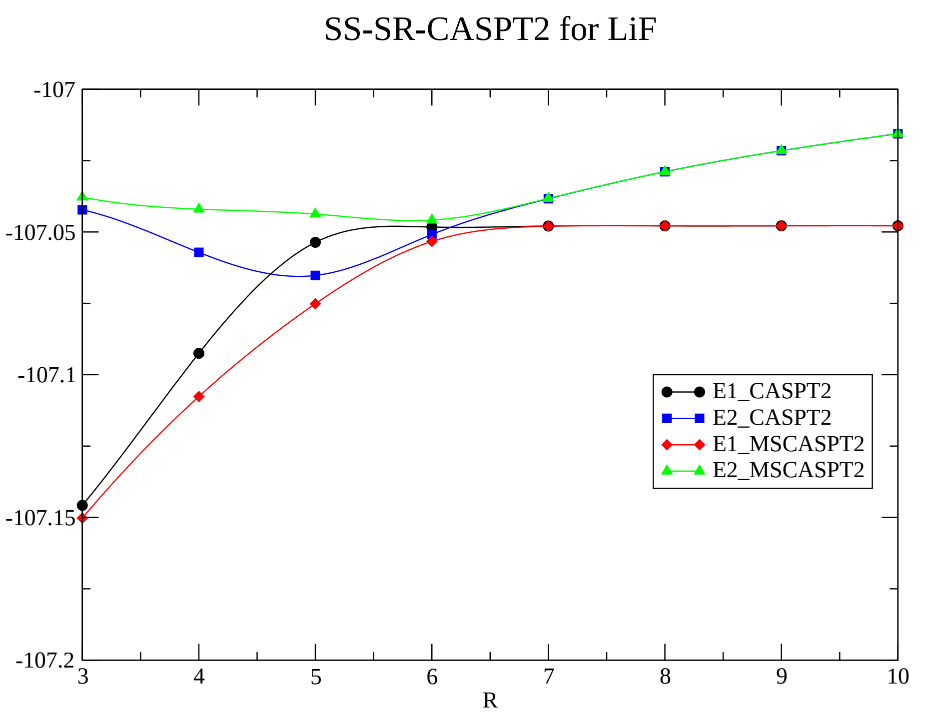
The variable MSENERGY(i) (with i=1,...nstates) is set to the multi-state energies obtained with half-sum diagonalization. If a Level Shift is present, MSENERGY(i) contains the multi-state energies obtained with half-sum diagonalization of the Bloch Hamiltonian whose diagonal elements (CASPT2 energies) have been corrected with the level shift.

Example: SS-SR-CASPT2 calculation for LiF

```
r=[3,4,5,6,7,8,9,10] ang

i=1
geometry={Li
          F,1,r(i)}

basis=vtz,F=avtz

hf                       !Hartree-Fock

do i=1,#r                !loop over range of bond distances
{multi
closed,3,0,0,0
occ,   5,2,2,0
state,2                  !SA-CASSCF for 2 states
canonical,ci}

{rs2,MIX=2,INIT
state,1,1}               !single state CASPT2 for reference state 1

e1_caspt2(i)=energy      !unmixed caspt2 energy for ground state

{rs2,MIX=2
state,1,2}               !single state CASPT2 for reference state 2

e2_caspt2(i)=energy      !unmixed caspt2 energy for excited state
e1_mscaspt2(i)=msenergy(1)   !ms-caspt2 energy for ground state
e2_mscaspt2(i)=msenergy(2)   !ms-caspt2 energy for excited state
enddo

{table,r,e1_caspt2,e2_caspt2,e1_mscaspt2,e2_mscaspt2
title,SS-SR-CASPT2 for LiF
plot,file='lif_sr_mscaspt2.plot'
}
```

http://www.molpro.net/info/current/examples/lif_sr_mscaspt2.com

This produces the plot



21.3.2 Performing MS-MR-CASPT2 calculations

In the case of multi-state multi-reference CASPT2 calculations, only a single run is needed:

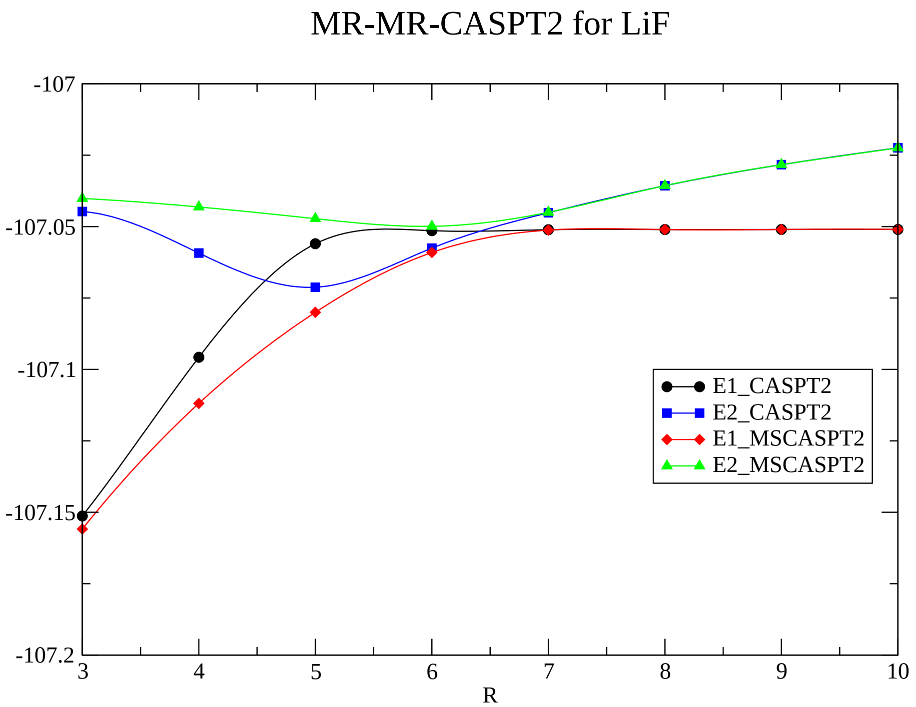
{`RS2,MIX=`*nstates*, *options*
`STATE,`*nstates*}

Example: MS-MR-CASPT2 calculation for LiF

```
r=[3,4,5,6,7,8,9,10] ang

i=1
geometry={Li
          F,1,r(i)}

basis=vtz,F=avtz

hf                       !Hartree-Fock

do i=1,#r                !loop over range of bond distances
{multi
closed,3,0,0,0
occ,   5,2,2,0
state,2                  !SA-CASSCF for 2 states
canonical,ci}

{rs2,MIX=2
state,2}                 !2-state CASPT2 with 2 reference states

e1_caspt2(i)=energy(1)       !unmixed caspt2 energy for ground state
e2_caspt2(i)=energy(2)       !unmixed caspt2 energy for ground state

e1_mscaspt2(i)=msenergy(1)  !ms-caspt2 energy for ground state
e2_mscaspt2(i)=msenergy(2)  !ms-caspt2 energy for excited state
enddo

{table,r,e1_caspt2,e2_caspt2,e1_mscaspt2,e2_mscaspt2
title,MS-MR-CASPT2 for LiF
plot,file='lif_mr_mscaspt2.plot'
}
```

http://www.molpro.net/info/current/examples/lif_mr_mscaspt2.com

This produces the plot



One can clearly see that this gives smoother potentials than the SS-SR-CASPT2 calculation in the previous section. Also, the avoided crossing is shifted to longer distances, which is due to the improvement of the electron affinity of F.

21.4 Modified Fock-operators in the zeroth-order Hamiltonian.

The g_1, g_2, and g_3 operators proposed by Andersson [Theor. Chim. Acta **91**, 31 (1995)] as well as a further g_4 operator may be used. g_4 makes CASPT2 calculations size extensive for cases in which a molecule dissociates to high-spin open-shell (RHF) atoms.

The index *n* of the operator to be used is specified on the `RS2`, `RS2C`, or `RS3` card:

`RS2`,*option*
`RS2C`,*option*
`RS3`,*option*

where *option* can be `G1`, `G2`, `G3`, or `G4`. This option can be followed or preceded by other valid options.

21.5 Level shifts

Level shifts are often useful to avoid intruder state problems in excited state calculations. MOLPRO allows the use of shifts as described by Roos and Andersson, [Chem. Phys. Lett. **245**, 215 (1995)]. The shift can be specified on the `RS2` or `RS2C` card

`RS2` [,G*n*] [,`SHIFT`=*shift*],`IPEA`=*value*
`RS2C` [,G*n*] [,`SHIFT`=*shift*],`IPEA`=*value*

Typical choices for the shift is are $0.1 - 0.3$. Only two figures after the decimal point are considered. The shift affects the results, the printed energies as well as the `ENERGY` variable include the energy correction for the shift as proposed by Roos and Andersson. At convergence, also the uncorrected energies are printed for comparison.

Alternatively (or in addition), the IPEA shift of G. Ghigo, B. O. Roos, and P.A. Malmqvist, Chem. Phys. Lett. **396**, 142 (2004) can be used. The implementation is not exactly identical to the one in MOLCAS, since in our program the singly external configurations are not (RS2) or only partially (RS2C) contracted. In Molpro, the shift is implemented as follows:

$\frac{1}{2}D_{pp}\varepsilon$ is added to the occupied part of the Fock matrix; in addition, 2ε is added as a general shift (not corrected). ε is the value specified with the `IPEA` option (default 0). A value of 0.20-0.25 is recommended. This removes intruder state problems to a large extent and usually improves the results. Note that the method is not exactly orbital invariant, and pseudo-canonical orbitals should be used (see `CANONICAL` option in MULTI).

It is possible to use `SHIFT` and `IPEA` simultaneously, but it does not make sense to use one of the `G`-options together with `IPEA`.

21.6 Integral direct calculations

`RS2`, `RS2C`, and `RS3` calculations with very large basis sets can be performed in integral-direct mode. The calculation will be direct if a global `DIRECT` or `GDIRECT` card appears earlier in the input. Alternatively, (mainly for testing) `DIRECT` can be specified as an option on the `RS`*n*`[C]` card:

`RS2` [,G*n*] [,`SHIFT=`*shift*] [,`DIRECT`]
`RS2C` [,G*n*] [,`SHIFT=`*shift*] [,`DIRECT`]

21.7 CASPT2 gradients

P. Celani and H.-J. Werner, J. Chem. Phys. **119**, 5044 (2003))

CASPT2 analytic energy gradients are computed automatically if a `FORCE` or `OPTG` command follows (see sections 41 and 42). Analytical gradients are presently only available for `RS2` calculations (not `RS2C`), and only for the standard $\hat{H}^{(0)}$ (not G1, G2 etc). Gradients can be computed for single-state calculations, as well as multi-state MS-MR-CASPT2 (see section 21.3.

In single state calculations, the gradient is automatically computed for the state computed in CASPT2/RSPT2 (i.e., using `STATE,1,2` the second state in the symmetry under consideration is computed, see section 21.2). The program works with state-averaged MCSCF (CASSCF) orbitals, and no `CPMCSCF` directive is needed. It is necessary that the state under consideration is included in the preceding (state-averaged) MCSCF/CASSCF. The RS2 gradient program can also be used to compute state-averaged MCSCF/CASSCF gradients by using the `NOEXC` directive.

In a multi-state MS-MR-CASPT2 calculation, the state for which the gradient is computed must be specified using the `ROOT` option (default `ROOT=1`), i.e.,

`RS2,MIX=`*nstates*, `ROOT=`*ioptroot*

where $1 \leq ioptroot \leq nstates$.

Level shifts can be used. By default, the exact gradient of the level-shift corrected energy is computed. For a non-zero shift, this requires to solve the CASPT2 Z-vector equations, which roughly doubles the computational effort. In single state calculations it is possible to ignore the effect of the level shift on the gradient and not to solve the Z-vector equation. This variant, which is described in the above paper, may be sufficiently accurate for many purposes. It is invoked using the `IGNORE` option, e.g.

```
RS2,SHIFT=0.2,IGNORE
OPTG
```

Any publications employing the CASPT2 gradients should cite the above paper. A citation for MS-CASPT2 gradient method is P. Celani and H.-J. Werner, *to be published*.

Example:

CASPT2 geometry optimizations for H_2O:

```
***
memory,8,m
gthresh,energy=1.d-10
!
basis=vdz
R=2.0
R0=R
Theta=100
geometry={O
          H1,O,R;
          H2,O,R,H1,THETA}

hf;accu,12

{multi;closed,2}

rs2,shift=0.3,ignoreshift      !ignore shift in computing gradient, i.e., no cp-caspt2
optg,gradient=1.d-5
e_opt(1)=energy
r_opt(1)=r
theta_opt(1)=theta
method(1)='rs2,analytical,ignore'

rs2,shift=0.3                  !exact gradient with shift
optg,gradient=1.d-5
e_opt(2)=energy
r_opt(2)=r
theta_opt(2)=theta
method(2)='rs2,analytical,exact'

rs2,shift=0.3                  !numerical gradient with shift
optg,gradient=1.d-5,numerical,fourpoint !use four-point numerical gradient
e_opt(3)=energy
r_opt(3)=r
theta_opt(3)=theta
method(3)='rs2,numerical'

rs2c,shift=0.3                 !numerical gradient of rs2c with shift
optg,gradient=1.d-5,fourpoint  !use four-point numerical gradient
e_opt(4)=energy
r_opt(4)=r
theta_opt(4)=theta
method(4)='rs2c,numerical'

table,method,r_opt,theta_opt,e_opt
digits,,4,4,8
```

http://www.molpro.net/info/current/examples/h2o_caspt2_opt.com

This produces the Table

```
METHOD                        R_OPT   THETA_OPT        E_OPT
```

```
 rs2,analytical,ignore   1.8250   102.1069   -76.22789382
 rs2,analytical,exact    1.8261   102.1168   -76.22789441
 rs2,numerical           1.8261   102.1168   -76.22789441
 rs2c,numerical          1.8260   102.1187   -76.22787681
```

MS-CASPT2 geometry optimization for the second excited $^3B_2$ state if H_2O:

```
***
memory,8,m
gthresh,energy=1.d-12
!
basis=vdz
R=2.0
R0=R
Theta=100
step=0.001
geometry={O
          H1,O,R;
          H2,O,R,H1,THETA}

hf;accu,12

multi        !state averaged casscf for various triplet states
closed,2
wf,10,1,2
state,3
wf,10,2,2
state,2
wf,10,3,2
state,3
canonical,2140.2

rs2,mix=3,root=2,shift=0.2   !optimized second 3B2 state
wf,10,3,2                    !3B2 wavefunction symmetry
state,3                      !include 3 states
optg,gradient=1.d-5          !geometry optimization using analytical gradients

e_opt(1)=msenergy(2)         !optimized ms-caspt2 energy
r_opt(1)=r                   !optimized bond distance
theta_opt(1)=theta           !optimized bond angle
method(1)='rs2,analytical'

rs2,mix=3,shift=0.2
wf,10,3,2                    !3B2 wavefunction symmetry
state,3                      !include 3 states
optg,variable=msenergy(2),gradient=1.d-5,fourpoint
                             !geometry optimization using numerical gradients

e_opt(2)=msenergy(2)         !optimized ms-caspt2 energy
r_opt(2)=r                   !optimized bond distance
theta_opt(2)=theta           !optimized bond angle
method(2)='rs2,numerical'

table,method,r_opt,theta_opt,e_opt
digits,,4,4,8
```

http:
//www.molpro.net/info/current/examples/h2o_mscaspt2_opt.com

This produces the table

```
 METHOD              R_OPT   THETA_OPT        E_OPT
```

```
rs2,analytical   2.4259    96.7213   -75.81630628
rs2,numerical    2.4259    96.7213   -75.81630628
```

21.8 Coupling MRCI and MRPT2: The CIPT2 method

P. Celani, H. Stoll, H.-J. Werner and P. J. Knowles, Mol. Phys. **102**, 2369 (2004).

For particularly difficult cases with strong intruder problems, or in which second-order perturbation theory fails to predict reliable results, a new method that couples MRCI and CASPT2 has been developed. This variant is invoked using the `CIPT2` directive:

`CIPT2`

In this case all excitations solely from active orbitals are treated by MRCI, while the remaining excitations involving inactive (closed-shell) orbitals are treated by second-order perturbation theory. Both methods are coupled by minimizing an appropriate energy functional. Of course, this method is much more expensive that MRPT2. The cost is comparable to the cost for an MRCI without correlating the inactive orbitals.

21.9 Further options for CASPT2 and CASPT3

Other options can be set using the `OPTION` command. These options are mainly used for testing purposes and should be used with care. It should be noted that the only option that can be modified in the RS2C program is `IFDIA`: all others only work with RS2/RS3.

`OPTION`,*code1=value,code2=value,...*

Of relevance for the CASPT2/3 program are the following options:

`IPROCS=0`	(Default). Calculation uses uncontracted singles with `RS2`.
`IPROCS=1`	Non-interacting singles are projected out during update. This is an approximate procedure which should be used with care.
`IPROCS=2`	The singles are fully internally contracted in `RS2`. This is achieved via a projection operator during the coefficient update and may be inefficient. G
`IPROCS=3`	Only singles with one or two holes in the closed-shells are internally contracted in `RS2` using a projection operator.
`IPROCI=0`	(Default). Calculation uses uncontracted internals with `RS2`.
`IPROCI=1`	Internals with two holes in the inactive space are internally contracted in `RS2` using a projection operator.
`IPROCS=3,IPROCI=1`	This combination of options reproduces with `RS2` the `RS2C` result using projection operators. This requires lot of memory and disk space and it is feasible only for small molecules.
`IFDIA=0`	(Default). All off-diagonal elements of the effective Fock matrix are included.
`IFDIA=1`	The internal-external block of the Fock-matrix is neglected. This eliminates the single-pair coupling.
`IFDIA=2`	All off-diagonal elements of the Fock matrix are neglected. This corresponds to CASPT2D of Andersson et al. Note: in this case the result is not invariant to rotations among active orbitals!

`IHINT=0`	(Default). Only one-electron integrals are used in the zeroth-order Hamiltonian for all interactions.
`IHINT=1`	The all-internal two-electron integrals are used in the zeroth-order Hamiltonian for the internal-internal and single-single interactions.
`IHINT=2`	The all-internal two-electron integrals in the zeroth-order Hamiltonian are used for the internal-internal, single-single, and pair-pair interactions. Using `IHINT=2` and `IDFIA=1` corresponds to Dyall's CAS/A method for the case that CASSCF references with no closed-shells (inactive orbitals) are used. Note that this requires more CPU time than a standard CASPT2 calculation. Moreover, convergence of the CAS/A method is often slow (denominator shifts specified on a `SHIFT` card may be helpful in such cases). In general, we do not recommend the use of `IHINT` with nonzero values.
`NOREF=1`	(Default). Interactions between reference configurations and singles are omitted.
`NOREF=0`	Interactions between reference configurations and singles are included. This causes a relaxation of the reference coefficients but may lead to intruder-state problems.
`IMP3=2`	After CASPT2 do variational CI using all internal configurations and the first-order wavefunctions of all states as a basis. In this case the second-order energy will correspond to the variational energy, and the third-order energy approximately to a Davidson-corrected energy. This is useful in excited state calculations with near-degeneracy situations.

22 NEVPT2 calculations

Reference literature:

C. Angeli, R. Cimiraglia, S. Evangelisti, T. Leininger and J. P. Malrieu, *J. Chem. Phys.*, **114**, 10252, (2001)

C. Angeli, R. Cimiraglia and J. P. Malrieu, *J. Chem. Phys.*, **117**, 9138, (2002)

C. Angeli, M. Pastore and R. Cimiraglia, *Theor. Chem. Acc.*, **117**, 743 (2007)

All publications resulting from use of this program must acknowledge the above.

22.1 General considerations

NEVPT2 is a form of second-order multireference perturbation theory which can be applied to CAS–SCF wavefunctions or, more generally, to CAS–CI wavefunctions. The term NEVPT is an acronym for *"n–electron valence state perturbation theory"*. While we refer the reader to the pertinent literature (see above), we limit ourselves to recalling here that the most relevant feature of NEVPT2 consists in that the first order correction to the wave function is expanded over a set of properly chosen *multireference* functions which correctly take into consideration the two–electron interactions occurring among the active electrons. Among the properties ensured by NEVPT2 we quote:

- Strict separability (size consistence): the energy of a collection of non–interacting systems equals the sum of the energies of the isolated systems
- Absence of intruder states: the zero-order energies associated to the functions of the outer space are well separated from the zero-order energy of the state being studied, thus avoiding divergences in the perturbation summation
- The first order correction to the wavefunction is an eigenfunction of the spin operators S^2 and S_z
- Electronically excited states are dealt with at the same level of accuracy as the ground state
- NEVPT2 energies are invariant under a unitary transformation of the active orbitals. Furthermore, the choice of canonical orbitals for the core and virtual orbitals (the default choice) ensure that the results coincide with those of an enlarged version of the theory fully invariant under rotations in the core and virtual orbital spaces, respectively
- NEVPT2 coincides with MP2 in the case of a HF wave function

NEVPT2 has been implemented in two variants both of which are present in MOLPRO, these are the *strongly contracted* (SC) and the *partially contracted* (PC) variants. The two variants differ by the number of perturber functions employed in the perturbation summation. The PC–NEVPT2 uses a richer function space and is in general more accurate than the SC–NEVPT2. The results of SC–NEVPT2 and PC–NEVPT2 are anyway usually very close to one another.

22.2 Input description

NEVPT2 must follow a CAS–SCF or CAS–CI calculation. The command

`NEVPT2`,*options*
has to be specified to carry out a second–order perturbation calculation. NEVPT2 is part of the MRCI program and uses the options of the latter. Of particular relevance are the options `CORE`, `CLOSED`, `OCC`, `WF` and `STATE` of the MRCI program. There is, at the moment, only one option specific to NEVPT2 which can be provided by the user:

`THRNEVPT2`	The threshold to discard small coefficients in the CAS wavefunction (default = 0.0),

The present implementation of NEVPT2 is state–specific, *i.e.* the perturbation theory can only be applied to a single state. The multi–state (or quasi–degenerate) version of NEVPT2 will be implemented in MOLPRO in the near future.

An example is provided where the energies of the ground state and of the first $^1A_2$ ($n \rightarrow \pi^*$) excited state of formaldehyde are calculated.

```
***,
memory,20,m
file,1,h2co.int
file,2,h2co.wf
gthresh,energy=1.d-9
gthresh,orbital=1.d-8
gthresh,civec=1.d-8

geomtyp=zmat
geometry
 O,,            0.000000000,     0.000000000,    0.0196594609
 C,,            0.000000000,     0.000000000,    2.3248507925
 H1,,           0.000000000,     1.7597110083,   3.3972521023
 H2,,           0.000000000,    -1.7597110083,   3.3972521023
end

basis=6-31G*

{hf
wf,16,1,0}

{multi
closed,4,0,1,0
occ,6,2,4,0
wf,16,1,0
state,1
natorb,2140.2,state=1.1
}

{nevpt2,thrden=1.0d-10,thrvar=1.0d-10
core,2,0,0,0
closed,4,0,1,0
occ,6,2,4,0
orbit,2140.2,state=1.1
wf,16,1,0
state,1,1
}

{multi
closed,4,0,1,0
occ,6,2,4,0
wf,16,4,0
state,1
start,2140.2
natorb,2141.2,state=1.4
}
{nevpt2,thrden=1.0d-10,thrvar=1.0d-10
core,2,0,0,0
closed,4,0,1,0
occ,6,2,4,0
orbit,2141.2,state=1.4
wf,16,4,0
state,1,1
}
```

http://www.molpro.net/info/current/examples/form_nevpt2.com

23 MØLLER PLESSET PERTURBATION THEORY

Closed-shell Møller-Plesset perturbation theory up to full fourth order [MP4(SDTQ)] is part of the coupled-cluster program.

The commands MP2, MP3, MP4 perform the MP calculations up to the specified order (lower orders are included).

MP4;NOTRIPL; performs MP4(SDQ) calculations.

Normally, no further input is needed if the MPn card directly follows the corresponding HF-SCF. Otherwise, occupancies and orbitals can be specified as in the CI program. The resulting energies are stored in variables as explained in section 8.8.

Dual basis set calculations are possible for the closed-shell methods, see section 24.9.

23.1 Expectation values for MP2

One-electron properties can be computed as analytical energy derivatives for MP2. This calculation is much more expensive than a simple MP2, and therefore only done if an EXPEC card follows the MP2 card (the GEXPEC directive has no effect in this case). The syntax of the EXPEC card is explained in section 6.13. For an example, see section 24.6.1.

The density matrix can be saved using

{MP2;DM[,*record.ifil*]}

See also sections 24.7 and 24.8.

For changing the accuracy and other parameters of the CPHF calculation see section 23.3.

23.2 Polarizabilities and second-order properties for MP2

Analytical MP2 static dipole polarizabilities and other second order properties can be computed using the POLARI. By default the dipole operator is used, but other one-electron operators can be specified on the POLARI directive. Currently, this is only working without frozen core orbitals, i.e., CORE,0 must be given. The dipole polarizabilities are stored in the variables POLXX, POLXY, POLXZ, POLYY, POLYZ, POLZZ.

Example:

```
{ mp2
 core,0
 polari,dm,qm}
```

Computes the full tensors for the dipole and quadrupole operators.

23.3 CPHF for gradients, expectation values and polarizabilities

The accuracy and other parameters of the CPHF calculations necessary to compute gradients and response properties can be modified using the CPHF directive:

CPHF, [THRMIN=*thrmin*], [THRMAX=*thrmax*], [MAXIT=*maxit*], [SHIFT=*shift*], [*SAVE*=*record*], [*START*=*record*], [DIIS=*idiis*], [DISM=idism]

THRMIN — CPHF convergence threshold (default 1.d-6). THRESH and ACCU are aliases for this.

THRMAX	initial CPHF convergence threshold in geometry optimizations. Once the geometry is converged to a certain accuarcy (depending on OPTCONV), the threshold is stepwise reduced to THRMIN. The default is THRMAX=min(1.d-6,THRMIN*100); Values larger than 1.d-6 are ignored.
MAXIT	Maxmimum number of iterations (default 50).
SHIFT	Level shift for CPHF (default 0.1).
DIIS	First macroiteration in which DIIS is used (default 1)
DISM	First microiteration in which DIIS is used (default 1; in microiterations the core contribution is frozen).
SAVE	Record on which the CPHF solution can be saved for later restarts. The solution is saved automatically in geometry optimizations and frequency calculations.
START	Record from which initial guess is read. A starting guess is read automatically in geometry optimizations and frequency calculations.

23.4 Density-fitting MP2 (DF-MP2, RI-MP2)

DF-MP2,*options*

invokes the density fitted MP2 program. The present implementation works only without symmetry. RI-MP2 is an alias for the command DF-MP2.

The following options can be specified:

BASIS_MP2=*basis*:	Fitting basis set. *basis* can either refer to a basis set defined in a BASIS block, or to a default fitting basis set (only available for correlation consistent basis sets). If a correlation consistent orbital basis set is used, the corresponding MP2 fitting basis is generated by default. In all other cases, the fitting basis must be defined.
THRAO=*value*:	Screening threshold for 3-index integrals in the AO basis
THRMO=*value*:	Screening threshold for 3-index integrals in the MO basis
THROV=*value*:	Screening threshold for 2-index integrals of fitting basis.
THRPROD=*value*:	Screening product threshold for first half transformation.
SPARSE=*value*:	If Non-zero, use sparse algorithm in second-half transformation (default).

See section 15 for a more general description of density fitting.

At present, expectation values and gradients cannot be computed with DF-MP2, but work with the local variant DF-LMP2.

23.5 Spin-component scaled MP2 (SCS-MP2)

The spin-component scaled MP2 energy as proposed by Grimme (J. Chem. Phys. **118**, 9095 (2003)) is printed automatically using the default scaling factors (1.2 for antiparallel spin, 1/3 for parallel spin). These factors can be modified using the options SCSFACS and SCSFACT, respectively, i.e.

MP2, SCSFACS=*facs*, SCSFACT=*fact*

The SCS-MP2 total energy is stored in the variable EMP2_SCS. Gradients can be computed for SCS-MP2 by setting the option SCSGRD=1. This only operational for density fitted MP2, i.e. using

DF-MP2,[DF_BASIS=*fitbasis*],SCSGRD=1,[SCSFACS=*facs*], [SCSFACT=*fact*]

followed by FORCES or OPTG. In the latter case, the geometry is optimized using the SCS-MP2 energy.

24 THE CLOSED SHELL CCSD PROGRAM

Bibliography:

C. Hampel, K. Peterson, and H.-J. Werner, Chem. Phys. Lett. 190, 1 (1992)

All publications resulting from use of this program must acknowledge the above.

The CCSD program is called by the CISD, CCSD, BCCD, or QCI directives. CID or CCD can be done as special cases using the `NOSINGL` directive. The code also allows to calculate Brueckner orbitals (QCI and CCSD are identical in this case). Normally, no further input is needed if the `CCSD` card follows the corresponding HF-SCF. Optional `ORBITAL`, `OCC`, `CLOSED`, `CORE`, `SAVE`, `START`, `PRINT` options work as described for the MRCI program in section 20. The only special input directives for this code are `BRUECKNER` and `DIIS`, as described below.

The following options may be specified on the command line:

`NOCHECK`	Ignore convergence checks.
`DIRECT`	Do calculation integral direct.
`NOSING`	Do not include singly external configurations.
`MAXIT`=*value*	Maximum number of iterations.
`SHIFTS`=*value*	Denominator shift for update of singles.
`SHIFTP`=*value*	Denominator shift for update of doubles.
`THRDEN`=*value*	Convergence threshold for the energy.
`THRVAR`=*value*	Convergence threshold for CC amplitudes. This applies to the square sum of the changes of the amplitudes.

The convergence thresholds can also be modified using

`THRESH`,`ENERGY`=*thrden*,`COEFF`=*thrvar*

Convergence is reached if the energy change is smaller than *thrden* (default 1.d-6) *and* the square sum of the amplitude changes is smaller than *thrvar* (default (1.d-10). The `THRESH` card must follow the command for the method (e.g., `CCSD`) and then overwrites the corresponding global options (see `GTHRESH`, sec. 6.11).

The computed energies are stored in variables as explained in section 8.8. As well as the energy, the T_1 diagnostic (T. J. Lee and P. R. Taylor, Int. J. Quant. Chem. S23 (1989) 199) and the D_1 diagnostic (C. L. Janssen and I. M. B. Nielsen, Chem. Phys. Lett. 290 (1998), 423, and T. J. Lee. Chem. Phys. Lett. 372 (2003), 362) are printed and stored for later analysis in the variables T1DIAG and D1DIAG, respectively.

24.1 Coupled-cluster, CCSD

The command `CCSD` performs a closed-shell coupled-cluster calculation. Using the `CCSD(T)` command, the perturbative contributions of connected triple excitations are also computed.

If the CCSD is not converged, an error exit will occur if triples are requested. This can be avoided using the `NOCHECK` option:

```
CCSD(T),NOCHECK
```

In this case the (T) correction will be computed even if the CCSD did not converge. Note: `NOCHECK` has no effect in geometry optimizations or frequency calculations.

For further information on triples corrections see under RCCSD.

24.2 Quadratic configuration interaction, QCI

`QCI` or `QCISD` performs quadratic configuration interaction, QCISD. Using the `QCI(T)` or `QCISD(T)` commands, the contributions of connected triples are also computed by perturbation theory. Normally, no further input is needed if the `QCI` card follows the corresponding HF-SCF. Otherwise, occupancies and orbitals can be specified as in the CI program. For modifying DIIS directives, see section 24.5. For first-order QCISD and QCIDF(T) properties see section **??**.

For avoiding error exits in case of no convergence, see `CCSD(T)`.

24.3 Brueckner coupled-cluster calculations, BCCD

`BCCD`,[`SAVE`=*record*],[`PRINT`],[`TYPE`=,*type*]

`BCCD` performs a Brueckner coupled-cluster calculation and computes Brueckner orbitals. With these orbitals, the amplitudes of the singles vanish at convergence. Using the `BCCD(T)` command, the contributions of connected triples are also computed by perturbation theory. Normally, no further input is needed if the `BCCD` card follows the corresponding HF-SCF. Otherwise, occupancies and orbitals can be specified as in the CI program. `BRUECKNER` parameters can be modified using the `BRUECKNER` directive.

The Brueckner orbitals and approximate density matrix can be saved on a `MOLPRO` dump record using the `SAVE` option. The orbitals are printed if the `PRINT` option is given. `TYPE` can be used to specify the type of the approximate density to be computed:

`TYPE=REF`	Compute and store density of reference determinant only (default). This corresponds to the BOX (Brueckner orbital expectation value) method of Chem. Phys. Lett. **315**, 248 (1999).
`TYPE=TOT`	Compute and store density with contribution of pair amplitudes (linear terms). Normally, this does not seem to lead to an improvement.
`TYPE=ALL`	Compute and store both densities

Note: The expectation variables are stored in variables as usual. In the case that both densities are made, the variables contain two values, the first corresponding to `REF` and the second to `TOT` (e.g., DMZ(1) and DMZ(2)). If `TYPE=REF` or `TYPE=TOT` is give, only the corresponding values are stored.

For avoiding error exits in case of no convergence, see `CCSD(T)`.

24.3.1 The `BRUECKNER` directive

`BRUECKNER`,*orbbrk,ibrstr,ibrueck,brsfak*;

This directive allows the modification of options for Brueckner calculations. Normally, none of the options has to be specified, and the `BCCD` command can be used to perform a Brueckner CCD calculation.

orbbrk:	if nonzero, the Brueckner orbitals are saved on this record.
ibrstr:	First iteration in which orbitals are modified (default=3).
ibrueck:	Iteration increment between orbital updates (default=1).
brsfak:	Scaling factor for singles in orbital updates (default=1).

24.4 Singles-doubles configuration interaction, CISD

Performs closed-shell configuration interaction, CISD. The same results as with the CI program are obtained, but this code is somewhat faster. Normally, no further input is needed. For specifying DIIS directives, see section 24.5

24.5 The `DIIS` directive

`DIIS`,*itedis,incdis,maxdis,itydis*;

This directive allows to modify the DIIS parameters for CCSD, QCISD, or BCCD calculations.

itedis: First iteration in which DIIS extrapolation may be performed (default=2).

incdis: Increment between DIIS iterations (default=1).

maxdis: Maximum number of expansion vectors to be used (default=6).

itydis: DIIS extrapolation type. itydis=1 (default): residual is minimized. itydis=2: ΔT is minimized.

In addition, there is a threshold *THRDIS* which may be modified with the `THRESH` directive. DIIS extrapolation is only done if the variance is smaller than *THRDIS*.

24.6 Examples

24.6.1 Single-reference correlation treatments for H_2O

```
***,h2o test
memory,1,m                           !allocate 1 MW dynamic memory
geometry={o;h1,o,r;h2,o,r,h1,theta}  !Z-matrix geometry input
basis=vtz                            !cc-pVTZ basis set
r=1 ang                              !bond length
theta=104                            !bond angle
hf                                   !do scf calculation

text,examples for single-reference correlation treatments

ci                                   !CISD using MRCI code
cepa(1)                              !cepa-1 using MRCI code
mp2                                  !Second-order Moeller-Plesset
mp3                                  !Second and third-order MP
mp4                                  !Second, third, and fourth-order MP4(SDTQ)
mp4;notripl                          !MP4(SDQ)
cisd                                 !CISD using special closed-shell code
ccsd(t)                              !coupled-cluster CCSD(T)
qci(t)                               !quadratic configuration interaction QCISD(T)
bccd(t)                              !Brueckner CCD(T) calculation
---
```

http://www.molpro.net/info/current/examples/h2o_ccsd.com

24.6.2 Single-reference correlation treatments for N_2F_2

```
***,N2F2 CIS GEOMETRY (C2h)
rnn=1.223,ang                        !define N-N distance
rnf=1.398,ang                        !define N-F distance
alpha=114.5;                         !define FNN angle
geometry={N1
          N2,N1,rnn
          F1,N1,rnf,N2,alpha
          F2,N2,rnf,N1,alpha,F1,180}
basis=vtz                            !cc-pVTZ basis set
$method=[hf,cisd,ccsd(t),qcisd(t),bccd(t)]  !all methods to use
do i=1,#method                       !loop over requested methods
$method(i)                           !perform calculation for given methods
e(i)=energy                          !save energy in variable e
enddo                                !end loop over methods
table,method,e                       !print a table with results
title,Results for n2f2, basis=$basis  !title of table
```

http://www.molpro.net/info/current/examples/n2f2_ccsd.com

This calculation produces the following table:

```
 Results for n2f2, basis=VTZ

 METHOD          E             E-ESCF
 CISD       -308.4634948   -0.78283137
 BCCD(T)    -308.6251173   -0.94445391
 CCSD(T)    -308.6257931   -0.94512967
 QCISD(T)   -308.6274755   -0.94681207
```

24.7 Saving the density matrix

DM[,*record.ifil*];

The effective first order density matrix is computed an stored in record *record* on file *ifil*. This currently works for closed-shell MP2, QCISD, and QCISD(T). See also NATORB. Note that this is much more expensive than a simple energy calculation, since the response equations have to be solved. Note: CCSD first order properties without orbital relaxation contribution can be computed using

```
ccsd
expec,dm
```

The orbital relaxation can be included using

```
ccsd
core,0
expec,relax,dm
```

Currently, this works only without core, i.e., CORE,0 is required.

24.8 Natural orbitals

NATORB,[RECORD=]*record.ifil*,[PRINT=*nprint*],[CORE[=*natcor*]];

Calculate natural orbitals. This currently only works for closed-shell MP2 and QCISD. The number of printed external orbitals in any given symmetry is *nprint*) (default 2). *nprint=-1* suppressed the printing. The natural orbitals and the density matrix are saved in a dump record *record* on file *ifil*. If *record.ifil* is specified on a DM card (see above), this record is used. If different records are specified on the DM and NATORB cards, an error will result. The record can also be given on the SAVE card. Note that the effective density matrix of non-variational methods like MP2 or QCISD does not strictly behave as a density matrix. For instance, it has non-zero matrix elements between core and valence orbitals, and therefore core orbitals are affected by the natural orbital transformation. Also, occupation numbers of core orbitals can be larger than 2.0. If CORE is given (*natcor=1*), the core orbitals are frozen by excluding them from the natural orbital transformation.

24.9 Dual basis set calculations

Dual basis set calculations are possible with the closed-shell MP2 and CCSD codes (conventional and local, also with density fitting where available). Normally this means that the Hartree-Fock calculation is done with a smaller basis set than the correlation calculation. In MOLPRO, two possibilities exist: the recommended one is to perform the HF and MP2 or CCSD calculations using the same basis set, and only omit higher angular momentum functions in the HF. This means that the resulting HF orbitals can be used directly in the correlation calculation. Alternatively, one can use entirely different basis sets in HF and the correlation calculation; in this case the orbitals in the correlation calculation are determined by a least square fit to the HF orbitals. This is less efficient (in particular in fully direct calculations) and somewhat less accurate. In any case, a new Fock matrix is computed in the MP2/CCSD program and block diagonalized in the occupied and virtual orbital subspaces. A perturbative singles correction is applied in the MP2 in order to reduce the HF basis set error.

Typically, the input is as follows:

```
basis=vtz(d/p)   !triple zeta basis set without f on heavy atoms and without
hf               !Hartree-Fock in the small basis
basis=vtz        !full cc-pVTZ basis set to be used in ccsd
ccsd(t),dual=1   !ccsd calculation
```

The option `dual=1` is required, otherwise the program will stop with an error message saying that the basis set of the reference orbitals is inconsistent. This is a precaution in order to avoid unexpected results.

Similarly, this works for other closed-shell single reference methods such as `MP2, QCISD, MP2-F12, CCSD-F12`, and for the local variants `LMP2, LQCISD, LCCSD` in either conventional or direct mode. Furthermore, dual basis set `DF-LMP2, DF-LCCSD, DF-LMP2-F12` calculations are possible.

25 EXCITED STATES WITH EQUATION-OF-MOTION CCSD (EOM-CCSD)

Excitation energies for singlet states can be computed using equation-of-motion (EOM) approach. For the excitation energies the EOM-CCSD method gives the same results as linear response CCSD (LR-CCSD) theory. Accurate results can only be expected for singly excited states. The states to be computed are specified on an `EOM` input card, which is a subcommand of `CCSD`. The following input forms are possible

`EOM`, *state1, state2, state3, ...*

Computes the given states. Each state is specified in the form *number.sym*, e.g., 5.3 means the fifth state in symmetry 3. Note that state 1.1 corresponds to the ground state CCSD wavefunction and is ignored if given.

`EOM`, $-n1.sym1$, $-n2, sym2, \ldots$

computes the first $n1$ states in symmetry *sym1*, $n2$ in *sym2* etc.

`EOM`, $n1.sym1$, $-n2, sym1, \ldots$

computes states $n1$ through $n2$ in symmetry *sym1*.

The different forms can be combined, e.g.,

`EOM`, -3.1, 2.2, 2.3, -5.3

computes states 1-3 in symmetry 1, the second excited state in symmetry 2, and the second through fifth excited states in symmetry 3. Note that state 1.1 is the ground-state CCSD wavefunction.

By default, an error exit will result if the CCSD did not converge and a subsequent EOM calculation is attempted. The error exit can be avoided using the `NOCHECK` option on the CCSD command (see also CCSD(T)).

25.1 Options for `EOM`

Normally, no further input is needed for the calculation of excitation energies.

EOM-CCSD amplitudes can be saved using `SAVE=`*record.ifil*. The vectors will be saved after every refreshing of the iteration space and at the end of the calculation. The calculation can be restarted from the saved vectors, if `START=`*record.ifil* is specified. The set of vectors to be computed can be different in old and restarted calculations. However, if both cards (`SAVE` and `START`) are specified and the records for saving and restarting are identical, the sets of vectors should be also identical, otherwise chaos. The identical `SAVE` and `START` records can be useful for potential energy surfaces calculations, see section 25.4.1.

By default, only excitation energies are calculated, since the calculation of properties is about two times as expensive, as the calculation of energies only. The one-electron properties and transition moments (expectation type, as defined in: J.F. Stanton and R.J. Bartlett, J. Chem. Phys., **98** 7029 (1993)) can be calculated by adding `TRANS=1` to `EOM` card. The CCSD ground state is treated as a special case. If `RELAX` option is specified on the `EXPEC` card, also the relaxed one-electron density matrix is calculated for the ground state. (Currently, the relaxed CCSD density matrix is available for all-electron calculations only.) By default, dipole moments are calculated. Other required properties can be specified using `EXPEC` card. The excited state densities are saved if a `DM` card is present. For an example see section 25.4.2. If properties are calculated, they are saved in `MOLPRO` variables, e.g. the x-component of the dipole moment is saved in `DMX`, its pure electron part in `DMXE`, transition moment – in `TRDMX` (left and right transition moments are stored separately). If `TRANS=2`, transition moments among excited states are also calculated. If `DENSAVE=`*record.ifil* is specified, excited-state densities (and transition densities, if `TRANS=2`) are saved to *record.ifil*, otherwise they are saved to the record given in `DM` card.

When properties are needed, the left EOM-CCSD wave functions are calculated first. It is possible to use them as starting guesses for the right EOM-CCSD wave functions. This option is controlled by `STARTLE` (default 0). If `STARTLE`=1, left vectors are just used as a start for right vectors; if `STARTLE`=2, starting vectors, obtained from the left vectors are additionally biorthogonalized to the left vectors; finally, if `STARTLE`=3, also the final right vectors are biorthogonalized to the left vectors. The last possibility is of particular importance for degenerate states.

It is possible to make the program to converge to a vector, which resembles a specified singles vector. This option is switched on by `FOLLOW=`*n* card (usually *n*=2 should be set). `FOLLOW` card should be always accompanied with `EXFILE=`*record.ifil* card, where *record.ifil* contains singles vectors from a previous calculation, see section 25.4.3.

25.2 Options for `EOMPAR` card

Normally, no further input is needed. However, some defaults can be changed using the `EOMPAR` directive:

`EOMPAR`, `key1=`*value1*, `key2=`*value2*,...

where the following keywords `key` are possible:

`MAXDAV=`*nv*	Maximum value of expansion vectors per state in Davidson procedure (default 20).
`INISINGL=`*ns*	Number of singly excited configurations to be included in initial Hamiltonian (default 20; the configurations are ordered according to their energy). Sometimes `INISINGL` should be put to zero in order to catch states dominated by double excitations.
`INIDOUBL=`*nd*	Number of doubly excited configurations to be included in initial Hamiltonian (default 10).
`INIMAX=`*nmax*	Maximum number of excited configurations to be included in initial Hamiltonian. By default, $nmax = ns + nd$.
`MAXITER=`*itmax*	Maximum number of iterations in EOM-CCSD (default 50).
`MAXEXTRA=`*maxex*	Maximum number of extra configurations allowed to be included in initial Hamiltonian (default 0). In the case of near degeneracy it is better to include a few extra configurations to avoid a slow convergence.
`EOMLOCAL=`*eoml*	If set to 0, non-local calculation (default). `EOMLOCAL`=1 switchs on the local module (experimental!).
`INIMAX=`*ini*	Number of CSFs included in initial Hamiltonian, used only if `INISINGL` and `INIDOUBL` are both zero.

All keywords can be abbreviated by at least four characters.

25.3 Options for EOMPRINT card

The following print options are mostly for testing purposes and for looking for the convergence problems.

EOMPRINT, key1=*value1*, key2=*value2*,...

where the following keywords key are possible:

DAVIDSON=*ipr*	Information about Davidson procedure: *ipr=1* print results of each "small diagonalization" *ipr=2* also print warning information about complex eigenvalues *ipr=3* also print hamiltonian and overlap matrix in trial space.
DIAGONAL=*ipr*	Information about configurations: *ipr=1* print the lowest approximate diagonal elements of the transformed hamiltonian *ipr=2* print orbital labels of important configurations *ipr=3* print all approximate diagonal elements *ipr=4* also print the long form of above.
PSPACE=*ipr*	Print information about the initial approximate hamiltonian: *ipr=2* print the approximate hamiltonian used to find the first approximation.
HEFF=*ipr*	Print information about effective Hamiltonian: *ipr=2* print columns of effective hamiltonian and overlap matrix in each iteration
RESIDUUM=*ipr*	Print information about residual vectors: *ipr=-1* no print in iteration *ipr=0* print energy values + residuum norm (squared) for each iteration (default) *ipr=1* also print warning about complex eigenvalue, and a warning when no new vectors is added to the trial space due to the too small norm of the residuum vector. *ipr=2* also print how many vectors are left
LOCEOM=*ipr*	*ipr=1* prints overlaps of sample and tested vectors in each iteration, if FOLLOW card is present. Increasing *ipr* switches on more and more printing, mostly related to the local EOM-CCSD method.
POPUL=*ipr*	if *ipr=1*, do a population analysis of the singles part of the rhs EOM-CCSD wave function. By default the Löwdin method is used. The Mulliken analysis can be forced by adding MULLPRINT=1 to EOM card. Note that a more correct (but more expensive) approach is to calculate and analyse the EOM-CCSD density matrix, see section 25.1.
INTERMED=*ipr*	Print intermediates dependent on ground state CCSD amplitudes: *ipr=0* no print (default) *ipr=1* print newly created intermediates *ipr=2* also print more intermediates-related information

25.4 Examples

25.4.1 PES for lowest excited states for hydrogen fluride

This example shows how to calculate potential energy surfaces for several excited states using restart from a previous calculation.

```
***, PES for several lowest states of hydrogen fluoride
memory,2,m
basis=avdz                                     ! define basis set
geometry={h;f,h,r}                             ! z-matrix
r=0.8 Ang                                      ! start from this distance
do n=1,100                                     ! loop over distances
rr(n)=r                                        ! save distance for table
hf                                             ! do SCF calculation
ccsd                                           ! do CCSD calculation, try to restart
start,4000.2,save,4000.2                       ! and save final T amplitudes
eom,-2.1,-1.2,-1.4,start=6000.2,save=6000.2    ! do EOM-CCSD calculation, try to restart
                                               ! and save final excited states' amplitudes



ebase(n)=energy(1)                             ! save ground state energy for this geometry
e2(n)=energy(2)-energy(1)                      ! save excitation energies for this geometry
e3(n)=energy(3)-energy(1)
e4(n)=energy(4)-energy(1)
r=r+0.01                                       ! increment distance
enddo                                          ! end of do loop
table,rr,ebase,e2,e3,e4                        ! make table with results
digits,2,8,5,5,5,5,5,5,5,5                     ! modify number of digits
head,R(Ang),EGRST,E_EXC(2.1),E_EXC(1.2),E_EXC(1.4)! modify headers of table
! title of table
title,EOM-CCSD excitation energies for hydrogen fluoride (in hartree), basis $basis
save,hf_eom_ccsd.tab                           ! save table in file
```

http://www.molpro.net/info/current/examples/hf_eom_pes.com

This calculation produces the following table:

```
 EOM-CCSD excitation energies for hydrogen fluoride (in hartree), basis AVDZ

   R(ANG)        EGRST     E_EXC(2.1)  E_EXC(1.2)  E_EXC(1.4)
     0.80   -100.23687380     0.56664     0.41204     0.56934
     0.81   -100.24094256     0.56543     0.40952     0.56812
     0.82   -100.24451598     0.56422     0.40695     0.56690
```

etc.

25.4.2 EOM-CCSD transition moments for hydrogen fluoride

This example shows how to calculate and store CCSD and EOM-CCSD density matrices, calculate dipole and quadrupole moments (transition moments from the ground to excited states are calculated), and how to use the EOM-CCSD excited state density for Mulliken population analysis.

```
***, Properties and transition moments for several lowest states of hydrogen fluoride
memory,2,m
basis=avdz                                        ! define basis set
geometry={h;f,h,r}                                ! z-matrix
r=0.92 Ang                                        ! define distance

hf                                                ! do SCF calculation
{ccsd                                             ! do CCSD calculation
dm,5600.2                                         ! density matrices will be stored here
expec,qm                                          ! require quadrupole moments
eom,-3.1,-2.2,-2.3,-2.4,trans=1}                  ! do EOM-CCSD calculation + properties

pop;density,5600.2,state=2.4                      ! make population analysis for state 2.4
```

http://www.molpro.net/info/current/examples/hf_eom_prop.com

This calculation produces the following table:

```
------------------------------------------------------------------------------
                        Final Results for EOM-CCSD
                             (moments in a.u.)
------------------------------------------------------------------------------

  State    Exc. Energy (eV)          X              Y              Z
   2.1          14.436

 Right transition moment         0.00000000     0.00000000     0.65349466
 Left  transition moment         0.00000000     0.00000000     0.68871635
 Dipole strength                 0.45007246
 Oscillator strength             0.15917669
 Dipole moment                   0.00000000     0.00000000     0.88758090
```

etc.

25.4.3 Calculate an EOM-CCSD state most similar to a given CIS state

This example shows how to force the convergence of the EOM-CCSD program to a state, which resembles at most a given CIS state.

```
***, EOM-CCSD, vector following procedure
memory,2,m
basis=avdz                                   ! define basis set
geometry={h;f,h,r}                           ! z-matrix
r=0.92 Ang                                   ! define distance
hf;save,2100.2                               ! do SCF calculation, save orbitals
cis,-4.4,exfile=6000.2                       ! do CIS calculation, save amplitudes
ccsd;save,4000.2                             ! do CCSD calculation, save amplitudes
eom,-4.4,checkovlp=1,exfile=6000.2           ! do EOM-CCSD calculation,
                                             ! check overlap of singles with CIS vectors
                                             ! stored in record given in exfile
eompar,inisingl=200,inidoubl=0               ! for first approximation take 200 single CSF
                                             ! of approximate hamiltonian
ccsd;start,4000.2                            ! do CCSD calculation, try to restart
eom,2.4,follow=2,exfile=6000.2,checkovlp=1   ! do EOM-CCSD calculation for state closest
                                             ! to 2.4 CIS state, check overlap of singles
                                             ! with CIS vectors stored in exfile
eompar,inisingl=200,inidoubl=0
eomprint,loce=1                              ! print overlaps of sample and EOM vectors in
                                             ! each iteration
```

http://www.molpro.net/info/current/examples/hf_eom_conv.com

In this example the CIS state 2.4 corresponds to the EOM-CCSD state 1.4!

25.5 Excited states with CIS

Excitation energies can also be calculated using the Configuration-Interaction Singles (CIS) method. By default, singlet excited states are calculated. Triplet excited states can be obtained by setting `triplet`=1 in `EOM` card. This method cannot be expected to give accurate results, but can be used for quite large molecules. The states to be computed are specified as in `EOM`. Setting `trans`=1 switches on the calculation of one-electron properties. By default, dipole moments are calculated. Other required properties can be specified using `EXPEC` card. Dipole transition moments are also calculated.

```
hf
cis,-3.1,1.2,trans=1
```

25.6 First- and second-order properties for CCSD

First-order and frequency-dependent second-order properties, derived from the expressions based on the expectation value of a one-electron operator, can be obtained with the `CPROP` directive for the closed-shell CCSD method. The methods are described in the following papers:

[1] B. Jeziorski and R. Moszynski, *Int. J. Quantum Chem.*, **48**, 161 (1993);
[2] T. Korona and B. Jeziorski, *J. Chem. Phys.*, **125**, 184109 (2006);
[3] R. Moszynski, P. S. Żuchowski and B. Jeziorski, *Coll. Czech. Chem. Commun.*, **70**, 1109 (2005);
[4] T. Korona, M. Przybytek and B. Jeziorski, *Mol. Phys.*, **104**, 2303 (2006).

Note that properties obtained from the expectation-value expression with the coupled cluster wave function are not equivalent to these derived from gradient or linear-response methods, although the results obtained with both methods are quite similar.

For the first-order properties the one-electron operators should be specified in the EXPEC card, while for the second-order properties – in the POLARI card. A density can be saved by specifying the DM card.

For the first-order properties the option XDEN=1 should be always given. Other options specify a type of the one-electron density, which can be either the density directly derived from the expectation-value expression, see Eq. (8) of Paper 2, or the modified formula, rigorously correct through the $\mathscr{O}(W^3)$ Møller-Plesser (MP) order, denoted as $\bar{X}(3)_{\text{resp}}$ in Papers 1 and 2. In the first case the option PROP_ORDER=*n* can be used to specify the approximation level for single and double excitation parts of the so-called *S* operator (see [2], Eq. (9)); $n = \pm 2, \pm 3, \pm 4$, where for a positive *n*: all approximations to *S* up to *n* are used, and for a negative *n* only a density with *S* obtained on the $|n|$ level will be calculated. Another option related to the *S* operator is HIGHW=*n*, where $n = 0, 1$; if *n*=0, some parts of S_1 and S_2 operators of a high MP order are neglected. Below an example of a standard use of this density is given:

```
CPROP,XDEN=1,PROP_ORDER=-4,HIGHW=1
```

In the second case the options X3RESP=1 and the CPHF,1 card (or alternatively the EXPEC card) should be specified,

```
CPROP,XDEN=1,X3RESP=1;CPHF,1
```

For the second-order properties always the following options should be given:

```
CPROP,PROPAGATOR=1,EOMPROP=1
```

The recommended CCSD(3) model from Paper 4 requires that additionally the PROP_ORDER=3 and HIGHW=0 options are specified. Frequencies for dynamic properties (in atomic units) should be given in variables OMEGA_RE (real parts) and OMEGA_IM (imaginary parts). If one of these arrays is not given, it is filled with zeros. Other options for the second-order properties involve

OMEGAG — (default 0.3). There are two linear-equation solvers, OMEGAG is a minimum frequency, for which the second solver (working for large frequencies) is used.

DISPCOEF=*n* — if $n > 0$, calculate dispersion integrals for the van der Waals coefficients with operators given in the POLARI card, using *n* as a number of frequencies for the numerical integration. In this case the frequency values given in OMEGA_RE and OMEGA_IM are ignored. If two molecules are calculated in the same script one after another, also the mixed dispersion integrals are calculated. The isotropic C_6 coefficient is stored in a variable DISPC6. All necessary informations for the calculation of dispersion integrals are written to the ascii file *name.dispinfo*, where *name* is the name of the MOLPRO script.

THRPROPAG — if given, use this threshold as a convergence criterion for the linear-equation solver for the first-order perturbed CCSD amplitudes.

STARTT1=*n* — various start options for the iterative linear-equation solver for the first-order perturbed CCSD amplitudes, the most useful is $n = 0$ (zero start) and $n = 7$ (start from the negative of the r.h.s. vector rescaled by some energetic factors dependent on the diagonal of the Fock matrix and the specified frequency).

26 OPEN-SHELL COUPLED CLUSTER THEORIES

Spin unrestricted (RHF-UCCSD) and partially spin restricted (RHF-RCCSD) open-shell coupled cluster theories as described in J. Chem. Phys. **99** (1993) 5219 (see also erratum, J. Chem. Phys., **112** (2000) 3106) are available in MOLPRO. In both cases a high-spin RHF reference wavefunction is used. No coupled cluster methods based on UHF orbitals are implemented in MOLPRO (the only correlation method in MOLPRO which uses UHF orbitals is UMP2). In the description that follows, the acronyms RCCSD and UCCSD are used, but the theories should normally be referred to as RHF-RCCSD, RHF-UCCSD, in order to distinguish them from alternative ansätze based on spin-unrestricted orbitals. The program will accept either the full or abbreviated acronyms as input commands.

In the RCCSD theory certain restrictions among the amplitudes are introduced, such that the linear part of the wavefunction becomes a spin eigenfunction (this is not the case in the UCCSD method, even if an RHF reference function is used). At present, the implementation of RCCSD is only preliminary, and no CPU time is saved by as compared to UCCSD. However, improved algorithms, as described in the above publication, are currently being implemented, and will be available in the near future.

The input is exactly the same as for closed-shell CCSD, except that RCCSD or UCCSD are used as keywords. By default, the open-shell orbitals are the same as used in the RHF reference function, but this can be modified using OCC, CLOSED, and WF cards.

Perturbative triples corrections are computed as follows:

RCCSD(T), UCCSD(T)	triples corrections are computed as defined by J. D. Watts, J. Gauss and R. J. Bartlett, J. Chem. Phys. **98** 8718 (1993).
RCCSD[T], UCCSD[T]	corrections are computed without contributions of single excitations (sometimes called CCSD+T(CCSD)) .
RCCSD-T, UCCSD-T	triples corrections are computed as defined by M. J. O. Deegan and P. J. Knowles, Chem. Phys. Letters 227 (1994) 321.

In fact, all three contributions are always computed and printed. The following variables are used to store the results (here CCSD stands for either UCCSD or RCCSD):

ENERGY	total energy for method specified in the input.
ENERGC	total CCSD energy without triples.
ENERGT(1)	total CCSD(T) energy.
ENERGT(2)	total CCSD[T] energy.
ENERGT(3)	total CCSD-T energy.

It should be noted that in open-shell cases the triples energy slightly depends on the treatment of core orbitals. In MOLPRO pseudo-canonical alpha and beta spin orbitals (Chem. Phys. Letters **186** (1991) 130) are generated by block-diagonalizing the corresponding Fock matrices in the space of valence orbitals, leaving frozen core orbitals untouched. Some other programs include the frozen core orbitals in the canonicalization and transformation. Because of core-valence mixing this leads to slightly different energies. Neither of the two methods can be regarded as better or more justified — it is just a matter of definition. However, the method in MOLPRO is more efficient since the subsequent integral transformation involves only valence orbitals and no core orbitals.

27 The MRCC program of M. Kallay (MRCC)

An interface exists to use the MRCC program of M. Kallay and J. Gauss within Molpro. The license and source code of the MRCC program must be obtained from Mihaly Kallay http://www.mrcc.hu/. Currently, only single reference methods with RHF reference functions are supported. Perturbative methods and CCn methods are only available for closed-shell. Furthermore, only serial execution is supported under MOLPRO, i.e. the mpp version cannot be used.

27.1 Installing MRCC

A file mrcc.tar.gz will be provided by by M. Kallay. It should be copied to the top level Molpro directory, then unpacked and built in the following way:

```
mkdir mrcc
tar -C mrcc -xzf mrcc.tar.gz
make mrcc
```

which will compile the MRCC program and link the MRCC executables into the MOLPRO bin directory which from here are automatically called by MOLPRO. Orbitals and other input information are communicated via external files, transparent to the user. Once the program is installed, please run make mrcctest in testjobs directory.

27.2 Running MRCC

The MRCC program is invoked by the command

MRCC,*options*
directives

The available options summarized in Table 11. For a detailed description please refer to the MRCC manual of M. Kallay (file ”manual” the mrcc directory)

In MOLPRO the method to be used can be given as a string (option METHOD=*string*). The available methods and the corresponding MRCC input parameters (see MRCC manual) as specified in Table 12.

Directives are usually not necessary, but the CORE, OCC, ORBITAL, MAXIT, directives work as in the MOLPRO CCSD program. In addition, the number of states can be given on a STATE directive and this has the same meaning as the EOM_NSTATES option.

Table 11: Options for MRCC

Option	Alias	Default value[a]	Meaning
METHOD	CALC	CC(n)	Computational method. See Table 12.
EXCITATION	LEVEL	-1	Excitation level in cluster operator
RESTART_CC	RESTART	0	Restart option. If 1, restart with previous amplitudes.
DIRECTORY	DIR	' '	Subdirectory in which MRCC runs (necessary for restart jobs)
EOM_NSING	NSING	-1	Number of excited singlet states in closed-shell case
EOM_NTRIP	NTRIP	0	Number of excited triplet states in closed-shell case
EOM_NSTATES	NDOUB	-1	Number of states in open shell case.
SYMM	SYMMETRY	-1	Symmetry of excited states
DENSITY	IDENS	0	Parameter for density calculation
HF		1	1 for canonical Hartree-Fock orbitals, 0 otherwise
SPATIAL		1	0 for spin-restricted orbitals, 1 for spin-unrestricted orbitals
NACTO		0	Number of active occupied orbitals
NACTV		0	Number of active virtual orbitals
SACC		0	Spin-adapted coupled cluster
DBOC		0	Diagonal BO correction
MEMORY		-1	Memory
TOL	ENERGY	-1.0	Energy convergence threshold
FREQ		0.0	Frequency for dynamic polarizabilities
FILE		fort	Name for MRCC fortran files
CONVER	ICONV	0	See mrcc manual
CS		1	See mrcc manual
DIAG		0	See mrcc manual
MAXEX		0	See mrcc manual

a) -1 means default value taken from MOLPRO

Table 12: Methods available in the MRCC program

Key	MRCC parameters METHOD	LEVEL	Notes
CI(n) configuration interaction methods			
CISD	0	2	
CISDT	0	3	
CISDTQ	0	4	
CI(N)	0	N	Specify excitation level N using LEVEL
CC(N) coupled cluster methods			
CCSD	1	2	
CCSDT	1	3	
CCSDTQ	1	4	
CC(N)	1	N	Specify excitation level N using LEVEL
CC(N-1)[N] coupled cluster methods			
CCSD[T]	2	3	
CCSDT[Q]	2	4	
CC(N-1)[N]	2	N	Specify excitation level N using LEVEL
CC(N-1)(N) coupled cluster methods. Also computes [n] corrections			
CCSD(T)	3	3	
CCSDT(Q)	3	4	
CC(N-1)(N)	3	N	Specify excitation level N using LEVEL
CC(n-1)(n)_L methods (also computes (n) and [n] corrections)			
CCSD(T)_L	4	3	
CCSDT(Q)_L	4	4	
CC(N-1)(N)_L	4	N	Specify excitation level N using LEVEL
CC(n)-1a methods			
CCSDT-1A	5	3	
CCSDTQ-1A	5	4	
CC(N)-1A	5	N	Specify excitation level N using LEVEL
CC(n)-1b methods			
CCSDT-1B	6	3	
CCSDTQ-1B	6	4	
CC(N)-1B	6	N	Specify excitation level N using LEVEL
CCn methods (only for ground states)			
CC3	7	3	
CC4	7	4	
CCN	7	N	Specify excitation level N using LEVEL
CC(n)-3 methods			
CCSDT-3	8	3	
CCSDTQ-3	8	4	
CC(N)-3	8	N	Specify excitation level N using LEVEL

Examples: Closed-shell ground-state calculations for H2O:

```
***,mrcc calculations for h2o
memory,8,m
gthresh,energy=1.d-8

geometry={
o;h1,o,r;h2,o,r,h1,theta}
theta=104
r=1 ang
basis=vdz

hf
mrcc,method=cc3;                   !CC3 calculation
method(1)=program
e(1)=energy                        !the final energy is returned in variable energy

ccsd(t)                            !CCSD(T) calculation using Molpro
method(2)='CCSD(T) (MOLPRO)'
e(2)=energy

mrcc,method=ccsd(t)                !CCSD(T) calculation using MRCC
method(3)='CCSD(T) (MRCC)'
e(3)=energy

mrcc,method=ccsdt,dir=mrccdir      !CCSDT calculation, run in directory mrccdir
method(4)=program
e(4)=energy

mrcc,method=ccsdt(q),restart=1,dir=mrccdir !CCSDT(Q) calculation
                                           !restart with previous amplitudes
method(5)=program
e(5)=energy

mrcc,method=CC(n),excitation=4,restart=1,dir=mrccdir !CCSDTQ calculation
method(6)=program
e(6)=energy

table,method,e
```

http://www.molpro.net/info/current/examples/h2o_mrcc.com

This yields

```
 METHOD                 E
 CC3               -76.23912734
 CCSD(T) (MOLPRO)  -76.23905150
 CCSD(T) (MRCC)    -76.23905150
 CCSDT             -76.23922746
 CCSDT(Q)          -76.23976632
 CCSDTQ            -76.23973043
```

Excitation energies for H2O:

```
***,h2o excitation energies
memory,8,m
gthresh,energy=1.d-8
geometry={
o;h1,o,r;h2,o,r,h1,theta}
theta=104
r=1 ang
basis=vdz
hf

ii=0
s=2                        !number of states in each symmetry
do sym=1,4                 !loop over irreps
ccsd;eom,-(s+0.1*sym);$p=molpro;save_energy
mrcc,method=ccsd, symm=sym,nstates=2;$p=mrcc;save_energy
mrcc,method=ccsdt,symm=sym,nstates=2;$p=mrcc;save_energy
s=1
enddo

{table,method,prog,states,e,exc
 sort,3}

save_energy={       !procedure to save results in variables
!nogprint,variable
e1=energy(1)
do i=1,#energy
ii=ii+1
e(ii)=energy(i)
method(ii)=program
prog(ii)=p
states(ii)=i+0.1*sym
exc(ii)=(e(ii)-e1)*toev
end do
}
```

http://www.molpro.net/info/current/examples/h2o_mrcc_eom.com

This yields

METHOD	PROG	STATES	E	EXC
CCSD	MOLPRO	1.1	-76.23580212	0.000
CCSD	MRCC	1.1	-76.23580212	0.000
CCSDT	MRCC	1.1	-76.23922746	0.000
CCSD	MOLPRO	1.2	-76.23580212	0.000
CCSD	MRCC	1.2	-76.23580212	0.000
CCSDT	MRCC	1.2	-76.23922746	0.000
CCSD	MOLPRO	1.3	-76.23580212	0.000
CCSD	MRCC	1.3	-76.23580212	0.000
CCSDT	MRCC	1.3	-76.23922746	0.000
CCSD	MOLPRO	1.4	-76.23580212	0.000
CCSD	MRCC	1.4	-76.23580212	0.000
CCSDT	MRCC	1.4	-76.23922746	0.000
CCSD	MOLPRO	2.1	-75.85033256	10.489
CCSD	MRCC	2.1	-75.85033257	10.489
CCSDT	MRCC	2.1	-75.85316687	10.505
CCSD	MOLPRO	2.2	-75.95093334	7.752
CCSD	MRCC	2.2	-75.95093335	7.752
CCSDT	MRCC	2.2	-75.95299013	7.789

```
 CCSD      MOLPRO     2.3   -75.77630664   12.504
 CCSD      MRCC       2.3   -75.77630665   12.504
 CCSDT     MRCC       2.3   -75.77972816   12.504
 CCSD      MOLPRO     2.4   -75.87776149    9.743
 CCSD      MRCC       2.4   -75.87776150    9.743
 CCSDT     MRCC       2.4   -75.88051189    9.761
```

Open-shell ground-state calculations for O2:

```
***,O2 tests
memory,8,m
gthresh,energy=1.d-8

geometry={o1;o2,o1,r1}
r1=2.2
set,state=1,symmetry=4,spin=2  ! Triplet sigma- state
basis=vdz

rhf
uccsd(t)
method(1)='UCCSD(T) MOLPRO'
e(1)=energy

rccsd(t)
method(2)='RCCSD(T) MOLPRO'
e(2)=energy

mrcc,method=ccsdt,dir=mrccdir
method(3)='CCSDT MRCC'
e(3)=energy

mrcc,method=ccsdtq,restart=1,dir=mrccdir
method(4)='CCSDT MRCC'
e(4)=energy

table,method,e
```

http://www.molpro.net/info/current/examples/o2_mrcc.com

This yields

```
 METHOD                  E
 UCCSD(T) MOLPRO   -149.9815472
 RCCSD(T) MOLPRO   -149.9812566
 CCSDT MRCC        -149.9816705
 CCSDT MRCC        -149.9832255
```

28 SMILES

SMILES is a package for molecular integrals with Slater functions implemented by J. Fernandez Rico, R. Lopez, G. Ramirez, I. Ema, D. Zorrilla and K.Ishida. It combines several techniques for the evaluation of the different types of integrals, a summary of which can be found in J. Fernandez Rico, R. Lopez, G. Ramirez, I. Ema, J. Comput. Chem., 25, 1987-1994 (2004) and J. Fernandez Rico, I. Ema, R. Lopez, G. Ramirez and K. Ishida, *SMILES: A package for molecular calculations with STO* in Recent Advances in Computational Chemistry. Molecular Integrals over Stater Orbitals, Telhat Ozdogan and Maria Belen Ruiz eds., ISBN ...

This code is not included in binary versions of Molpro, and by default the code is not included when building from source code; one should use the `-slater` configure option to enable compilation of the code.

The SMILES module is invoked by the `intyp='SLATER'` card. Name for ancillary files generated by the package can be supplied by `SLFILES=`*filename*. Default name is 'slscratch'.

Three-center two-electron integrals of types $(AB|AC)$ and $(AB|CD)$ are computed by means of Gaussian expansions (STO-nG). The default length of the expansions is 9 (STO-9G). However, integrals obtained with STO-9G expansions may have not sufficient accuracy for post-HF calculations, specially with high quality basis sets. In these cases, the length of the expansions can be changed setting the variable `NGSSTO=` *number of gaussians*. Though a maximum of `NGSSTO=30` is allowed, the lengths of the expansions actually available in the package depend on the (n,l) quantum numbers. If an expansion not included is required, the program takes the largest currently available.

Program limitations: In the current version, a maximum of 511 basis functions is allowed, and contracted functions cannot be used.

28.1 INTERNAL BASIS SETS

The following internal basis sets are available.

Table 13: Internal basis sets in SMILES

	Atoms			
Basis set	H-He	Li-Be	B-Ne	Na-Ar
VB1	[3,1]	[5,1]	[5,3,1]	—
CVB1	[3,1]	[6,2]	[6,4,1]	—
FVB1	[3,2,1]	[5,3,1,1]	[5,4,2,1]	—
ZVB1	[3,1]	[4,3,1]	[4,3,1]	[6,5,1]
VB2	[4,2,1]	[6,2,1]	[6,4,2,1]	—
CVB2	[4,2,1]	[7,3,2]	[7,5,3,1]	—
ZVB2	[4,2,1]	[5,4,2,1]	[5,4,2,1]	[7,6,2,1]
VB3	[5,3,2,1]	[7,3,2,1]	[7,5,3,2,1]	—
CVB3	[5,3,2,1]	[8,4,3,2]	[8,6,4,3,1]	—
ZVB3	[5,3,2,1]	[6,5,3,2,1]	[6,5,3,2,1]	[8,7,3,2,1]

28.2 EXTERNAL BASIS SETS

External basis sets can be supplied in a file that must be located in the working directory. Each record will contain the following data (free format):

I N L EXP NG

where:

I: atom type index (integer)
N: principal quantum number (integer)
L: angular quantum number (integer)
EXP: exponent (double precision)
NG: number of gaussians for the $(AB|AC)$ and $(AB|CB)$ integrals (integer)

Atom type index is used to establish the correspondence between the basis functions and the centers (atoms). All records having the same value of I will define the basis set for all the atoms of the corresponding type.

Example:

For a calculation on the CO_2 molecule with the following geometry data:

```
Geometry={
   3

C1, 0.000000000000E+00, 0.000000000000E+00, 0.000000000000E+00
O2, 0.000000000000E+00, 0.000000000000E+00,-0.117963799946E+01
O3, 0.000000000000E+00, 0.000000000000E+00, 0.117963799946E+01
}
```

Clementi and Roetti's Single Zeta basis set could be supplied in an external file like:

```
1   1   0         5.67263   12
1   2   0         1.60833   12
1   2   1         1.56788   12
2   1   0         7.65781   12
2   2   0         2.24588   12
2   2   1         2.22662   12
```

The first three records define the basis set for atoms of the first type (carbon in this example), and the following three, the basis set for atoms of the second type (oxygen).

NG is mandatory even in case of diatomics, though gaussian expansions will not be used, the value of NG being irrelevant in this case.

28.3 Example

Example using internal basis set for H_2O.

```
memory,20,m

if(.not.modul_slater) then
exit
end if

geomtyp=zmat
geometry={
o;h1,o,r;h2,o,r,h1,theta
}
r=0.96 ang
theta=102

intyp='SLATER'
basis=VB1

hf
ccsd(t)
multi
mrci
```

http://www.molpro.net/info/current/examples/slater.com

29 LOCAL CORRELATION TREATMENTS

29.1 Introduction

The local correlation program of MOLPRO can currently perform closed-shell LMP2, LMP3, LMP4(SDTQ), LCISD, LQCISD(T), and LCCSD(T) calculations. For large molecules, all methods scale linearly with molecular size, provided very distant pairs are neglected, and the integral-direct algorithms are used.

Much higher efficiency is achieved by using density fitting (DF) approximations to compute the integrals. Density fitting is available for all local methods up to LCCSD(T), as well as for analytical LMP2 gradients. Only iterative triples methods like LCCSDT-1b can currently not be done with density fitting.

The errors introduced by DF are negligible, and the use of the DF methods is highly recommended. Linear scaling can be obtained in DF-LMP2 using the `LOCFIT` option (see Ref. 11); in DF-LCCSD(T), the most important parts also scale linearly, but some transformation steps scale quadratically.

Energy gradients are available for LMP2, DF-LMP2, DF-SCS-LMP2, and LQCISD (in the latter case only for `LOCAL=1`, i.e. the local calculation is simulated using the canonical program, and savings only result from the reduced number of pairs).

Local explicitly correlated methods (DF-LMP2-R12 and DF-LMP2-F12 are described in section 31.

Before using these methods, it is strongly recommended to read the literature in order to understand the basic concepts and approximations. A recent review [1] and Ref. [2] may be suitable for an introduction.

References:

Review:
[1] H.-J. Werner and K. Pflüger, *On the selection of domains and orbital pairs in local correlation treatments*, Ann. Rev. Comp. Chem., in press. (preprint available under `http://www.theochem.uni-stu`

General local Coupled Cluster:
[2] C. Hampel and H.-J. Werner, *Local Treatment of electron correlation in coupled cluster (CCSD) theory*, J. Chem. Phys. **104**, 6286 (1996).
[3] M. Schütz and H.-J. Werner, *Local perturbative triples correction (T) with linear cost scaling*, Chem. Phys. Letters **318**, 370 (2000).
[4] M. Schütz, *Low-order scaling local electron correlation methods. III. Linear scaling local perturbative triples correction (T)*, J. Chem. Phys. **113**, 9986 (2000).
[5] M. Schütz and H.-J. Werner, *Low-order scaling local electron correlation methods. IV. Linear scaling local coupled-cluster (LCCSD)*, J. Chem. Phys. **114**, 661 (2001).
[6] M. Schütz, *Low-order scaling local electron correlation methods. V. Connected Triples beyond (T): Linear scaling local CCSDT-1b*, J. Chem. Phys. **116**, 8772 (2002).
[7] M. Schütz, *A new, fast, semi-direct implementation of Linear Scaling Local Coupled Cluster Theory*, Phys. Chem. Chem. Phys. **4**, 3941 (2002).

Multipole treatment of distant pairs:
[8] G. Hetzer, P. Pulay, H.-J. Werner, *Multipole approximation of distant pair energies in local MP2 calculations*, Chem. Phys. Lett. **290**, 143 (1998).

Linear scaling local MP2:
[9] M. Schütz, G. Hetzer and H.-J. Werner, *Low-order scaling local electron correlation methods. I. Linear scaling local MP2*, J. Chem. Phys. **111**, 5691 (1999).

[10] G. Hetzer, M. Schütz, H. Stoll, and H.-J. Werner, *Low-order scaling local electron correlation methods II: Splitting the Coulomb operator in linear scaling local MP2*, J. Chem. Phys. **113**, 9443 (2000).

Density fitted local methods:
[11] H.-J. Werner, F. R. Manby, and P. J. Knowles, *Fast linear scaling second-order Møller-Plesset perturbation theory (MP2) using local and density fitting approximations*, J. Chem. Phys. **118**, 8149 (2003). [12] M. Schütz and F.R. Manby, *Linear scaling local coupled cluster theory with density fitting. Part I: 4- external integrals*, Phys. Chem. Chem. Phys. **5**, 3349 (2003).
[13] Polly, H.-J. Werner, F. R. Manby, and Peter J. Knowles, *Fast Hartree-Fock theory using local density fitting approximations*, Mol. Phys. **102**, 2311 (2004).
[14] H.-J. Werner and M. Schütz, *Low-order scaling coupled cluster methods (LCCSD(T)) with local density fitting approximations*, in preparation.

LMP2 Gradients and geometry optimization:
[15] A. El Azhary, G. Rauhut, P. Pulay and H.-J. Werner, *Analytical energy gradients for local second-order Møller-Plesset perturbation theory*, J. Chem. Phys. **108**, 5185 (1998).
[16] G. Rauhut and H.-J. Werner, *Analytical Energy Gradients for Local Coupled-Cluster Methods*, Phys. Chem. Chem. Phys. **3**, 4853 (2001).
[17] M. Schütz, H.-J. Werner, R. Lindh and F.R. Manby, *Analytical energy gradients for local second-order Møller-Plesset perturbation theory using density fitting approximations*, J. Chem. Phys. **121**, 737 (2004).

LMP2 vibrational frequencies:
[18] G. Rauhut, A. El Azhary, F. Eckert, U. Schumann and H.-J. Werner, *Impact of Local Approximations on MP2 Vibrational Frequencies*, Spectrochimica Acta **55**, 651 (1999).
[19] G. Rauhut and H.-J. Werner *The vibrational spectra of furoxan and dichlorofuroxan: a comparative theoretical study using density functional theory and Local Electron Correlation Methods*, Phys. Chem. Chem. Phys. **5**, 2001 (2003).
[20] T. Hrenar, G. Rauhut and H.-J. Werner, *Impact of local and density fitting approximations on harmonic vibrational frequencies*, J. Phys. Chem. A., **110**, 2060 (2006).

Intermolecular interactions and the BSSE problem:
[21] M. Schütz, G. Rauhut and H.-J. Werner, *Local Treatment of Electron Correlation in Molecular Clusters: Structures and Stabilities of* $(H_2O)_n$, $n = 2-4$, J. Phys. Chem. **102**, 5997 (1998). See also [2] and references therein.
[22] N. Runeberg, M. Schütz and H.-J. Werner, *The aurophilic attraction as interpreted by local correlation methods*, J. Chem. Phys. **110**, 7210 (1999).
[23] L. Magnko, M. Schweizer, G. Rauhut, M. Schütz, H. Stoll, and H.-J. Werner, *A Comparison of the metallophilic attraction in* $(X\text{-}M\text{-}PH_3)_2$ *(M=Cu, Ag, Au; X=H, Cl)*, Phys. Chem. Chem. Phys. **4**, 1006 (2002).

29.2 Getting started

The local correlation treatment is switched on by preceding the command name by an `L`, i.e., by using the `LMP2`, `LMP3`, `LMP4`, `LQCISD`, `LCCSD`, or `LCISD` commands.

The `LQCISD` and `LCCSD` commands can be appended by a specification for the perturbative treatment of triple excitations (e.g., `LCCSD(T0)`):

- `(T)` Use the default triples method. Currently this is T0.
- `(T0)` Non-iterative local triples. This is the fastest triples option. It is usually sufficiently accurate and recommended to be used in most cases.

(T1) T0 plus one perturbative update of the triples amplitudes. If the accuracy of T0 is insufficient (very rarely the case!), this can be used to improve the accuracy. The computational cost is at least twice as large as for T0. In contrast to T0, the triples amplitudes must be stored on disk, which can be a bottleneck in calculations for large molecules. Also the memory requirements are substantially larger than for T0.

(T1C) As T1, but a caching algorithm is used which avoids the simultaneous storage of all triples amplitudes on disk (as is the case for `(T1)` or `(TF)`). Hence, T1C requires less disk space but more CPU-time than T1. The more disk space is made available for the caching algorithm (using the T1DISK option on the local card, see below), the less CPU time is used.

(TF) Full iterative triples calculation. With full domains and without weak pair approximations this gives the same result as a canonical (T) calculation. Typically, 3-5 iterations are needed, and therefore the computational effort is 2-3 times larger than for (T1). The disk and memory requirements are the same as for T1. The T0 energy is also computed and printed. `TFULL` and `FULL` are aliases for `TF`.

(TA) As TF, but the T1 energy is also computed. Since the first iteration is different for T1, the convergence of the triples iterations is slightly different with TF and TA (TF being somewhat faster in most cases). `TALL` and `ALL` are aliases for `TA`.

Density fitting can be invoked by prepending the command name by `DF-`, e.g. `DF-LMP2`, `DF-LCCSD(T0)` etc. In density fitting calculations an additional auxiliary basis set is needed. Details about choosing such basis sets and other options for density fitting are described in sections 29.10 and 15.

The general input for local coupled LMP2 or coupled cluster calculations is:

`LMP2`,*options*	Local MP2 calculation
`LCCSD`,*options*	Local CCSD calculation
`LCCSD(T0)`,*options*	Local CCSD(T0) calculation

The same options as on the command line can also be given on subsequent `LOCAL` and `MULTP` directives. Instead of using the `MULTP` directive, the `MULTP` option on the command line can also be used.

In the following, we will first give a summary of all options and directives. These will be described in more detail in the subsequent sections. For new users it is recommended to read section 29.9 at the end of this chapter before starting calculations.

29.3 Summary of options

Many options can be specified on the command line. For all options appropriate default values are set, and so these options must usually be modified only for special purposes. For convenience and historical reasons, alias names are available for various options, which often correspond to the variable name used in the program. Table 14 summarizes the options, aliases and default values. In the following, the parameters will be described in more detail.

Table 14: Summary of `local` (*multp*) options and their default values

Parameter	Alias	Default value	Meaning
General Parameters:			
LOCAL		4	determines which program to use.
MULTP		0	turns on multipole approximations for distant pairs.
SAVEDOM	SAVE	0	specifies record for saving domain info.
RESTDOM	START	0	specifies record for reading domain info.
LOCORB		0	activates or deactivates orbital localization.
LOC_METHOD			specifies which localization method to use.
CANONICAL		0	allows to use canonical virtual orbitals (for testing).
PMDEL	CPLDEL	0	discards contributions of diffuse functions in PM localization.
SAVORB	SAVLOC	0	specifies record for saving local orbitals.
DOMONLY		0	if 1, only domains are made. if 2, only orbital domains are made.
Parameters to define domains:			
THRBP	DOMSEL	0.98	Boughton-Pulay selection criterion for orbital domains.
NPASEL		0	charge used in the NPA selection criterion for orbital domains.
CHGMIN		0.01	determines the minimum allowed atomic charge in domains.
CHGMINH		0.03	as CHGMIN, but used for H-atoms (default 0.03).
CHGMAX		0.40	If the atomic charge is larger than this value, the atom is always included in the domain.
MAXANG	MAXL	99	angular momentum restriction for BP domain selection
MAXBP		0	determines how atoms are ranked in BP procedure.
MULLIKEN	LOCMUL	0	determines the method to determine atomic charges.
MERGEDOM		0	merges overlapping domains.
DELCOR	IDLCOR	2	delete projected core AOs up to certain shell.
DELBAS	IBASO	0	determines how to remove redundancies.
Distance criteria for domain extensions:			
REXT		0	criterion for all pair domains.
REXTS		0	criterion for strong pair domains.
REXTC		0	criterion for strong and close pair domains.
REXTW		0	criterion for strong, close, and weak pair domains.
Connectivity criteria for domain extensions:			
IEXT		0	criterion for all pair domains.
IEXTS		0	criterion for strong pair domains.
IEXTC		0	criterion for strong and close pair domains.
IEXTW		0	criterion for strong, close, and weak pair domains.
Parameters to select pair classes:			
USE_DIST		1	determines if distance of connectivity criteria are used.
RCLOSE	CLOSEP	1	distance criterion for selection of weak pairs.
RWEAK	WEAKP	3	distance criterion for selection of weak pairs.
RDIST	DISTP	8	distance criterion for selection of distant pairs.
RVDIST	VERYD	15	distance criterion for selection of very distant pairs.
ICLOSE		1	connectivity criterion for selection of weak pairs.
IWEAK		2	connectivity criterion for selection of weak pairs.
IDIST		5	connectivity criterion for selection of distant pairs.
IVDIST		8	connectivity criterion for selection of very distant pairs.
CHGMIN_PAIRS	CHGMINP	0.20	determines minimum charge of atoms used for pair classification.
KEEPCL		0	determines if close pairs are included in LCCSD.

Parameter	Alias	Default value	Meaning
Parameter for multipole treatment of exchange operators:			
DSTMLT		3	multipole expansion level for distant pairs
Parameters for energy partitioning:			
IEPART		0	If nonzero: do energy partitioning.
EPART		3.0	cutoff parameter for determining individual monomers.
Parameters for redundancy check using DELBAS=1 (not recommended)			
TYPECHECK	TYPECHK	1	activates basis function type restrictions.
DELSHL	IDLSHL	1	determines if whole shells are to be deleted.
DELEIG	IDLEIG	1	determines how to select redundant functions.
DELCMIN	CDELMIN	0.1	parameter for use with DELEIG=1
Parameters for choosing operator domains in LCCSD			
OPDOM	IOPDOM	5	determines how operator domains are determined for LCCSD
RMAXJ		8	distance criterion for J-operator list.
RMAXK		8	distance criterion for K-operator list.
RMAXL		15	distance criterion for L-operator list.
RMAX3X		5	distance criterion for 3-ext integral list.
RDOMJ		0	distance criterion for K-operator domains.
RDOMK		8	distance criterion for J-operator domains.
IMAXJ		5	connectivity criterion for J-operator list.
IMAXK		5	connectivity criterion for K-operator list.
IMAXL		8	connectivity criterion for L-operator list.
IMAX3X		3	connectivity criterion for 3-ext integral list.
IDOMJ		0	connectivity criterion for K-operator domains.
IDOMK		5	connectivity criterion for J-operator domains.
Miscellaneous options:			
SKIPDIST	SKIPD	3	determines at which stage weak and distant pairs are eliminated
ASYDOM	JITERM	0	parameter for use of asymmetric domains
LOCSING	LOCSNG	0	determines virtual space used for singles
PIPEKAO	LOCAO	0	activates AO localization criterion
NONORM		2	determines whether projected functions are normalized
LMP2ALGO	MP2ALGO	3	if nonzero, use low-order scaling method in LMP22 iterations
OLDDEF		0	allows to revert to older defaults
T1DISK		10	maximum disk space (in GByte) for T1 caching algorithm
Thresholds:			
THRBP		0.98	Threshold Boughton-Pulay method.
THRPIP		1.d-12	Threshold for Pipek-Mezey localization.
THRORB		1.d-6	Threshold for eliminating projected orbitals with small norm.
THRLOC		1.d-6	Threshold for eliminating redundant projected orbitals.
THRCOR		1.d-1	Threshold for eliminating projected core orbitals.
THRMP2		1.d-8	Threshold for neglecting small fock matrix elements in the LMP2 iteration.

29.4 Summary of directives

The same standard directives as in the canonical programs, e.g., OCC, CLOSED, CORE, WF, ORBITAL are also valid in the local methods. In addition, there are some directives which only apply to local calculations:

LOCAL	Invokes local methods and allows to specify the same options as on the command line.
MULTP	As LOCAL, but multipole approximations are used for distant pairs.
DOMAIN	Define domains manually (not recommended).
MERGEDOM	Allows to merge domains
REGION	Allows to select regions of a molecule to be treated at a certain level of theory.
ENEPART	Analysis of pair energies.
SAVE	Save domains and LCCSD amplitudes.
START	Restart with domains and LCCSD amplitudes from a previous calculation.

29.5 General Options

LOCAL=*local* — Determines which method is used:
LOCAL=0: Conventional (non-local) calculation.
LOCAL=1: Local method is simulated using canonical MOs. The local basis is used only at an intermediate stage to update the amplitudes in each iteration (only for testing).
LOCAL=2: Calculation is done in local basis, but without using local blocking (i.e. full matrices are used). This is the most expensive method and only for testing.
LOCAL=3: Fully local calculation (obsolete).
LOCAL=4: Fully local calculation (default). This is the fastest method for large molecules with many weak pairs and requires minimum memory.

LOCORB=*option* — If this option is given and *option*$>$ 0, the orbitals are localized using the Pipek-Mezey technique. If this option is not given or *option*=0 (default), the orbitals are localized unless localized orbitals are found in the orbital record (cf. ORBITAL directive and LOCALI command). In the latter case, the most recent localized orbitals are used. Setting *option*=-1 switches the localization off. If *option*$>$ 1 the localized orbitals are printed. Note: Boys localization can only be performed using the LOCALI command. The program will use the Boys orbitals if they are found in the orbital record and the LOCORB option is absent or *option*$\leq$ 0.

LOC_METHOD=*method* — This option allows to select between Pipek-Mezey (*method*=PM) or Natural Orbitals (*method*=NBO) localization. If Pipek-Mezey orbitals are requested, the default Boughton-Pulay domain selection will be used. When *method*=NBO, the domain selection will be based on the NPA charges, with NPASEL=0.03 (by default). In both cases, the domain selection parameter can be explicitly given (cf. DOMSEL and

NPASEL options) in order to use another domain criterion. If localized orbitals are found in the orbital record, but the type does not coincide, the orbitals are again localized according to *method* and used in the calculation.

SAVORB=*record* Allows the localized and projected orbitals to be saved in *record=name.ifil* for later use (e.g. plotting). The two orbital sets are stored in the same dump record and can be restored at later stages using ORBITAL,*record*,[TYPE=]LOCAL or ORBITAL,*record*,[TYPE=]PROJECTED, respectively.

DOMONLY=*value* If *value*> 0 only domains are made, but no energy is computed. This can be used to check and save the domains for later use.

DSTMLT=*level* Determines the expansion level of the multipole expansion of distant pairs (e.g. 1 means dipole approximation, 2 quadrupole approximation and so on). The default for MULTP is 3.

INTERACT Automatically determine individual molecules in a calculation and set appropriate pair lists for computing interaction energies. See section 29.9.8 for more details.

Parameters for energy partitioning:

IEPART=*value* enables/disables energy partitioning.
iepart=0: Energy partitioning is disabled.
iepart=1: Energy partitioning is enabled.
iepart=2: Energy partitioning is enabled. Additionally, a list of all pair energies and their components is printed.

EPART=*cutoff* Cutoff parameter to determine individual monomers in a cluster (i.e. centre groups). Should be somewhat larger than the largest intramolecular bond length (given in a.u.).

Miscellaneous options:

SKIPDIST=*skipdist* Test-parameter. Its value should only affect the efficiency but not influence the results.
skipdist=-1: Weak and distant pairs are set to zero after MP2 but are not eliminated from the pair list and not skipped in any loop.
skipdist=0: No pairs are deleted from pair list, but weak and distant pairs are skipped in the loops were appropriate.
skipdist=1: Very distant pairs are neglected from the beginning. Distant pairs are eliminated after MP2.
skipdist=2: As skipdist=1, but also weak pairs are eliminated after MP2.
skipdist=3: As skipdist=2, but distant pairs are eliminated from the operator list in case of LMP2 with multipole approximations for distant pairs. This is the default.

ASYDOM=*jiterm* Experimental test parameter. Enables the use of asymmetric domains for distant pairs. The asymmetric domain approximation supplements the multipole approximation for distant pairs, as it suppresses the treatment of configurations for which no integrals can be computed by multipole expansion. This leads to computational savings and improved numerical stability.
jiterm=0: Disable asymmetric domains.
jiterm=-1: Enable asymmetric domains (default).

jiterm=-2: Enable a variation of the asymmetric domain formalism: Exchange operators will initially be projected to the asymmetric domain instead of simply packed.

LOCSING=*locsing* — If *locsing.ne.0*, the single excitations use the full space, i.e., they are not treated locally. This is only works for LOCAL=1.

MAXANG=*lmax* — The purpose of this experimental option is to reduce the basis set sensitivity of the Boughton-Pulay (BP) method for domain selection. Only basis functions with angular momentum up to *lmax-1* are included when computing the overlap of the approximate and exact orbitals. For example, MAXANG=2 means to omit all contributions of d, f and higher angular momentum functions. To obtain reasonable domains, the value of THRBP must often be reduced (to 0.97 or so). This option should only be used with care!

PIPEKAO=*option* — If *option*≥ 0, the orbitals are localized my maximizing the coefficients of basis functions of a given type at a given atom. Normally, this is only useful to uniquely define degenerate orbitals in atoms. For instance, when this option is used to localize the orbitals for a dimer like $(Ar)_2$ at a very long distance, clean s, p_x, p_y, and p_z atomic orbitals will be obtained. It is not recommended to use this option for molecular calculations!

NONORM=*value* — Determines if projected functions are normalized (not recommended).
value=-1: projected orbitals are normalized before redundancy check.
value=0: projected orbitals are normalized after redundancy check (default).
value=1: projected orbitals are normalized in redundancy check, afterwards unnormalized.
value=2: projected orbitals are never normalized (default in gradient calculations).

LMP2ALGO=*value* — If nonzero, use low-order scaling method in LMP2 iterations. Values can be 1, 2, or 3, and 3 is usually fastest if large basis sets are used.

OLDDEF=*value* — For compatibility with older versions: if nonzero, revert to old defaults. Options set before this may be overwritten.

Thresholds:

THRPIP=*thresh* — Threshold for Pipek-Mezey localization. The localization is assumed to be converged if all 2×2 rotation angles are smaller then *thresh*. The default is $1.d-12$. It can also be modified globally using GTHRESH, LOCALI=*thresh*.

THRORB=*thresh* — Threshold for eliminating functions from pair domains whose norm is smaller then *thresh* after projecting out the occupied space. The default is *throrb=1.d-6*.

THRLOC=*thresh* — Threshold for eliminating redundant basis functions from pair domains. For each eigenvalue of $\tilde{\mathbf{S}}^{ij} < thresh$ one function is deleted. The default is 1.d-6. The method used for deleting functions depends on the parameters IDLEIG and IBASO.

THRMP2=*thresh* — Threshold for neglecting small fock matrix couplings in the LMP2 iterations (default 1.d-8). Specifying a larger threshold speeds up the iterations but may lead to small errors in the energy. In the initial iterations, a larger threshold is chosen automatically. It is gradually reduced to the specified final value during the iterations.

`THRCOR`=*thresh* Threshold for deleting projected core orbitals. The functions are only deleted if their norm is smaller than *thresh* (default 0.1)

The thresholds can also be specified on the `THRESH` directive.

29.6 Options for selection of domains

The following sections describe the most important options which affect the domains.

29.6.1 Standard domains

Standard domains are always determined first. They are used to define strong, close, weak, and distant pairs. More accurate results can be obtained with extended domains, as described in section 29.6.2.

`THRBP`=*value* Threshold for selecting the atoms contributing to orbital domains using the method of Boughton and Pulay (BP). As many atoms as needed to fulfill the BP criterion are included in a domain. The order in which atoms are considered depends on the parameter `MAXBP`, see below. The default is `THRBP=0.98`. `THRBP=1.0` includes all atoms into each orbital domain, i.e., leads to full domains. If no pairs are neglected, this should yield the canonical MP2 energy.

The criterion is somewhat basis dependent. See section 29.9.4 for recommended values of this threshold.

`CHGMIN`=*value* determines the minimum allowed Mulliken (or Löwdin) charge for an atom (except H) in a domain, i.e., atoms with a smaller (absolute) charge are not included, even if the `THRBP` criterion is not fulfilled (default 0.01).

`CHGMINH`=*value* as `CHGMIN`, but used for H-atoms (default 0.03).

`CHGMAX`=*value* If the atomic charge is larger than this value, the atom is included, independent of any ranking.

`MAXBP`=*maxbp* If *maxbp=1*, the atoms are ranked according to their contribution to the Boughton-Pulay overlap. If *maxbp=0* (default), the atoms are ranked according to atomic charges. In both cases atoms with charges greater than `CHGMAX` are always included, and atoms with the same charges are added as groups.

`MULLIKEN`=*option* Determines the method to determine atomic charges. `MULLIKEN=0` (default): squares of diagonal elements of $\mathbf{S}^{\frac{1}{2}}\mathbf{C}$ are used (Löwdin charges); `MULLIKEN=1`: Mulliken gross charges are used. The first choice is less basis set dependent and more reliable with diffuse basis sets.

`MERGEDOM`=*number* If *number* is greater than zero, all orbital domains containing *number* or more atoms in common are merged (*number=1* is treated as *number=2*, default 0). This is particularly useful for geometry optimizations of conjugated or aromatic systems like, e.g., benzene. In the latter case, `MERGEDOM=1` causes the generation of full π-domains, i.e., the domains for all three π-orbitals comprise all carbon basis functions. Note that the merged domains are generated after the above print of orbital domains, and information about merged domains is printed separately. See section 29.9.7 for further discussion of geometry optimizations.

There are some other options which should normally not be modified:

`DELBAS`=*ibaso* This parameter determines the method for eliminating redundant functions of pair domains.
ibaso=0: The space of normalized eigenvectors of $\tilde{\mathbf{S}}^{ij}$, which correspond to small eigenvalues, is eliminated (default). Any other value is not recommended and not further documented.

`DELCOR`=*nshell* Activates elimination of basis functions corresponding to core orbitals. If *nshell=1*, only $1s$-functions are eliminated from projected space. If *nshell=2* (default) $1s$ functions on first-row atoms, and $1s$, $2s$, and $2p$-functions are eliminated on second-row atoms. Nothing is eliminated on H or He atoms. If effective core potentials are used, nothing is deleted at the corresponding atom. Also, functions are only deleted if the norm of the projected function is below `THRCOR` (default 0.1)

29.6.2 Extended domains

There are two alternative modes for domain extensions: either *distance criteria* `REXT`, `REXTS`, `REXTC`, or `REXTW` can be used. These are in Bohr and refer to the minimum distance between any atom in a standard orbital domain $[ij]$ and another atom. If an atom is found within the given distance, all PAOs at this atom are added to the domain $[ij]$. Alternatively, *connectivity criteria* `IEXT`, `IEXTS`, `IEXTC`, or `IEXTW` can be used. These refer to the number of bonds between any atom contained in the standard domain $[ij]$ and another atom. The advantage of distance criteria is that they select also atoms within the given radius which are not connected to the present domain by bonds. On the other hand, the connectivity criteria are independent of different bond lengths, e.g., for first and second-row atoms. Only one of the two possibilities can be used, i.e., they are mutually exclusive.

`REXT`=*value* Distance criterion for extension of all pair domains.

`REXTS`=*value* Distance criterion for extension of strong pair domains.

`REXTC`=*value* Distance criterion for extension of strong and close pair domains.

`REXTW`=*value* Distance criterion for extension of strong, close, and weak pair domains.

`IEXT`=*value* Connectivity criterion for extension of all pair domains.

`IEXTS`=*value* Connectivity criterion for extension of strong pair domains.

`IEXTC`=*value* Connectivity criterion for extension of strong and close pair domains.

`IEXTW`=*value* Connectivity criterion for extension of strong, close, and weak pair domains.

By default, domains are not extended, i.e., the default values of all parameters listed above are zero. Note that the pair classes are determined on the basis of the standard domains, and therefore domain extensions have no effect on the pair lists.

Also note that the computational effort increases with the fourth power of the domain sizes and can therefore increase quite dramatically when extending domains. This does not affect the linear scaling behaviour in the asymptotic limit.

29.6.3 Manually Defining orbital domains (DOMAIN)

It is possible to define the domains "by hand", using the DOMAIN directive:

DOMAIN,*orbital*,*atom1*, *atom2* ...

where *orbital* has the form iorb.isym, e.g., 3.1 for the third orbital in symmetry 1, and *atomi* are the atomic labels as given in the Z-matrix geometry input, or, alternatively, the Z-matrix row numbers. All basis functions centred at the given atoms are included into the domain. For instance

DOMAIN,3.1,C1,C2

defines a domain for a bicentric bond between the carbon atoms C1 and C2. The DOMAIN directive must be given after any OCC, CLOSED, or CORE directives. Note that the order of the localized orbitals depends on the localization procedure, and could even change as function of geometry, and therefore manual DOMAIN input should be used with great care. The domains of all orbitals which are not explicitly defined using DOMAIN directive are determined automatically as usual.

29.7 Options for selection of pair classes

There are two alternative modes for defining the pair classes: either *distance criteria* RCLOSE, RWEAK, RDIST, RVDIST can be used. These are in Bohr and refer for a given orbital pair (ij) to the minimum distance $R^{(ij)}$ between any atom in the standard orbital domains $[i]$ and any atom in the standard orbital domains $[j]$. Alternatively, the *connectivity criteria* ICLOSE, IWEAK, IDIST, IVDIST can be used. These refer to the minimum number of bonds between any atom contained in the standard domain $[i]$ and any atom contained in the standard domain $[j]$ The advantage of using connectivity criteria is the independence of the bond lengths, while the advantage of distance criteria (default) is that they are also effective in non-bonding situations. Only one of the two possibilities can be used, i.e., they are mutually exclusive. The use of distance criteria is the default. Using connectivity criteria for pair selection requires to set the option USE_DIST=0.

USE_DIST	(default 1) If nonzero, use distance criteria, otherwise connectivity criteria.
CHGMIN_PAIRS	Only atoms in the primary domains are considered for the pair classification if the atomic Löwdin charge is larger than CHGMIN_PAIRS (default value 0.2). This criterion was introduced in order to reduce the dependence of the pair selection on localization tails.
RCLOSE	(default 1) Strong pairs are defined by $0 \leq R^{(ij)} <$ RCLOSE. Close pairs are defined by RCLOSE $\leq R^{(ij)} <$ RWEAK.
RWEAK	(default 3) Weak pairs are defined by RWEAK $\leq R^{(ij)} <$ RDIST.
RDIST	(default 8) Distant pairs are defined by RDIST $\leq R^{(ij)} <$ RVDIST.
RVDIST	(default 15) Very distant pairs for which $R^{(ij)} \geq$RVDIST are neglected.

ICLOSE	(default 1) Strong pairs are separated by less that ICLOSE bonds. Close orbital pairs are separated by at least ICLOSE bonds but less than IWEAK bonds.
IWEAK	(default 2) Weak orbital pairs are separated by at least IWEAK bonds but less than IDIST bonds.
IDIST	(default 5) Distant orbital pairs are separated by at least IDIST bonds but less than IVDIST bonds.
IVDIST	(default 8) Very distant orbital pairs (neglected) are separated by at least IVDIST bonds.
KEEPCL	(default 0) If KEEPCL=1, the LMP2 amplitudes of close pairs are included in the computation of the strong pair LCCSD residuals. If KEEPCL=2 all close pairs are fully included in the LCCSD (this does not affect the triples list). This option is not yet implemented as efficiently as it could, and can therefore lead to a significant increase of the CPU time.

Setting RCLOSE or RWEAK to zero means that all pairs up to the corresponding class are treated as strong pairs (RWEAK=0 implies RCLOSE=0). For instance, RCLOSE=0 means that strong and close pairs are fully included in the LCCSD (in this case KEEPCL=1 has no effect). Note, however, that setting RCLOSE=0 increases the length of the triples list. Setting RDIST=0 means that all distant pairs are treated as weak pairs. This does not affect RWEAK and RCLOSE and has no effect unless multipole approximations are used for distant pairs. Setting RVDIST=0 means that no very distant pairs are neglected. Again, this has no effect on the other distance parameters.

29.8 Directives

29.8.1 The LOCAL directive

The LOCAL directive can be used to specify options for local calculations. If this directive is inside the command block of a local calculation, the options are used only for the current calculation, and this is entirely equivalent as if they were specified on the command line. The LOCAL directive can also be given outside a command block, and in this case the options are used for all subsequent local correlation calculations in the same input.

Example:

```
DF-LMP2,THRBP=0.985
```

is equivalent to

```
{DF-LMP2
LOCAL,THRBP=0.985}
```

In the following example the LOCAL directive is global and acts on all subsequent local calculations, i.e. both calculations will use THRBP=0.985

```
LOCAL,THRBP=0.985
DF-LMP2       !local MP2 calculation
OPTG          !geometry optimization using the DF-LMP2 energy
DF-LCCSD(T)   !local coupled cluster at the optimized structure.
```

29.8.2 The MULTP directive

The MULTP directive turns on the multipole approximations for distant pairs, as described in Ref. [8]. Further options can be given as described above for the LOCAL directive.

LOCAL,MULTP,*options*

is equivalent to

MULTP,*options*

The level of the multipole approximation can be chosen using option DSTMLT (default 3) (1 means dipole approximation, 2 quadrupole approximation and so on).

The multipole approximation reduces the computational cost of LMP2 calculations for very large molecules, but leads to some additional errors, see Ref. [8]. It is normally not recommended to be used in coupled-cluster calculations and should never be used for computing intermolecular forces. It can also not be used in geometry optimizations or gradient calculations.

29.8.3 Saving the wavefunction (SAVE)

The wavefunction can be saved for later restart using

SAVE,*record*

where *record* has the usual form, e.g., 4000.2 means record 4000 on file 2. If this directive is given, the domain information as well as the amplitudes are saved (for MPn the amplitudes are not saved). If just the domain information should be stored, the SAVE option on the LOCAL directive must be used (cf. section 29.3).

29.8.4 Restarting a calculation (START)

Local CCSD or QCISD calculations can be restarted using

START,*record*

The record given must have been saved in a previous local calculation using the SAVE directive (otherwise this directive is ignored). If the START directive is given, the domain information as well as the amplitudes of the previous calculation are used for restart. It is possible, for instance, to start a local CCSD calculation with the amplitudes previously saved for a local QCISD calculation (but of course it is not possible to use a record saved for a non-local CCSD or QCISD calculation). If it is intended only to use the domain information but not the amplitudes for a restart, the START option on the command line or LOCAL directive must be used (cf. section 29.3).

29.8.5 Correlating subsets of electrons (REGION)

In large molecules, it may be sufficient to correlate only the electrons in the vicinity of an *active group*, and to treat the rest of the molecule at the SCF level. This approach can even be extended, different correlation levels may be used for different sections of the system. The REGION directive allows the specification of a subset of atoms:

REGION,METHOD=*method*,[DEFAULT=*default_method*], [TYPE=INCLUSIVE|EXCLUSIVE], *atom1*, *atom2* ...

The orbitals located at these atoms will be treated at the level specified in *method*. The remaining orbitals will be treated as defined in *default*. If not given by the user, the latter option will be set to HF.

The orbital selection can be done in two ways. If *type* is set to INCLUSIVE, any orbital containing one of the atoms in its domain centre list will be included. If type is set to EXCLUSIVE, the program will only add orbitals whose domains are exclusively covered by the given atoms. Any local correlation treatment can be given as *method*, with the restriction that only MP2 and HF can be used as *default_method*. Up to two REGION directives may be included in a single calculation, ordered according to the correlation level (*method*) specified for the region. The highest level region should be given last.

It is advisable to check the region orbital list and the orbital domains printed by the program. The use of regions may significantly reduce the computation time, and, provided the active atoms are sensibly chosen, may give still sufficiently accurate results for the *active group*, e.g. bond lengths and bond angles.

29.8.6 Domain Merging (MERGEDOM)

The restriction of the virtual space in local calculations may result in discontinuities for reaction path calculations due to changes of the geometry dependent domains. This may be avoided by the use of a MERGEDOM directive

MERGEDOM,[NEIGHBOUR=*value*],[CENTERS=[*atom1, atom2*...]], [RECORD=...],CHECK

This directive provides augmented domains, which can be saved (using option or directive SAVE, see section 29.8.3) for later use in reaction paths or in single point calculations (in cases where the orbital domain description is unbalanced). The use of the *neighbour* option works in the same way as the local option MERGEDOM, with *value* specifying the number of coincident centres. If the *centres* option is used, an atom list should be given (enclosed by square brackets). The domains of all orbitals located exclusively at these atoms will be merged, and the resulting merged domains will be used for all these orbitals.

One may also give a *record* number from a previously saved local calculation. The domain list contained in the record will be matched to the current one, and orbital domains augmented (merged) to include both sets. This domain definition should then be adequate for calculations on both points (and all those in between). This procedure can be repeated to include more geometries. In this way domains can be defined that are appropriate for a whole range of geometries (e.g. a reaction path), and if these domains are used in all calculations a strictly smooth potential energy surface is obtained.

29.8.7 Energy partitioning for molecular cluster calculations (ENEPART)

The local character of occupied and virtual orbitals in the local correlation treatment also offers the appealing possibility to decompose the intermolecular interaction energy of molecular clusters into individual contributions of different excitation classes. This allows to distinguish between intramolecular-, dispersive-, and ionic components of the correlation contribution to the interaction energy (cf. M. Schütz, G. Rauhut and H.J. Werner, J. Phys. Chem. **102**, 5197 (1998)). The energy partitioning algorithm is activated either by supplying the ENEPART directive:

ENEPART,[*epart*],[*iepart*]

or by giving the parameters as options on the command line.

The *epart* parameter determines the cutoff distance for (intramolecular) bond lengths (in a.u., default 3 a.u.) and is used to automatically determine the individual monomer subunits of the cluster. The *iepart* parameter enables the energy partitioning, if set to a value larger than zero (default 1). Additionally, if iepart is set to 2, a list of all intermolecular pair energies and their components is printed.

The output section produced by the energy partitioning algorithm will look similar to the following example:

```
energy partitioning enabled !
centre groups formed for cutoff [au] = 3.00
 1   :O1  H11 H12
 2   :O2  H21 H22
energy partitioning relative to centre groups:
intramolecular correlation:      -.43752663
exchange dispersion       :       .00000037
dispersion energy         :      -.00022425
ionic contributions       :      -.00007637
```

The centre groups correspond to the individual monomers determined for *epart=3*. In the present example, two water monomers were found. The correlation energy is partitioned into the four components shown above. The exchange dispersion, dispersion and ionic components reflect directly the related intermolecular components of the complex, while the intramolecular correlation contribution to the interaction energy has to be determined by a super-molecular calculation, i.e. by subtracting the (two) corresponding monomer correlation energies from the intramolecular correlation component of the complex given in the output.

Alternatively, the following form can be used:

`ENEPART,RMAX=[`*r1,r2,r3,...*`]`

and the program will then print the energy contributions of all pairs in the ranges between the given distances (in bohr, enclosed by square brackets, e.g., `enepart,rmax=[0,3,5,7,9,11]`). A second list in which the contributions are given as a function of the number of bonds between the pair domains will also be printed.

29.9 Doing it right

The local correlation methods in MOLPRO employ localized molecular orbitals (LMOs). Pipek-Mezey localization is recommended, but Boys localization is also possible. The virtual orbital space is spanned by non-orthogonal projected atomic orbitals (PAOs). The local character of this basis makes it possible to introduce two distinct approximations: first, excitations are restricted to *domains*, which are subspaces of (PAOs) that are spatially close to the orbitals from which the electrons are being excited. Secondly, the orbital pairs are classified according to their importance (based on distance or connectivity criteria), and only *strong pairs* are treated at the highest level (e.g. CCSD). The remaining *weak* and *distant* pairs are treated at the LMP2 level, and *very distant* pairs are neglected. These approximations lead to linear scaling of the computational resources as a function of the molecular size.

Naturally, such approximation can introduce some errors, and therefore the user has to be more careful than with standard black box methods. On the other hand, the low-order scaling makes it possible to treat much larger systems at high levels of theory than it was possible so far.

This section summarizes some important points to remember when performing local correlation calculations.

29.9.1 Basis sets

For numerical reasons, it is useful to eliminate projected core orbitals, since these may have a very small norm. By default, projected core orbitals are eliminated if their norm is smaller then 0.1 (this behaviour can be changed using the DELCOR and THRCOR options). For local calculations we recommend the use of generally contracted basis sets, e.g., the correlation consistent cc-pVnZ sets of Dunning and coworkers. For these basis sets the core basis functions are uniquely defined, and will always be eliminated if the defaults for DELCOR and THRCOR are used.

The correlation consistent basis sets are also recommended for all density fitting calculations, since optimized fitting basis sets are available for each basis.

29.9.2 Symmetry and Orientation

1. Turn off symmetry! Otherwise, you won't get appropriately localized orbitals (local orbitals will tend to be symmetry equivalent instead of symmetry adapted). Symmetry can be used only if all atoms are symmetry unique. This allows the local treatment of planar molecules in C_s symmetry. But note that neither the multipole program nor the density fitting programs support symmetry at all, so choose always C_1 symmetry for DF-calculations or with the MULTP option.

To turn off symmetry, specify NOSYM as the first line of your geometry input, e.g.

```
geometry={
  nosym
  O1
  H1,O1,roh
  H2,O1,roh,h1,hoh
}
```

Alternatively, add

```
SET,ZSYMEL=NOSYM
```

before the geometry block.

2. Use NOORIENT! We recommend to use the NOORIENT option in the geometry input, to avoid unintended rotations of the molecule when the geometry changes. This is particularly important for geometry optimizations and for domain restarts in calculations of interaction energies (see section 29.9.8).

29.9.3 Localization

By default, Pipek-Mezey localization is used and performed automatically in the beginning of a local correlation calculation. Thus

```
df-hf         !Hartree-Fock with density fitting
df-lmp2       !LMP2 using the Pipek-Mezey LMOs
```

is equivalent to

```
df-hf         !Hartree-Fock with density fitting
locali,pipek  !Orbital localization using the Pipek-Mezey criterion
df-lmp2       !LMP2 using the Pipek-Mezey LMOs
```

Boys localization can be used as well, but in this case the localization must be done beforehand, e.g.

```
df-hf         !Hartree-Fock with density fitting
locali,boys   !Orbital localization using the Boys criterion
df-lmp2       !LMP2 using the Boys LMOs
```

Poor localization is sometimes an intrinsic problem, in particular for strongly conjugated systems or when diffuse basis sets are used. This is caused by localization tails due to the overlapping diffuse functions. The problem is particularly frequent in calculations of systems with short bonds, e.g., aromatic molecules. It can be avoided using directive

`PIPEK,DELETE=`*n*

with $n = 1$ or 2. This means that the contributions of the n most diffuse basis functions of each angular momentum type are ignored in the localization. This often yields much better localized orbitals when diffuse basis sets are used. For aug-cc-pVTZ, $n = 2$ has been found to work very well, while for aug-cc-pVDZ n=1

In rare cases it might also happen that the localization procedure does not converge. It is them possible to choose a second-order Newton-Raphson localization scheme, using the directive

`PIPEK,METHOD=2,`[`DELETE=`*n*]

Alternatively (recommended) one can use

`PIPEK,METHOD=3,`[`DELETE=`*n*]

which first performs a few standard Pipek-Mezey iterations and the invokes the second-order localization scheme. This then usually converges very quickly.

29.9.4 Orbital domains

The orbital domains are determined automatically using the procedure of Boughton and Pulay, J. Comput. Chem., **14**, 736 (1993) and J. Chem. Phys. **104**, 6286 (1996). For higher accuracy the domains can be extended, and in this way the canonical result can be systematically approached (cf. Ref. [1] and section 29.6.2). Details are described in section 29.6.

In most cases, the domain selection is uncritical for saturated molecules. Nevertheless, in particular for delocalized systems, it is recommended always to check the orbital domains, which are printed in the beginning of each local calculation. For such checking, the option `DOMONLY=1` can be used to stop the calculation after the domain generation. The orbital domains consist of all basis functions for a subset of atoms. These atoms are selected so that the domain spans the corresponding localized orbital with a preset accuracy (alterable with option `THRBP`). A typical domain output, here for water, looks like this:

```
 Orbital domains

   Orb.   Atom    Charge       Crit.
   2.1    1 O1     1.17        0.00
```

```
        3 H2     0.84        1.00
3.1     1 O1     2.02        1.00
4.1     1 O1     1.96        1.00
5.1     1 O1     1.17        0.00
        2 H1     0.84        1.00
```

This tells you that the domains for orbitals 2.1 and 5.1 comprise the basis functions of the oxygen atom and and one hydrogen atom, while the domains for orbitals 3.1 and 4.1 consist of the basis function on oxygen only. The latter ones correspond to the oxygen lone pairs, the former to the two OH bonds, and so this is exactly what one would expect. For each domain of AOs, corresponding projected atomic orbitals (PAOs) are generated, which span subspaces of the virtual space and into which excitations are made. Options which affect the domain selection are described in section 29.6. Improper domains can result from poorly localized orbitals (see section 29.9.3 or a forgotten `NOSYM` directive. This does not only negatively affect performance and memory requirements, but can also lead to unexpected results.

The default for the selection criterion `THRBP` is 0.98. This works usually well for small basis sets like cc-pVDZ. For larger basis sets like cc-pVTZ we recommend to use a slightly larger value of 0.985 to ensure that enough atoms are included in each domain. For cc-pVQZ recommend `THRBP=0.990` is recommended. In cases of doubt, compare the domains you get with a smaller basis (e.g., cc-pVDZ).

The choice of domains usually has only a weak effect on near-equilibrium properties like equilibrium geometries and harmonic vibrational frequencies. More critical are energy differences like reaction energies or barrier heights. In cases where the electronic structure strongly changes, e.g., when the number of double bonds changes, it is recommended to compare DF-LMP2 and DF-MP2 results before performing expensive LCCSD(T) calculations. More balanced results and smooth potentials can be obtained using the `MERGEDOM` directive, see section 29.8.6.

29.9.5 Freezing domains

In order to obtain smooth potential energy surfaces, domains must be frozen. The domain information can be stored using the `SAVE` option and recovered using the `START` option. Alternatively, the `SAVE` and `START` can be used, see section 29.8.3. In the latter case, also the CCSD amplitudes are saved/restarted. Freezing domains is particularly important in calculations of intermolecular interactions, see section 29.9.8. Domains that are appropriate for larger ranges of geometries, such as reaction pathways, can be generated using the `MERGEDOM` directive, section 29.8.6. The domains are automatically frozen in geometry optimizations and frequency calculations, see section 29.9.7. Note, however, that this automatic procedure only works if a single local calculation is involved in the optimization. In case of optimizations with counterpoise correction the domains for the complex and each monomer must be frozen individually in different records using the `SAVE` and `START` directives.

29.9.6 Pair Classes

The *strong*, *close*, *weak* and *distant* pairs are selected using distance or connectivity criteria as described in more detail in section 29.7. *Strong* pairs are treated by CCSD, all other pairs by LMP2. In triples calculations, all orbital triples (ijk) are included for which (ij), (ik), and (jk) are *close pairs*. In addition, one of these pairs is restricted to be strong. The triples energy depends on the strong and close pair amplitudes. The close pair amplitudes are taken from the LMP2 calculation. Thus, increasing the distance or connectivity criteria for close and weak pairs will lead to more accurate triples energies. While for near equilibrium properties like geometries and harmonic vibrational frequencies the default values are normally appropriate, larger distance criteria are sometimes needed when computing energy differences, in particular barrier heights. In cases of doubt, `RWEAK` should first be increased until convergence is reached, and then `RCLOSE` can be varied as well. Such tests can be performed with small basis sets like cc-pVDZ, and the optimized values then be used in the final calculations with large basis sets.

29.9.7 Gradients and frequency calculations

Geometry optimizations [15-17] and numerical frequency calculations [18-20] can be performed using analytical energy gradients [15-17] for local MP2. LMP2 geometry optimizations are particularly attractive for weakly bound systems, since virtually BSSE free structures are obtained (see section 29.9.8 and Refs. [21-23]). For reasons of efficiency it is strongly advisable to use the DF-LMP2 Gradient [17] for all geometry optimizations. Setting `SCSGRD=1` on the `DF-LMP2` command or `DFIT` directive activates the gradient with respect to Grimmes SCS scaled MP2 energy functional (see also section DFIT). Analytical energy gradients are not yet available for the multipole approximation of distant pairs, and therefore `MULTP` cannot be used in geometry optimizations or frequency calculations.

In geometry optimizations, the domains are allowed to vary in the initial optimization steps. When the stepsize drops below a certain threshold (default 0.01) the domains are automatically frozen. In numerical Hessian or frequency calculations the domains are also frozen. It is therefore not necessary to include `SAVE` and `START` options.

Particular care must be taken in optimizations of highly symmetric aromatic systems, like, e.g., benzene. In D_{6h} symmetry, the localization of the π-orbitals is not unique, i.e., the localized orbitals can be rotated around the C_6 axis without changing the localization criterion. This redundancy is lost if the symmetry is slightly distorted, which can lead to sudden changes of the localized orbitals. If now the domains are kept fixed using the `SAVE` and `START` options, a large error in the energy might result. On the other hand, if the domains are not kept fixed, their size and quality might change during the optimization, again leading to spurious energy changes and divergence of the optimization.

The best way to avoid this problem is to use the `MERGEDOM=1` option (see section 29.6). If this option is given, the domains for the π orbitals will comprise the basis functions of all six carbon atoms, and the energy will be invariant with respect to unitary transformations among the three π orbitals. Note that this problem does not occur if the symmetry of the aromatic system is lowered by a substituent.

Redundant orbital rotations can also lead to convergence difficulties of the `Pipek-Mezey` localization. This can be overcome by using

```
PIPEK,METHOD=2
```

or

```
PIPEK,METHOD=3
```

With `METHOD=2`, the second derivatives of the localization criterion with respect to the orbital rotations is computed and diagonalized, and rotations corresponding to zero eigenvalues are eliminated. This method converges quadratically. With `METHOD=3` first a few iterations with the standard Pipek-Mezey method are performed, then the second-order method is invoked. This appears to be the most robust and accurate localization method.

Finally, we note that the LMP2 gradients are quite sensitive to the accuracy of the SCF convergence (as is also the case for MP2). If very accurate structures are required, or if numerical frequencies are computed from the gradients, the default SCF accuracy might be insufficient.

We recommend in such cases to add an `ACCU,14` directive (possibly even `ACCU,16`) after the `HF` command. Indicative of insufficient SCF accuracy are small positive energy changes near the end of the geometry optimization.

29.9.8 Intermolecular interactions

Local methods are particularly useful for the calculation of weak intermolecular interactions since the basis set superposition error (BSSE) is largely reduced [1,13,14] and counterpoise corrections are usually not necessary (provided the BSSE of the underlying Hartree-Fock is small). However, one must be careful to define the domains properly and to include all intermolecular pairs at the highest computational level. A convenient way to define appropriate domains and pair lists is to use the option `INTERACT=1`. If this option is given, individual molecules are identified automatically and all intermolecular pairs are automatically treated as strong pairs and included in the LCCSD. Similarly, appropriate triples lists are generated for LCCSD(T) calculations. It is required that all orbital domains are located on individual molecules. Note however that the inclusion of the intermolecular pairs strongly increases the number of strong pairs and triples, and therefore high-level calculations can become very expensive.

For calculations of interaction potentials of weakly interacting systems, the domains of the subsystems should be determined at a very large distance and saved using the `SAVE=`*record* option on the `LOCAL` or `MULTP` directive, or the `SAVE` directive (see section 29.8.3). If the asymptotic energy is not needed it is sufficient to do this initial calculation using option `DOMONLY=1`). These domains should then be reused in the subsequent calculations at all other intermolecular distances by using the `START=`*record* option or the `START` directive (see section 29.8.4). Only in this way the basis set superposition error is minimized and normally negligible (of course, this does not affect the BSSE for the SCF, and therefore the basis set should be sufficiently large to make the SCF BSSE negligible).

Usually, diffuse basis functions are important for obtaining accurate intermolecular interactions. When diffuse basis sets are used, it may happen that the Pipek-Mezey localization does not yield well localized orbitals. This problem can in most cases be overcome by using the directive

`PIPEK,DELETE=`*n*

as described in section 29.9.3

A final warning concerns local density fitting (see sections 29.10 and 15): local fitting must not be used in counterpoise calculations, since no fitting functions would be present on the dummy atoms and this can lead to large errors.

For examples and discussions of these aspects see Refs. [21-23]

29.10 Density-fitted LMP2 (`DF-LMP2`) and coupled cluster (`DF-LCCSD(T0)`)

Density-fitting LMP2 and LCCSD calculations can be performed by adding the prefix `DF-` to the command name. The input is as follows:

`DF-LMP2`,[*options*]
`DF-LCCSD(T)`,[*options*]

Options for density fitting can be mixed with any options for `LOCAL`. Options for density fitting can also be given on a `DFIT` directive (see section 15).

The most important options for density fitting in local methods are

BASIS_MP2=*string* Fitting basis set used in LMP2 and in LCCSD for integrals with up to 2 external orbitals. If a correlation consistent basis set is used (e.g. cc-pVTZ) the corresponding fitting basis for MP2 us used by default (cc-pVTZ/MP2FIT). Otherwise the fitting basis set must be defined in a preceding basis block (see section 11).

BASIS_CCSD=*string* Fitting basis set used in LCCSD for integrals over 3- and 4-external orbitals. The default is BASIS_MP2 and this is usually sufficient. However, the accurate approximation of 4-external integrals in LCCSD requires larger fitting basis sets than LMP2. Therefore, in order to minimize fitting errors, it is recommended to use the next larger fitting basis, e.g., BASIS_CCSD=VQZ for orbital basis VTZ.

LOCFIT=*value*: If LOCFIT=1 local fitting is enabled. This is necessary to achieve linear scaling in DF-LMP2 (see Refs. [11-14]). The errors introduced by local fitting are usually very small, but there are some exceptions. For instance, LOCFIT=1 must not be used in counterpoise calculations, see section 29.9.8) Note that for small molecules LOCFIT=1 can be more expensive than LOCFIT=0.

For further details and options for density fitting see section 15.

30 LOCAL METHODS FOR EXCITED STATES

30.1 Local CC2 and ADC(2)

Bibliography:

General local CC2 for excited states:
[1] D. Kats, T. Korona and M. Schütz, *Local CC2 electronic excitation energies for large molecules with density fitting*, J. Chem. Phys. **125**, 104106 (2006).
[2] D. Kats, T. Korona and M. Schütz, *Transition strengths and first-order properties of excited states from local coupled cluster CC2 response theory with density fitting*, J. Chem. Phys. **127**, 064107 (2007).
Laplace transformed local CC2 for excited states:
[3] D. Kats and M. Schütz, *A multistate local coupled cluster CC2 response method based on the Laplace transform*, J. Chem. Phys. **131**, 124117 (2009).
[4] D. Kats and M. Schütz, *Local Time-Dependent Coupled Cluster Response for Properties of Excited States in Large Molecules*, Z. Phys. Chem. **224**, 601 (2010).
[5] K. Freundorfer, D. Kats, T. Korona and M. Schtz, *Local CC2 response method for triplet states based on Laplace transform: Excitation energies and first-order properties*, J. Chem. Phys., accepted, (2010).
Previous work on local methods for excited states (LEOM-CCSD):
[6] T. Korona and H.-J. Werner *Local treatment of electron excitations in the EOM-CCSD method*, J. Chem. Phys. **118**, 3006 (2003).

All publications resulting from use of this program must acknowledge Ref. [3] (calculations of singlet states) and [5] (calculations of triplet states and properties).

The command LT-DF-LCC2 calls the Laplace transformed LCC2 program, and LT-DF-LADC(2) calls LADC(2). The excited states of interest should be specified on the EOM card.

30.2 Options for `EOM`

see also section 25.1.

In the case of LT-DF-LCC2/ADC(2) multistate calculations are possible, and it is recommended to calculate more states as needed.

The parameters on the `EOM` card:

`EOM`,-*n.1*, `key1`=*value1*, `key2`=*value2*,...

where *n.1* is the last state of interest, e.g., with `EOM`,-*5.1* the four lowest excited states will be calculated. The following keywords `key` are possible:

`SINGLET`=*ising*	If set to 1, singlet states will be calculated (default).
`TRIPLET`=*itrip*	If set to 1, triplet states will be calculated (not implemented for ADC(2)).
`EXFILE`=*record.ifil*	Record for converged CIS eigenvectors (default 6100.2).
`SAVE`=*record.ifil*	Record for save of restart information.
`SAVET`=*record.ifil*	Record for restart information for triplet states (if calculated together with singlet states).
`START`=*record.ifil*	Record for restart of previous calculation.
`STARTT`=*record.ifil*	Record for restart of previous calculation (if calculated together with singlet states).
`NSRCH4ST`=*nst*	In the first *nst* iterations in the Davidson diagonalisation the excited state domains are determined for each basis vector in the Davidson subspace ("search for states") (default 7).
`THRLCH`=*thrlch*	threshold for the Davidson procedure. If smaller than zero, the Davidson procedure is skipped and DIIS is started directly instead (possible only for restart, `SAVE` and `START` have to be identical).
`THRLCD`=*thrlcd*	threshold for DIIS.
`ADC2`=*adc2*	if = 1, do ADC(2) calculation instead of CC2. if = 2, use LT-DF-LMP2 for the ground state.

Default local approximations are defined according to procedure described in Ref. [3] (Laplace domains).

`INTFRAC`=*fracint*	Rough criterion for specifying eom-domains from laplace-transformed integrals (default 0.8).
`INTEXC`=*excint*	Criterion for specifying important orbitals from laplace-transformed integrals (default 0.999).
`REALFRAC`=*fracreal*	Exact criterion for specifying eom-domains from laplace-transformed integrals (default 0.98).
`FULLFRAC`=*fracfull*	Check for all orbital domains (complete sum over all orbitals) (default 0.95).

To switch to local aproximations calculated according to Ref. [6] (Boughton-Pulay procedure for excited states), set `INTFRAC` to zero.

Occupied orbital pair lists are calculated from important orbitals (Refs. [1,5]).

`EWEAKPAIR=`*ewpair*	Distance criterion for excited state orbital pairs definition (default 5).
`ALLSTRONG=`*allstr*	Different possibilities for excited state orbital pairs: 0: $[ij] \leq ewpair$, $[im] \leq ewpair$, $[mn] \leq ewpair$; 1: $\forall[ij]$, $\forall[im]$, $[mn] \leq ewpair$; 2: $\forall[ij]$, $[im] \leq ewpair$, $[mn] \leq ewpair$ (default); where i, j are important orbitals and m, n non-important. More detailed see Ref. [1].

One can try to improve the convergence of iterative procedures by changing following parameters

`PRECOND=`*prec*	Type of preconditioner in LT-DF-LCC2/ADC(2) and DF-CIS: 0: canonical orbital energies; 3: solving linear equations including diagonal part of $(H-F)^{\text{CIS}}$ with MINRES (default).
`HSCAL=`*hscl*	Scaling factor for diagonal part of $(H-F)^{\text{CIS}}$ matrix in the case of linear equations preconditioner (default 0.7).

Properties are also activated on the `EOM` card:

`PROPES=`*prop*	States for which properties (dipole moments) should be calculated, e.g., `PROPES=-3.1+5.1-8.1+15.1` $\Longrightarrow$ 2.1 3.1 5.1 6.1 7.1 8.1 15.1
`TRANES=`*tran*	States for which transition moments should be calculated, syntax like for properties. Is not implemented for ADC(2) and for triplet states.
`DENSAVE=`*rec.ifil*	Record, where densities calculated for states *prop* can be saved (for printing, see 30.4)

30.3 Parameters on `LAPLACE` card

`NPOINTS`	Number of quadrature points (default (for CC2) 3).
`NPOINTS_CC2S`	Number of quadrature points in CC2 in "search for states" (default 1).
`DISTRF`	Type of distribution function in functional for determination of Laplace points: 0: integral with f_x=1; 4: sum of distributions f_x (default).
`NULLPOINT`	If larger than zero, use also a Laplace-point with t=0. In case of CC2 is always zero.

30.4 Print options

Setting `GPRINT,CIVECTOR=1` prints the ten largest canonical contributions from singles vectors in each iteration.

For `EOMPRINT` card:

`LOCEOM`	≥ 0: print eom domain informations
`POPUL`	≤ 0: dont print population analysis

The densities saved with `DENSAVE` can be written into a cube file. Corresponding commands in `CUBE` module 34.7 are

`DENSITY,`*rec.ifil*`,cc2` Creates a cube file from the ground state density saved on record *rec.ifil*.

`DENSITY,`*rec.ifil*`,full,state=`*st.1* Creates a cube file from the excited state density of state *st.1*.

`DENSITY,`*rec.ifil*`,diff,state=`*st.1* Creates a cube file from the difference density (`full` - `cc2`) for state *st.1*.

30.5 Examples

Example 1:

```
***,The five lowest singlet excited states of water molecule
memory,64,m
gdirect
symmetry,nosym;orient,noorient
geometry={
o ,,  0.000,  0.000,  0.119
h1,,  1.423,  0.000, -0.947
h2,, -1.423,  0.000, -0.947 }
basis={
default,vdz
set,mp2fit
default,vdz/mp2fit
set,jkfit
default,vdz/jkfit }
hf
{lt-df-lcc2                         !ground state CC2
eom,-6.1                            !five lowest states
eomprint,popul=-1,loceom=-1 }       !minimize the output
```

The excitation energies (in eV) stand after the calculation in array `OMEGAF_S` for singlet states and in `OMEGAF_T` for triplet states.

Example 2:

```
***,The five lowest triplet excited states and properties
memory,64,m
gdirect
symmetry,nosym;orient,noorient
geometry={
o ,,  0.000,  0.000,  0.119
h1,,  1.423,  0.000, -0.947
h2,, -1.423,  0.000, -0.947 }
basis={
default,vdz
set,mp2fit
default,vdz/mp2fit
set,jkfit
```

```
default,vdz/jkfit }
hf
{lt-df-lcc2                    !ground state CC2
!five lowest triplet states, dipole moments for four lowest, density saved:
eom,-6.1,triplet=1,singlet=0,propes=-5.1,densave=5000.2
eomprint,popul=-1,loceom=-1 }   !minimize the output
!ground state density, file h2o0_density.cube:
{cube,h2o0,,80,80,80
density,5000.2,cc2 }
!difference excited state density, file h2o1_density.cube:
{cube,h2o1,,80,80,80
density,5000.2,diff,state=2.1 }
```

31 EXPLICITLY CORRELATED METHODS

Explicitly correlated calculations provide a dramatic improvement of the basis set convergence of MP2 and CCSD correlation energies. Such calculations can be performed using the commands of the form

command, *options*

where *command* can be one of the following:

MP2-F12	Closed-shell canonical MP2-F12. By default, the fixed amplitude ansatz (`FIX`, see below) is used, but other ansätze are also possible. The F12-corrections is computed using density fitting, and then added to the MP2 correlation energy obtained without density fitting.
DF-MP2-F12	As MP2-F12, but the DF-MP2 correlation energy is used. This is less expensive than MP2-F12.
DF-LMP2-F12	Closed-shell DF-MP2-F12 with localized orbitals. Any method and ansatz as described in J. Chem. Phys. **126**, 164102 (2007) can be used (cf. secions 31.2,31.7).
DF-RMP2-F12	Spin-restricted open-shell DF-RMP2-F12 using ansatz 3C. Any method as described in J. Chem. Phys. **128**, 154103 (2008) can be used (cf. sections 31.2,31.7).
CCSD-F12	Closed-shell CCSD-F12 approximations as described in J. Chem. Phys. **127**, 221106 (2007). By default, the fixed amplitude ansatz is used and the CCSD-F12A and CCSD-F12B energies are computed. Optionally, the command can be appended by A or B, and then only the corresponding energy is computed. For more details see section 31.10.
CCSD-F12c	Closed-shell CCSD(F12*) approximation as proposed by Hättig, Tew and Köhn. In Molpro this is denoted f12c. As compared to CCSD-F12a/b it requires additional computational effort. Since in some parts the implementation is brute-force without paging algorithms, large memory may be required. In most cases there is no gain in accuracy as compared to f12b and therefore the use of this method is normally not recommended. Currently CCSD-F12c is not available for open-shell cases.
CCSD(T)-F12	Same as CCSD-F12, but perturbative triples are added.

CCSD(T)-F12c	Same as CCSD-F12c, but perturbative triples are added.
UCCSD-F12	Open-shell unrestricted UCCSD-F12 approximations as described by G. Knizia, T. B. Adler, and H.-J. Werner, J. Chem. Phys. **130**, 054104 (2009). Restricted open-shell Hartree-Fock (RHF) orbitals are used. Optionally, the command can be appended by A or B, and then only the corresponding energy is computed. For more details see section 31.10.
UCCSD(T)-F12	Same as UCCSD-F12, but perturbative triples are added.
RCCSD(T)-F12	Similar to UCCSD(T)-F12, but the partially spin-adapted scheme is used.

Published work arising from these methods should cite the following:

F. R. Manby, J. Chem. Phys. **119**, 4607 (2003)
(for the density fitting approximations in linear R12 methods)

A. J. May and F. R. Manby, J. Chem. Phys. **121**, 4479 (2004)
(for the frozen geminal expansions)

H.-J. Werner and F. R. Manby, J. Chem. Phys. **124**, 054114 (2006);
F. R. Manby, H.-J. Werner, T. B. Adler and A. J. May, J. Chem. Phys. **124**, 094103 (2006);
H.-J. Werner, J. Chem. Phys. **129**, 101103 (2008);
T. B. Adler, H.-J. Werner, and F. R. Manby, J. Chem. Phys. **130**, 054106 (2009);
(for DF-LMP2-F12).

H.-J. Werner, T. B. Adler, and F. R. Manby, J. Chem. Phys. **126**, 164102 (2007)
(for all other closed-shell MP2-F12 methods).

G. Knizia and H.-J. Werner, J. Chem. Phys. **128**, 154103 (2008)
(for all open-shell F12 calculations).

T. B. Adler, G. Knizia and H.-J. Werner, J. Chem. Phys. **127**, 221106 (2007)
(for CCSD(T)-F12).

G. Knizia,T. B. Adler, and H.-J. Werner, J. Chem. Phys. **130**, 054104 (2009)
(for CCSD(T)-F12 and UCCSD(T)-F12 calculations).

T. B. Adler and H.-J. Werner, J. Chem. Phys. **130**, 241101 (2009)
(for LCCSD-F12).

K.A. Peterson, T. B. Adler, and H.-J. Werner, J. Chem. Phys. **128**, 084102 (2008)
(for the VnZ-F12 basis sets)

K. E. Yousaf and K. A. Peterson, J. Chem. Phys. **129**, 184108 (2009)
(for the VnZ-F12/OPTRI basis sets)

K. E. Yousaf and K. A. Peterson, Chem. Phys. Lett., **476**, 303 (2009)
(for the AVnZ/OPTRI basis sets)

In the following, we briefly summarize the ansätze and approximations that can be used. For more details and further references to related work of other authors see H.-J. Werner, T. B. Adler, and F. R. Manby, *General orbital invarient MP2-F12 theory*, J. Chem. Phys. **126**, 164102 (2007) (in the following denoted **I**).

31.1 Reference functions

The MP2-F12, CCSD-F12, and UCCSD-F12 methods must use conventional (non-density fitted) spin-restricted Hartree-Fock reference functions (HF or RHF). DF-HF cannot be used for these methods. This restriction is necessary to ensure that the Fock matrix is diagonal and consistent with the integrals used in these methods. For DF-MP2-F12, DF-LMP2-F12, and DF-RMP2-F12 either HF or DF-HF reference functions can be used.

Currently, only finite dipole fields can be applied, other perturbations are not yet supported. ECPs can be used, but this is still experimental and not extensively benchmarked. The Douglas-Kroll-Hess Hamiltonian cannot be used in combination with F12 methods.

31.2 Wave function Ansätze

The so called "ansatz" determines the definition of the explicitly correlated wave function. This is to be distinguished from the various approximations that can be used to approximate the Hamiltonian matrix elements. Generally, we use ansatz **3** (cf. **I**), for which the projector has the form

$$\hat{Q}_{12} = (1-\hat{o}_1)(1-\hat{o}_2)(1-\hat{v}_1\hat{v}_2),$$

where $\hat{o}_i$ is a one-electron projector for electron i onto the occupied space, and $\hat{v}_i$ projects onto the virtual orbital space. In the case that domain approximations are used in local explicitly correlated wave functions, the operators $\hat{v}$ are replaced by operators $\hat{d}^{ij}$ that project just onto the domain for the orbital pair ij.

In MOLPRO the following wave function ansätze can be used:

31.2.1 The *general ansatz*

The conventional external pair functions are augmented by terms of the form

$$|u^{\mathrm{F12}}_{ijp}\rangle = \sum_{p=\pm 1}\sum_{kl} T^{ijp}_{kl}\hat{Q}_{12}\hat{F}_{12}|kl\rangle$$

This ansatz is orbital invariant (i.e., the same results are obtained with canonical or localized orbitals), but it often suffers from geminal basis set superposition errors. Furthermore, singularities may occur in the zeroth-order Hamiltonian, in particular for larger systems. Therefore, this ansatz is normally not recommended.

31.2.2 The *diagonal ansatz* (`D`)

The sum over kl in equation (63) is restricted to ij. This ansatz is not orbital invariant and size consistent only when localized orbitals are used. However, geminal basis set superposition errors are absent and therefore the results are often more accurate than with the general ansatz.

31.2.3 The *fixed amplitude ansatz* (FIX)

The diagonal ansatz is used and the amplitudes of the explicitly correlated configurations are determined by the wavefunction cusp conditions, i.e.

$$T_{ij}^{ij,1} = \frac{1}{2}$$
$$T_{ij}^{ij,-1} = \frac{1}{4}$$

This ansatz is orbital invariant, size consistent and free of GBSSE.

31.3 RI Approximations

Various approximations such as A, B, C, HY1, HY2 exist for the matrix elements of the first-order Hamiltonian (see **I**). They differ in the way the RI approximations are made. In the limit of a complete RI basis, approximations B and C are identical and yield the exact result for a given wave function ansatz. We generally recommend approximation C, which is simpler and more efficient than approximation B. Normally, the union of the AO and RI basis sets is used to approximate the resolution of the identity (CABS approach). In the hybrid approximations (HY1, HY2, HX) only the AO basis is used in some less important terms. Together with the recommended approximation C, HY1 or HY2 can be used; HY2 is more accurate, HY1 more efficient. In most cases, approximation 3C(HY1) provides an excellent compromise between accuracy and efficiency. In approximation A, all terms involving exchange operators are neglected. This approximation is used along with local approximations in our low-order scaling LMP2-F12/3*A(loc) method that can be applied to large molecules (cf. section 31.11)

If the extended Brillouin condition (EBC, see **I**) is assumed, the explicitly correlated and conventional amplitude equations decouple and can be solved independently. These approximations are denoted by a star, e.g. **3***C.

31.4 Basis sets

In MOLPRO the F12 integrals can only be computed using density fitting (DF) approximations. The many electron integrals are approximated by resolutions of the identity (RI) expansions. Thus, F12 calculations require three different basis sets: the orbital (AO) basis, the DF basis, and the RI basis.

We recommend as AO basis sets the augmented correlation consistent basis sets (denoted AVnZ) or the specially optimized correlation consistent F12 basis sets (denoted VnZ-F12, cf. K.A. Peterson and H.-J. Werner, J. Chem. Phys. **128**, 084102 (2008)). Normally, triples zeta basis sets (AVTZ or VTZ-F12) yield excellent results that are close to the basis set limit. Diffuse basis functions are rather essential both for the HF and MP2-F12 energies, and therefore the standard VTZ sets are not recommended. If the AVnZ or VnZ-F12 orbital basis sets are used, suitable density fitting (DF) basis and resolution of the identity (RI) basis sets are automatically chosen. For the AVnZ orbital basis sets, AVnZ/MP2FIT amd VnZ/JKFIT basis sets are used by default for the DF and RI, respectively. The associated optimized CABS basis set of Peterson et al. can be chosen by specifying `RI_BASIS=OPTRI`. For the VnZ-F12 orbital basis, the associated CABS (OPTRI) basis sets are used by default. Other basis sets can be chosen using the `DF_BASIS`, `DF_BASIS_EXCH` and `RI_BASIS` options (cf. section 31.6). See section 15 for more details about density fitting.

This is an example for using multiple basis sets for density fitting and resolution of the identity

```
***,h2o
geom={o;
      h1,o,r;
      h2,o,r,h1,theta}
r=0.97 ang
theta=104

basis={
default,avtz
set,df
default,avtz/mp2fit   !density fitting basis
set,jk
default,avtz/jkfit    !density fitting basis for Fock and exchange matrices
set,ri
default,avtz/optri    !ri cabs basis
}
hf
ccsd(t)-f12,df_basis=df,df_basis_exch=jk,ri_basis=ri
```

http://www.molpro.net/info/current/examples/h2o_basissets1.com

The following two examples yield identical results:

```
***,h2o
geom={o;
      h1,o,r;
      h2,o,r,h1,theta}
r=0.97 ang
theta=104

basis={
default,avtz
set,df,context=mp2fit
default,avtz          !density fitting basis
set,jk,context=jkfit
default,avtz          !density fitting basis for Fock and exchange matrices
set,ri,context=optri
default,avtz          !ri cabs basis
}
hf
ccsd(t)-f12,df_basis=df,df_basis_exch=jk,ri_basis=ri
```

http://www.molpro.net/info/current/examples/h2o_basissets2.com

```
***,h2o
geom={o;
      h1,o,r;
      h2,o,r,h1,theta}
r=0.97 ang
theta=104

basis=avtz
hf
ccsd(t)-f12,df_basis=avtz/mp2fit,df_basis_exch=avtz/jkfit,ri_basis=avtz/optri
```

http://www.molpro.net/info/current/examples/h2o_basissets3.com

In the latter example the specifications MP2FIT and JKFIT could be omitted, i.e. the input

`ccsd(t)-f12,df_basis=avtz,df_basis_exch̄vtz,ri_basis=optri`

would be equivalent. In fact, since default density fitting basis sets are used

```
ccsd(t)-f12,ri_basis=optri
```

would be sufficient. Note, however, that without the specification OPTRI the avtz/jkfit basis set would be used for the RI. For example,

```
basis=avtz
hf
ccsd(t)-f12
```

is equivalent to

```
basis=avtz
hf
ccsd(t)-f12,df_basis=avtz/mp2fit,df_basis_exch=vtz/jkfit,ri_basis=vtz/jkfit
```

If the VnZ-F12 basis sets are used, the OPTRI sets are used by default, i.e.,

```
basis=vtz-f12
hf
ccsd(t)-f12
```

is equivalent to

```
basis=vtz-f12
hf
ccsd(t)-f12,df_basis=avtz/mp2fit,df_basis_exch=vtz/jkfit,\\
ri_basis=vtz-f12/optri
```

31.5 Symmetry

Symmetry cannot be used in DF-LMP2-F12 calculations. However, in MP2-F12, DF-MP2-F12, DF-RMP2-F12, CCSD(T)-F12 and UCCSD(T)-F12 calculations Abelian symmetry can be used as usual; in these cases the preceding DF-MP2-F12 calculations will be automatically performed without symmetry, and the integrals that are necessary for subsequent CCSD-F12 or UCCSD-F12 calculations will be transformed to the symmetry adapted basis. This is fully automatic and transparent to the user. Note, however, that the prefix DF- turns off symmetry automatically, and if you want to use symmetry in the HF calculations preceding the DF-MP2-F12 or DF-MP2-F12 calculations the symmetry elements (or `AUTO`) must be specified using the `SYMMETRY` directive (setting `SYMMETRY,AUTO` works fine).

31.6 Options

There are many options available, but these are hardly needed. Normally, when standard orbital basis sets such as aug-cc-PVTZ or VTZ-F12 are used, appropriate defaults are used and no further options are needed. A typical input simply reads

```
basis=vtz-f12
hf
ccsd(t)-f12
```

We recommend to specify options only when necessary and when it is well understood what they mean!

Options for canonical and local versions:

`DF_BASIS=`*basis* Select the basis for density fitting (see section 15 for details). *basis* can either refer to a set name defined in the basis block, or to a default MP2 fitting basis (e.g., `DF_BASIS=VTZ` generates the `VTZ/MP2FIT` basis). By default, the MP2FIT basis that corresponds to the orbital basis is used.

`DF_BASIS_EXCH=`*basis* Select the density fitting basis for computing the exchange and Fock operators. By default, the JKFIT basis sets which correspond to the orbital basis are used.

`RI_BASIS=`*basis* Select the basis for the resolution of the identity (RI). This can refer to a default basis set or a set name defined in a basis block. For F12 methods the Hartree-Fock JKFIT basis sets perform well for the RI, despite having been optimized for other purposes. These sets are used by default for the AVnZ orbital basis sets. The basis type can be appended to the basis name after a slash, e.g. `RI_BASIS=AVQZ/JKFIT` would use the specified JKFIT set, or `RI_BASIS=AVQZ/OPTRI` would use the optimized CABS basis sets of Peterson et al. These are recommended for the AVnZ and VnZ-F12 basis sets and used by default for the latter ones. Note that each OPTRI basis is associated to a specific orbital basis. Therefore, the name of the OPTRI basis must either be the same as that of the orbital basis, or be omitted, e.g., `RI_BASIS=OPTRI` selects automatically the correct set for the current orbital basis.

In case of R12-methods (which are not recommended to be used), the RI basis should be chosen to be a large uncontracted AO basis (at least AVQZ). Contraction/uncontraction can be forced appending `(CONTRACT)` or `(UNCONTRACT)` to the basis name, e.g., `RI_BASIS=AVQZ(UNCONTRACT)/ORBITAL`. If other options are given in parenthesis, these can be separated by commas, e.g., `RI_BASIS=AVQZ(f/d,UNCONTRACT)/ORBITAL`.
Alternative forms, which should work as well, are
`RI_BASIS=AVQZ(f/d)(UNCONTRACT)/ORBITAL`
or
`RI_BASIS=AVQZ(f/d)/ORBITAL(UNCONTRACT)`.
Note that the CONTRACT/UNCONTRACT option cannot be used with basis set names previously defined in a basis block.

`CONTEXT=`*context* Can be used to change the default type for the RI basis, e.g. `CONTEXT=OPTRI` will use the OPTRI basis sets that correspond to the VnZ-F12 or AVnZ basis sets.

`ANSATZ=`*ansatz* Select the explicitly correlated ansatz *ansatz* methods. See section 31.7 for the possibilities and further details.

`GEM_BASIS` Basis set name for geminal expansion; atom labels are ignored. This can either be `OPTFULL` (full nonlinear fit of the geminal expansion), `EVEN` (even tempered fit), or refer to a set name defined in a previous `BASIS` block. Default is `OPTFULL`.

`GEM_TYPE` Frozen geminal type: `LINEAR` or `SLATER`, default is `SLATER`.

GEM_NUMBER	Number of Gaussian geminal functions (default 6).
GEM_CENTRE	Centre of even tempered geminal exponents, if GEM_BASIS=EVEN (default 1.0).
GEM_RATIO	Ratio of even tempered geminal exponents, if GEM_BASIS=EVEN (default 3.0).
GEM_BETA	Exponent for Slater-type frozen geminal, or parameter for weight function in other frozen geminal models (default 1.0 a_0^{-1}). It is possible to specify extra exponents for core-core and core-valence correlation. If two values are given (in square brackets), the first is used for valence pairs, the second for core-core (cc) and core-valence (cv) pairs. If three values are given, the first is used for vv, the second for cv, and the third for cc correlation.
GEM_OMEGA	Exponent for weighting function (default -1, which means a value derived from GEM_BETA).
GEM_MOM	Exponent for r in omega fitting (default 0).
GEM_M	Exponent for r in weighting function (default 0).
GEM_MAXIT	Max. number of iterations in geminal optimization (default 200).
GEM_PRINT	Print parameter for geminal optimization (default 0).
GEM_DEBUG	Debug option for geminal optimization (default 0).
GEM_ACC	Convergence threshold for geminal line search (default 0.001).
GEM_FAC	Scaling factor for exponents in geminal optimization (default 1.0).
GEM_METHOD	Geminal optimization method (augmented Hessian (AH) or Newton-Raphson (NR), default AH).
GEM_TRUST	Trust ratio in AH geminal optimization (default 0.4).
GEM_SHIFT	Hessian shift in AH geminal optimization (default 0).
GEM_NUMERICAL	Flags numerical integration in geminal optimization (default 0).
GEM_PLOT	Geminal plot file (default blank).
GEM_OPT_FULL	If nonzero (default), fit each geminal independently to Gaussians (if several exponents are used). If zero, the first exponent is fitted, unless GEM_BETA_OPT is specified.
GEM_BETA_OPT	Exponent used to fit the Gaussian expansion.
SIM_MULTGEM	Only for calculation with multiple exponents: if nonzero, loop externally over integral program for different exponents. This is implied automatically if each geminal is fitted independently. If the same Gaussian exponents are used for each Slater exponent, SIM_MULTGEM=0 can be used (default). In this case the integral program handles the general contractions (slightly faster).
PRINT=*ipri*	Select output level: ipri=0 — Standard output ipri=1 — Standard output plus more detailed information about integral evaluations. ipri=2 — Debugging output
THRBINV	Threshold below which non-physical eigenvalues are projected from approximate B matrices
THRF12	Threshold for integral screening contribution.

31.7 Choosing the ansatz and the level of approximation

The Ansatz can be chosen using the `ANSATZ` option and/or by options on the command line.

`3A`	Ansatz 3A;
`3*A`	Ansatz 3A with EBC approximation
`3B`	Ansatz 3B
`3*B`	Ansatz 3B with EBC approximation
`3C`	Ansatz 3C
`3*C`	Ansatz 3C with EBC approximation

The ansatz can be further detailed by appending options in parenthesis, e.g.

`ANSATZ=3B(D)`

These options can be one of

`D`	Use diagonal ansatz
`FIX`	Use diagonal ansatz with fixed amplitudes (orbital invariant)
`FIXC`	Use diagonal ansatz with fixed amplitudes and canonical orbitals
`DX`	Use diagonal ansatz and assume X-matrix to be diagonal (only for Ansatz 3A)
`GBC`	Use GBC approximation (only for 3B, default in 3A)
`EBC`	Use EBC approximation (same as *)
`HX1`	Use HX1 approximation (only for 3B).
`HY1`	Use HY1 approximation (only for 3B and 3C).
`HY2`	Use HY2 approximation (only for 3B and 3C).
`HY`	Default hybrid approximation. HX1 and HY2 approximation in 3B, HY2 in 3C.
`NOZ`	Neglect Z terms (only for 3B and 3C).
`NOX`	Neglect X terms (only for 3A and 3B).

Several options separated by commas can be given. For instance

`ANSATZ=3C(FIX,HY1)`

uses the diagonal ansatz 3C(D) with fixed coefficients and the hybrid (HY1) approximation.

Alternatively or in addition, the following options can be given on the command line:

`DIAG=1`	Use diagonal ansatz.
`DIAGX=1`	Use diagonal ansatz and assume X-matrix to be diagonal.
`GBC=1`	Use GBC approximation (only for 3B, default in 3A).
`EBC=1`	Use EBC approximation (same as *).
`HYPRID=n`	Use HYn approximation.
`HYPRIDX=1`	Use HX1 approximation.

`NOZ=1`	Neglect Z terms (only 3B and 3C).
`NOX=1`	Neglect X terms (only 3A and 3B).
`FIX=1`	Use diagonal ansatz with fixed coefficient approximation (orbital invariant).
`FIX=2`	Use diagonal ansatz with fixed coefficient approximation. Evaluate only first order energy expression, not the Hylleraas functional. Very fast but less accurate and reliable!
`FIXCAN=1`	Use diagonal ansatz with fixed coefficient approximation and canonical orbitals. A non-iterative method is used to evaluate the energy. This is equivalent to `FIX=1,CANONICAL=1` and is most efficient.
`FIXCAN=-1`	As `FIXCAN=1`, but equations are solved iteratively (test purpose only).
`CABS=1`	Use CABS (default). If `CABS=0` is given, CABS is disabled. However, if `RI_BASIS=OPTRI`, the orbital and OPTRI basis sets are automatically merged, and then exactly the same results as with `CABS=1` are obtained.
`ORTHO_CABS=1`	Construct CABS basis from orthogonal MOs and ABS basis rather than AO and RI basis.
`THRABS=`*thrabs*	Threshold for smallest eigenvalue of S in auxiliary ABS (only used with `ORTHO_CABS=1`; default=`THRCABS`).
`THRCABS=`*thrcabs*	Threshold for smallest eigenvalue of S in CABS (default 1.d-8).
`THRCABS_REL=`*thrcabs_rel*	Relative CABS threshold (default 1.d-9). The actual threshold is `max(thrcabs,eigmax*thrcabs_rel`, where `eigmax` is the largest eigenvalue of the overlap matrix.
`PRINT=`*level*	Print parameter. `PRINT=1` give information about all computed integrals and the iterations.
`DEBUG=`*level*	Can be used to obtain extended debug print.
`SOLVE=0`	Use a most efficient pair-specific fully iterative method (default).
`SOLVE=1`	Use simple fully iterative method.
`SOLVE=2`	Use pair specific iterative method (more expensive).
`SOLVE=3`	Use pair specific non iterative method (most expensive, only with canonical orbitals).
`CANONICAL=1`	Use canonical orbitals and full domains.
`DOMSEL=1`	Use full domains and localized orbitals (unless `CANONICAL=1` is given).
`SCALE_TRIP=1`	Scale triples energy as explained in section 31.10.
`SINGLES`	If set to one, include CABS singles correction (default=1)
`CORE_SINGLES`	If set to one, include CABS singles correction for core orbitals (default=0)
`EXTGEN`	For open-shell systems: If 1 (default,recommended), include all occupied valence orbital pairs for mn in T^{ij}_{mn}, independent of spin (as described in J. Chem. Phys. 130, 054104 (2009), section II.E). If 0, use only pairs mn where the spins of i and m, and j and n are equal.

For instance

```
ANSATZ=3C,fix=1,hybrid=1,canonical=1
```

implies a canonical 3C calculation with diagonal ansatz 3C, using fixed coefficient and hynrid approximations. The combination of the options `fix=1` and `canonical=1` implies a non-iterative calculation of the energy and is recommended. The above is equivalent to all of the following:

```
ANSATZ=3C(FIXC,HY1)
ANSATZ=3C(D,FIXC,HY1)
ANSATZ=3C(D,HY1,FIX),canonical=1
```

Note that the HF convergence threshold should be rather strict to obtain accurate results (use `ACCU,14` in the HF).

Numerous further options are for specialist use only and not described here. See `explicit.registry` for a full list.

31.8 CABS Singles correction

By default, the perturbative CABS singles correction as described in J. Chem. Phys. **127**, 221106 (2007) and J. Chem. Phys. **128**, 154103 (2008) is included in the reference energy of all MP2-F12 and CCSD-F12 calculations (closed and open-shell, except for LMP2-F12/3*A(loc), which is done with a different program). The corrected reference energy is stored in variable `ENERGR`, so that `ENERGY-ENERGR` are the total correlation energies. For the setting of other variables by the F12 programs see section 31.12.

The singles correction can be turned off by option `SINGLES=0`, e.g.

```
MP2-F12,SINGLES=0
```

The contribution of core orbitals to the singles energy is not included by default, but can be turned on by option `CORE_SINGLES`, e.g.

```
MP2-F12,CORE_SINGLES=1
```

However, we do not recommended the use of core singles, because they depend sensitively on the CABS basis construction and do not offer significant improvements in relative energies.

31.9 Pair specific geminal exponents

Different Slater exponents can be used for core-core, core-valence and valence-valence pairs as described in H.-J. Werner, G. Knizia, and F. R. Manby (Mol. Phys., submitted). The exponents are specified using the `GEM_BETA` option, e.g., `GEM_BETA=[1.0, 1.7, 2.5]` (see options, section 31.6). The three values are used for vv, cv, cc pairs, respectively. In most cases, core pairs can be defined by using the `CORE` directive: all orbitals that are not core and do not belong to the default valence shell are then treated as core. If part of the default valence shell is to be taken as core (e.g., the 3d shell in first-row transition metals), the core can be defined via the `PRCORE` variable, e.g., `prcore=[4,2,2,1,4,2,2,1]` for Cu_2. This variable must then be defined before the F12 calculation that should use it. The following example shows calculation for Br_2, in which the 3*d* shell is treated as core.

```
memory,32,m
gthresh,energy=1.d-8
geometry={br;br,br,rmin}

rmin=2.281 ANG

basis={
ecp,Br,ecp10mdf
sp,Br,aug-cc-pVTZ-PP;c;
d,Br,338.996,103.217,42.3638,18.4356,8.37254,3.80222,1.68677,0.677520
c,1.8,0.001524,0.015673,0.072400,0.186303,0.323881,0.374534,0.257418,0.068051
d,Br,2.9173,0.6300,0.2228
f,Br,1.4417

set,dfmp
s,Br,40.9778,22.0940,14.1569,6.30933,3.49893,2.07145,1.22411,0.706125,0.421245,0.235424,0.1334
p,Br,31.5779,17.7012,10.3673,5.56829,3.34725,1.86205,1.12814,0.520803,0.290648,0.155863
d,Br,35.4419,14.4095,7.13728,4.02429,2.19901,1.18121,0.537101,0.320289,0.151982
f,Br,16.6283,7.92639,3.69263,1.89967,1.02171,0.418748
g,Br,58.3022,21.9356,7.85160,5.60937,2.04334,1.37529
h,Br,7.51453,3.15089

set,ri
s,Br,18.7563,9.07737,4.87021,2.00555,0.900841,0.080149
p,Br,26.0294,17.4402,5.55778,3.70554,2.47230,1.18351,0.511543,0.177391
d,Br,15.9542,11.4584,8.89027,1.44361,0.945611,0.334128
f,Br,42.4889,15.5186,10.3279,6.36172,2.80704,0.961178,0.415934
g,Br,24.3260,9.29525,3.78033,1.65385,0.919948
h,Br,18.4810,7.54637,2.84990,0.966374
i,Br,11.4466
}

explicit,ri_basis=ri,df_basis=dfmp,df_basis_exch=def2-qzvpp/jkfit

hf;accu,16
i=1
{mp2-f12,gem_beta=[1.25];core,2,1,1,,2,1,1}
emp2f12(i)=energy-energr
ef12_vv(i)=energ_vv
ef12_cv(i)=energ_cv
ef12_cc(i)=energ_cc
$beta(i)='[1.25]'
i=i+1
{mp2-f12,gem_beta=[0.8,1.7];core,2,1,1,,2,1,1}
emp2f12(i)=energy-energr
ef12_vv(i)=energ_vv
ef12_cv(i)=energ_cv
ef12_cc(i)=energ_cc
$beta(i)='[0.8,1.7]'

i=i+1
{mp2-f12,gem_beta=[0.8,1.7,2.2];core,2,1,1,,2,1,1}
emp2f12(i)=energy-energr
ef12_vv(i)=energ_vv
ef12_cv(i)=energ_cv
ef12_cc(i)=energ_cc
$beta(i)='[0.8,1.7,2.2]'

table,beta,ef12_cc,ef12_cv,ef12_vv,emp2f12
Title,Results for Br2, basis vdz-f12
```

http://www.molpro.net/info/current/examples/br2_f12_multgem.com

31.10 CCSD(T)-F12

The CCSD-F12 and UCCSD-F12 programs first do DF-MP2-F12/3C(FIX) (closed-shell) or DF-RMP2-F12/3C(FIX) (open-shell) calculations, and then perform the CCSD-F12 (UCCSD-F12) without density fitting. By default, the CCSD-F12A and CCSD-F12B energies are both computed. A specific method can be requested by appending A or B to the -F12 suffix. Furthermore, instead of the 3C(FIX) ansatz, different ansätze (e.g. 3C) can be used. In this case the amplitudes of the explicitly correlated terms are determined in the MP2-F12 calculation and kept fixed in the CCSD-F12.

It should be noted that these methods involve approximations and do not yield the exact CCSD-F12 energies. Preliminary experience has shown that the CCSD-F12A method slightly overestimates the correlation energies, while CCSD-F12B underestimates them. For AVDZ or AVTZ basis sets, CCSD-F12A usually gives very good results, but for larger basis sets it may overestimate the basis set limit and converge from below to the limit. Thus, convergence may not be monotonic, and extrapolation of the correlation energies should not be attempted. CCSD-F12B usually converges monotonically from below to the limit and gives best results for AVQZ and larger basis sets. Thus, we currently recommend CCSD-F12A for AVDZ and AVTZ basis sets, and CCSD-F12B for larger basis sets (rarely needed).

The perturbative triples correction can be invoked by using CCSD(T)-F12 or UCCSD(T)-F12. There is no direct F12 correction to the triples, and therefore the basis set error of the triples is not affected by the F12 (small changes of the triples energy arise from the fact that the doubles amplitudes are affected by the F12 terms). In many cases, a simple and pragmatic improvement of the triples energy can be obtained by scaling the triples energy contribution as

$$\Delta E_{(T*)} = \Delta E_{(T)} * E_{corr}^{MP2-F12} / E_{corr}^{MP2}$$

This can be done automatically by setting option `SCALE_TRIP=1`, i.e.

```
CCSD(T)-F12,SCALE_TRIP=1
```

31.11 DF-LMP2-F12 calculations with local approximations

Local variants of the DF-MP2-F12 methods are invoked by the commands DF-LMP2-F12 with ansatz 3*A(loc). These calculations are performed with a different program than non-local calculations, and currently do not include the perturbative CABS singles correction (since CABS is not used in this program). The (loc) option implies that the LMP2 calculation with domain approximations is performed, and by default a local projector as described in H.-J. Werner, J. Chem. Phys. (submitted, 2008) is used. This yields very similar energies as the DF-LMP2-F12/3*A(D,NOX) method with full domains at lower cost, and the method can be applied to quite large molecules. The local projector can also be used with other ansätze, by specifying, e.g. `ANSATZ=3C(loc)`, but in this case the local projection is only simulated and does not save any time. Whether or not the local projector is used depends on the option `USEPAO`, see below.

Special options for these local variants are (local RI works only with ansatz 3*A):

`PAIRS` Specifies which pairs to be treated by R12 or F12 (`STRONG|CLOSE|WEAK|ALL`; pairs up to the given level are included). The default is `ALL`. Note that even with `ALL` very distant pairs are neglected if these are neglected in the LMP2 as well.

`USEVRT` — If zero, the $1-pp+po+op-p'o-op'$ form of the projector is used, and local RI approximations apply to p and p'. If set to 1, the $1+oo-vv-p'o-op'$ form of the projector is used; any local RI approximation then applies only to the RI contribution p'.

`USEPAO` — If `USEPAO=1` use pair-specific local projectors instead of vv. This is the default if either the ansatz contains '(LOC)' or if domain approximations are made in the LMP2 (i.e., `DOMSEL<1` is explicitly specified). Otherwise the default is `USEPAO=0`. `USEPAO=1` automatically implies `USEVRT=1`, i.e. local RI approximations only affect the RI contributions p'. Furthermore, if `USEPAO=1` is specified and `DOMSEL` is not given, default domains are assumed in the LMP2. If `USEPAO=0`, full domains (`DOMSEL=1`) will be used.

`FULLAO` — if `USEVRT=0` and `FULLAO=1`, local RI approximations only apply to the RI contributions p'. This should give the same results as `USEVRT=1` (only applies if `USEPAO=0`).

`DEBUG` — Parameter for debug print

`LOCFIT_F12` — If set to one, use local fitting. Default is no local fitting (`LOCFIT_F12=0`)

`LOCFIT_R12` — Alias for `LOCFIT_F12`. Local fitting is not recommended in R12 calculations.

`FITDOM` — Determine how the base fitting domains are determined (only applies if `LOCFIT_F12=1`):
0: Fitdomains based on united operator domains;
1: Fitdomains based in orbital domains (default);
2: Fitdomains based on united pair domains using strong pairs;
3: Fitdomains based on united pair domains using strong, close and weak pairs. Note: This is the only option implemented in the DF-LMP2 program. Therefore, the DF-LMP2 and DF-LMP2-F12 programs might give slightly different results if default values are used.

`RDOMAUX` — Distance criterion for density fitting domain extensions in case of local fitting. The default depends on `FITDOM`.

`IDOMAUX` — Connectivity criterion for density fitting domain extensions in case of local fitting.

`RAODOM` — Distance criterion for RI domain extensions. Zero means full RI basis (default). If `USEPAO=1` or `USEVRT=1` or `FULLAO=1` a value of 5 bohr is recommended. In other cases the local RI domains must be very large (`RAODOM>12`) and the use of local RI approximations is not recommended.

`IAODOM` — Connectivity criterion for RI domain extensions. Zero means full RI basis (default). Values greater or equal to 2 should lead to sufficiently accurate results, provided the local projector (`USEPAO=1`) is used.

`THRAO` — Screening threshold for coulomb integrals in the AO or RI basis.

`THRF12` — Screening threshold for F12 integrals.

`THRMO` — Screening threshold for half transformed integrals.

`THRPROD` — Product screening threshold in the first half transformation.

`NOMP2` — If set to 1, only the F12 calculation is performed, and the LMP2 is skipped. This is sometimes useful if full domains are used, since the iterative LMP2 then causes a big overhead and needs a lot of memory.

It is then more efficient to do the a DF-MP2 calculation separately and compute the total energy as the sum of the DF-MP2 energy and the F12 energy

Further options for density fitting are described in section 15, and further options to choose the ansatz in section 31.7.

Typical inputs for calculations with local approximations are:

```
!parameters for local density fitting:
DFIT,LOCFIT_F12=1,FITDOM_MP2=1,IDOMAUX_MP2=3,DSCREEN=1
!LMP2-F12(loc) with local RI:
{DF-LMP2-F12,ANSATZ=3*A(LOC),DOMSEL=0.985,RAODOM=5,PAIRS=WEAK}
```

This would perform a local MP2 with a Boughton-Pulay domain completeness criterion of 0.985. In the F12 part, distant pairs are not included (`PAIRS=WEAK`) and the local projector is used (`USEPAO=1`, default). Local density fitting and local RI approximations are used.

A corresponding non-local calculation (still using localized orbitals and the diagonal ansatz) would be

```
{DF-LMP2-F12,ANSATZ=3*A(LOC),DOMSEL=1.0,USEVRT=1,NOMP2=1}
ecorr_F12=ef12

{DF-MP2}
ecorr_MP2=energy-energr                !mp2 correlation energy
ecorr_MP2_F12=ecorr_MP2+ecorr_F12      !total correlation energy
```

Note: The use of local DF and RI domains is still experimental and should be use with care!

31.12 Variables set by the F12 programs

The following variables are set by the F12 programs:

`ENERGR`	Reference energy. This includes the perturbative CABS singles correction if computed.
`ENERGY`	Total energy of the requested method (including the F12 and singles corrections). `ENERGY(1)` and `ENERGY(2)` hold the F12A and F12B values, respectively (if both are computed).
`ENERGC`	Total CCSD-F12 energies in ccsd-f12 or uccsd-f12 calculations. `ENERGC(1)` and `ENERGC(2)` hold the F12A and F12B values, respectively (if both are computed). The difference of `ENERGY` and `ENERGC` is the triples energy contribution.
`ENERGT`	Triples energy contribution. This is a vector. The corresponding methods are stored in `METHODT` (strings).
`EMP2`	Total MP2 energy (excluding F12 correction, but including the singles correction).
`EMP2_SCS`	Total SCS-MP2 energy (excluding F12 correction, but including the singles correction).

EMP2_SING	Singlet MP2 correlation energy (excluding F12 correction).
EMP2_TRIP	Triplet MP2 correlation energy (excluding F12 correction).
EMP2_STRONG	Strong pair contribution to the LMP2 correlation energy (where applicable).
EMP2_CLOSE	Close pair contribution to the LMP2 correlation energy (where applicable).
EMP2_WEAK	Weak pair contribution to the LMP2 correlation energy (where applicable).
EMP2_DIST	Distant pair F12 contribution to the LMP2-F12 correlation energy (where applicable).
EF12	F12 contribution to the MP2-F12 correlation energy for the requested ansatz.
EF12S	F12 contribution to the MP2-F12 correlation energy using EBC approximation for the requested ansatz.
EF12D	F12 contribution to the MP2-F12 correlation energy using EBC approximation and diagonal (DX) approximation for the requested ansatz.
EF12_SING	Singlet F12 contribution to the MP2-F12 correlation energy for the requested ansatz.
EF12_TRIP	Triplet F12 contribution to the MP2-F12 correlation energy for the requested ansatz.
EF12_STRONG	Strong pair F12 contribution to the LMP2-F12 correlation energy (where applicable).
EF12_CLOSE	Close pair F12 contribution to the LMP2-F12 correlation energy (where applicable).
EF12_WEAK	Weak pair F12 contribution to the LMP2-F12 correlation energy (where applicable).
EF12_DIST	Distant pair F12 contribution to the LMP2-F12 correlation energy (where applicable).
EF12_SCS	F12 contribution to the MP2-F12 correlation energy for the requested ansatz.
EF12_SINGLES	Total CABS singles contribution (in closed-shell case equal to EF12_RHFRELAX).
EF12_RHFRELAX	CABS singles correction of the reference energy (only the spin-free contribution is used).
ANSATZ	The requested ansatz (string variable)

Variables corresponding to EF12_* exist also for EF12S_* and EF12D_*.

In case of doubt or problems, try in a test calculation

```
SHOW,ENERG*,EMP2*,EF12*
```

This should show all relevant variables that exist. Note that system variables are internally stored with an underscore as a prefix, and this may be shown by the SHOW command. The variables can be accessed with or without underscore (but if the user defines a variable with the same name then the underscore is needed to access the system variable and not the user variable).

32 THE FULL CI PROGRAM

This module is the determinant full CI program, as described in

P.J. Knowles and N.C. Handy, Chem. Phys. Letters 111 (1984) 315,
P.J. Knowles and N.C. Handy, Comp. Phys. Commun. 54 (1989) 75.

Published work resulting from the use of this program should cite these references.

The program in normal use finds the lowest eigenvector of the complete CI hamiltonian matrix; more sophisticated use is possible, but not documented here. The program is interfaced to free standing versions such as supplied in the CPC program library by use of the `DUMP` option.

The program is called with the command `FCI`.

32.1 Defining the orbitals

`ORBIT`,*name.file*;

name.file specifies the record from which orbitals are read. The default is the set of orbitals from the last SCF, MCSCF or CI calculation.

32.2 Occupied orbitals

`OCC`,$n_1, n_2, \ldots, n_8$;

n_i specifies numbers of occupied orbitals (including `CORE`) in irreducible representation number i. If not given, the default is the complete basis set.

32.3 Frozen-core orbitals

`CORE`,$n_1, n_2, \ldots, n_8$;

n_i is the number of frozen-core orbitals in irrep number i. These orbitals are doubly occupied in all configurations, i.e., not correlated. If no `CORE` card is given, the program uses the same core orbitals as the last CI calculation; if there was none, then the atomic inner shells are taken as core. To avoid this behaviour and correlate all electrons, specify

`CORE`

32.4 Defining the state symmetry

The number of electrons and the total symmetry of the wavefunction are specified on the `WF` card:

`WF`,*elec,sym,spin*

where

elec:	is the number of electrons
sym:	is the number of the irreducible representation
spin:	defines the spin symmetry, *spin*= $2S$ (singlet=0, doublet=1, triplet=2, etc.)

32.5 Printing options

PRINT,*code,value*;

Print options. Generally, the value determines how much intermediate information is printed. *value*=-1 means no print (default for all codes). if *value* is omitted, it is taken as zero, which is usually appropriate. Specification of higher values will generate more output. The following codes are allowed:

ORBITAL	Print molecular orbitals
INTEGRAL	Print integrals
TIMING	Print extra timing information
DIAGONAL	Print diagonal elements of Hamiltonian
HAMILTONIAN	Print much intermediate information

32.6 Interface to other programs

DUMP;

causes the FCI diagonalization to be bypassed, with input information and transformed integrals being written to a formatted file FCIDUMP. The format is as described in Comp. Phys. Commun. 54 (1989) 75.

33 SYMMETRY-ADAPTED INTERMOLECULAR PERTURBATION THEORY

33.1 Introduction

The SAPT (symmetry-adapted intermolecular perturbation theory) program calculates the total interaction energy between closed-shell molecules as a sum of individual first and second order interaction terms, namely electrostatic $E^{(1)}_{\text{pol}}$, induction $E^{(2)}_{\text{ind}}$ and dispersion $E^{(2)}_{\text{disp}}$ accompanied by their respective exchange counterparts ($E^{(1)}_{\text{exch}}$, $E^{(2)}_{\text{exch-ind}}$ and $E^{(2)}_{\text{exch-disp}}$). The latter ones arise due to electron exchange between the monomers when the molecules are close to each other and are sometimes denoted as Pauli repulsion. Since all above terms are accessible through density matrices and static and dynamic density-density response functions of the monomers, in principle (see section 33.4) no calculation of the dimer wave function is required. Therefore SAPT is free from the basis set superposition error which occurs in the supermolecular approach.

References:

General Symmetry-adapted perturbation theory and many-body SAPT:
[1] B. Jeziorski, R. Moszynski and K. Szalewicz, Chem. Rev. **94**, 1887. (1994).

DFT-SAPT:
[2] G. Jansen and A. Heßelmann, J. Phys. Chem. A **105**, 646 (2001).
[3] A. Heßelmann and G. Jansen, Chem. Phys. Lett. **357**, 464 (2002).
[4] A. Heßelmann and G. Jansen, Chem. Phys. Lett. **362**, 319 (2002).
[5] A. Heßelmann and G. Jansen, Chem. Phys. Lett. **367**, 778 (2003).

[6] A. Heßelmann and G. Jansen, Phys. Chem. Chem. Phys. **5**, 5010 (2003).

Density fitting DFT-SAPT (DF-DFT-SAPT):
[7] A. Heßelmann, G. Jansen and M. Schütz, J. Chem. Phys. **122**, 014103 (2005).

(See also:
K. Szalewicz, K. Patkowski and B. Jeziorski, Struct. Bond **116**, 43 (2005)
and references therein for a related approach to DFT-SAPT termed SAPT(DFT))

33.2 First example

A typical input for SAPT has the following form:

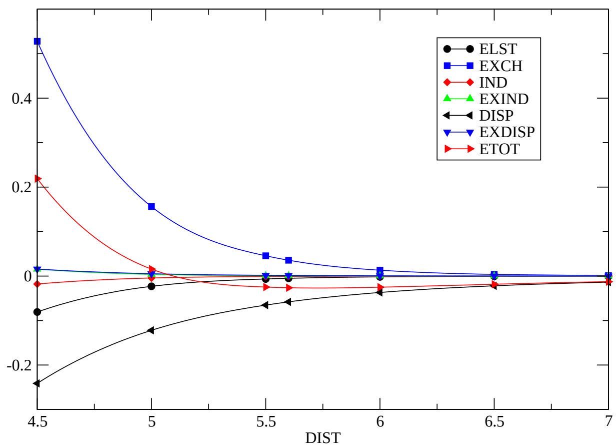
```
r=5.6
geometry={nosym; he1; he2,he1,r}
basis=avqz

!wf records
ca=2101.2
cb=2102.2

!monomer A
dummy,he2
{hf; save,$ca}
sapt;monomerA

!monomer B
dummy,he1
{hf; start,atdens; save,$cb}
sapt;monomerB

!interaction contributions
sapt;intermol,ca=$ca,cb=$cb
```

Here the `sapt;monomerA/B` store some informations about the two monomers which are needed in the subsequent SAPT calculation invoked by `sapt;intermol`. The individual interaction energy terms are stored (in millihartree) in distinct variables and may be collected in arrays for producing potential energy surfaces. For example the input

```
geometry={nosym; he1; he2,he1,r}
basis=avtz

!wf records
ca=2101.2
cb=2102.2

!distances
dist=[4.5,5.0,5.5,5.6,6.0,6.5,7.0]

do i=1,#dist
```

```
   r=dist(i)

   !monomer A
   dummy,he2
   {hf; save,$ca}
   sapt;monomerA

   !monomer B
   dummy,he1
   {hf; start,atdens; save,$cb}
   sapt;monomerB

   !interaction contributions
   sapt;intermol,ca=$ca,cb=$cb

   elst(i)=E1pol;   exch(i)=E1ex
   ind(i)=E2ind;    exind(i)=E2exind
   disp(i)=E2disp;  exdisp(i)=E2exdisp
   etot(i)=E12tot

   data,truncate,$ca
enddo

{table,dist,elst,exch,ind,exind,disp,exdisp,etot
ftyp,d,d,d,d,d,d,d,d
plot}
```

yields the plot



Currently SAPT only accepts single-determinant wave functions for the monomers, i.e. from Hartree-Fock or Kohn-Sham DFT (see next section) calculations. This means that if Hartree-Fock wave functions are used for monomer, the following quantity is obtained (zero in superscript denotes that no intramonomer correlation is accounted for) [1].

$$E_{\text{SAPT}} = E^{(10)}_{\text{pol}} + E^{(10)}_{\text{exch}} + E^{(20)}_{\text{ind,resp.}} + E^{(20)}_{\text{exch-ind,resp.}} + E^{(20)}_{\text{disp}} + E^{(20)}_{\text{exch-disp}}$$

No point group symmetry can be exploited in a SAPT calculation.

33.3 DFT-SAPT

It is of crucial importance to account for the *intra*molecular correlation effects of the individual SAPT terms since Hartree-Fock theory often yields poor first- and second-order electrostatic properties. While this can be done using many-body perturbation theory [1] (in a double perturbation theory ansatz) a more efficient way is to use static and time-dependent DFT theory. This variant of SAPT, termed as DFT-SAPT [2-6], has in contrast to Hartree-Fock-SAPT the appealing feature that the polarisation terms ($E^{(1)}_{\rm pol}$, $E^{(2)}_{\rm ind}$, $E^{(2)}_{\rm disp}$) are potentially exact, i.e. they come out exactly if the exact exchange-correlation (xc) potential and the exact (frequency-dependent) xc response kernel of the monomers were known. On the other hand, this does not hold for the exchange terms since Kohn-Sham theory can at best give a good approximation to the exact density matrix of a many-body system. It has been shown [6] that this is indeed the the case and therefore DFT-SAPT has the potential to produce highly accurate interaction energies comparable to high-level supermolecular many-body perturbation or coupled cluster theory. However, in order to achieve this accuracy, it is of crucial importance to correct the wrong asymptotic behaviour of the xc potential in current DFT functionals [3-5]. This can be done by using e.g.:

```
{ks,lda; asymp,<shift>}
```

which activates the gradient-regulated asymptotic correction approach of Grüning *et al.* (J. Chem. Phys. **114**, 652 (2001)) for the respective monomer calculation. The user has to supply a `shift` parameter ($\Delta_{\rm xc}$) for the bulk potential which should approximate the difference between the HOMO energy ($\varepsilon_{\rm HOMO}$) obtained from the respective standard Kohn-Sham calculation and the (negative) ionisation potential of the monomer (IP):

$$\Delta_{\rm xc} = \varepsilon_{\rm HOMO} - (-{\rm IP}) \tag{63}$$

This method accounts for the derivative discontinuity of the exact xc-potential and that is missing in approximate ones. Note that this needs to be done only once for each system. (See also section 33.7.2 for an explicit example).

Concerning the more technical parameters in the DFT monomer calculations it is recommended to use lower convergence thresholds and larger intergration grids compared to standard Kohn-Sham calculations.

33.4 High order terms

It has been found that third and higher-order terms become quite important if one or both monomers are polar. As no higher than second-order terms are currently implemented in SAPT, one may use a non-correlated estimation of those terms by using supermolecular Hartree-Fock (see e.g. [7]). This can be done by adapting the following template:

```
!dimer
hf
edm=energy

!monomer A
dummy,<monomer2>
{hf; save,$ca}
ema=energy
```

```
sapt;monomerA

!monomer B
dummy,<monomer1>
{hf; start,atdens; save,$cb}
emb=energy
sapt;monomerB

!interaction contributions
sapt,sapt_level=2;intermol,ca=$ca,cb=$cb

esup=(edm-ema-emb)*1000. mH
dHF=esup-e1pol-e1ex-e2ind-e2exind
```

which stores the resulting δ(HF) term in `dHF`.

33.5 Density fitting

In order to be able to study interactions between extended monomers one can use density fitting to approximate the integrals in SAPT [7]. For this one may use the input:

```
{sapt;intermol,ca=$ca,cb=$cb,fitlevel=3
dfit,basis_coul=jkfit,basis_exch=jkfit,basis_mp2=mp2fit,cfit_scf=3}
```

with in the basis section defined `jkfit` and `mp2fit` fitting basis sets (see section 15).

Currently only the ALDA xc-kernel is implemented for the case `SAPT_LEVEL`=3 and `SAPT_FITLEVEL`=3. This means that a corresponding SAPT calculation would be uncompatible with hybrid-DFT monomer orbitals/orbital energies. Therefore it is recommended to use nonhybrid functionals in the case the dispersion/exchange-dispersion energy terms are requested in a DF-DFT-SAPT run. Another possibility is to localise the xc-potential via, e.g., the OEP method (see also example in section 33.7.3).

33.6 SAPT with ECP's

If effective core potentials (ECP's) are used in the monomer calculations, it is important to add the δ(HF) term to the SAPT interaction energy (see K. Patkowski, K. Szalewicz, *J. Chem. Phys.* **127** (2007) 164103). For examples for the calculation of δ(HF) see sections 33.4 and 33.7.

33.7 Examples

33.7.1 HF-SAPT calculation of the H_2O dimer using the δ(HF) correction

```
gthresh,energy=1.d-8,orbital=1.d-8,grid=1.d-8
symmetry,nosym
orient,noorient
GEOMTERY={
1,O1,,0.00000000,0.00000000,0.00000000
2,H1,,0.00000000,0.00000000,1.83606000
3,H2,,1.77604000,0.00000000,-0.4656040
4,O2,,-0.6605540,0.00000000,5.54064000
5,H3,,-1.6582100,-1.4536300,6.05324000
6,H4,,-1.6582100,1.45363000,6.05324000
}
basis=avdz

!sapt files
ca=2101.2
cb=2102.2

!dimer
hf
edm=energy

!monomer A
dummy,o2,h3,h4
{hf; save,$ca}
ema=energy
sapt;monomerA

!monomer B
dummy,o1,h1,h2
{hf; start,atdens; save,$cb}
emb=energy
sapt;monomerB

!interaction contributions
sapt,SAPT_LEVEL=3;intermol,ca=$ca,cb=$cb,icpks=1

!HF supermolecular interaction energy and delta(HF) contribution
eint_hf=(edm-ema-emb)*1000 mH
delta_hf=eint_hf-e1pol-e1ex-e2ind-e2exind

!add E2disp + E2exch-disp to HF interaction energy
eint_sapt=eint_hf+e2disp+e2exdisp
```

http://www.molpro.net/info/current/examples/h2odimer_sapt_hf.com

33.7.2 DFT-SAPT calculation of the NeAr dimer using the δ(HF) correction

```
gthresh,energy=1.d-8,orbital=1.d-8,grid=1.d-8
symmetry,nosym
orient,noorient
geometry={
Ne,,0.0,0.0,0.0
Ar,,0.0,0.0,6.5}
basis=avtz

!=========delta(HF) contribution for higher order interaction terms====
!sapt files
ca=2101.2
cb=2102.2

!dimer
hf
edm=energy

!monomer A
dummy,ar
{hf; save,$ca}
ema=energy
sapt;monomerA

!monomer B
dummy,ne
{hf; start,atdens; save,$cb}
emb=energy
sapt;monomerB

!interaction contributions
sapt,SAPT_LEVEL=2;intermol,ca=$ca,cb=$cb,icpks=1

!calculate high-order terms by subtracting 1st+2nd order energies
eint_hf=(edm-ema-emb)*1000 mH
delta_hf=eint_hf-e1pol-e1ex-e2ind-e2exind


!=========DFT-SAPT at second order intermol. perturbation theory====
!sapt files
ca=2103.2
cb=2104.2

!shifts for asymptotic correction to xc potential
eps_homo_pbe0_ar=-0.440936      !HOMO(Ar)/PBE0 functional
eps_homo_pbe0_ne=-0.589207      !HOMO(Ne)/PBE0
ip_ar=0.5792                    !exp. ionisation potential
ip_ne=0.7925                    !exp. ionisation potential
shift_ar=ip_ar+eps_homo_pbe0_ar !shift for bulk xc potential (Ar)
shift_ne=ip_ne+eps_homo_pbe0_ne !shift for bulk xc potential (Ne)

!monomer A
dummy,ar
{ks,pbe0; asymp,shift_ne; save,$ca}
sapt;monomerA

!monomer B
dummy,ne
{ks,pbe0; start,atdens; asymp,shift_ar; save,$cb}
sapt;monomerB

!interaction contributions
sapt;intermol,ca=$ca,cb=$cb,icpks=0

!add high-order approximation to obtain the total interaction energy
eint_dftsapt=e12tot+delta_hf
```

http://www.molpro.net/info/current/examples/near_sapt_acdft.com

33.7.3 DF-DFT-SAPT calculation of the NeAr dimer using the δ(HF) correction

```
gdirect; gthresh,energy=1.d-8,orbital=1.d-8,grid=1.d-8
symmetry,nosym
orient,noorient
geometry={
Ne,,0.0,0.0,0.0
Ar,,0.0,0.0,6.5}
basis={
set,orbital; default,avtz         !for orbitals
set,jkfit;   default,avtz/jkfit   !for JK integrals
set,mp2fit;  default,avtz/mp2fit  !for E2disp/E2exch-disp
set,dflhf;   default,avtz/jkfit   !for LHF
}

!=========delta(HF) contribution for higher order interaction terms====
ca=2101.2; cb=2102.2 !sapt files

!dimer
{df-hf,basis=jkfit,locorb=0}
edm=energy

!monomer A
dummy,ar
{df-hf,basis=jkfit,locorb=0; save,$ca}
ema=energy; sapt;monomerA

!monomer B
dummy,ne
{df-hf,basis=jkfit,locorb=0; save,$cb}
emb=energy; sapt;monomerB

!interaction contributions
{sapt,SAPT_LEVEL=2;intermol,ca=$ca,cb=$cb,icpks=1,fitlevel=3
dfit,basis_coul=jkfit,basis_exch=jkfit,cfit_scf=3}

!calculate high-order terms by subtracting 1st+2nd order energies
eint_hf=(edm-ema-emb)*1000 mH
delta_hf=eint_hf-e1pol-e1ex-e2ind-e2exind

!=========DFT-SAPT at second order intermol. perturbation theory====
ca=2103.2; cb=2104.2 !sapt files;

!shifts for asymptotic correction to xc potential
eps_homo_pbe0_ar=-0.440936      !HOMO(Ar)/PBE0 functional
eps_homo_pbe0_ne=-0.589207      !HOMO(Ne)/PBE0
ip_ar=0.5792                    !exp. ionisation potential
ip_ne=0.7925                    !exp. ionisation potential
shift_ar=ip_ar+eps_homo_pbe0_ar !shift for bulk xc potential (Ar)
shift_ne=ip_ne+eps_homo_pbe0_ne !shift for bulk xc potential (Ne)

!monomer A, perform LPBE0AC calculation
dummy,ar
{df-ks,pbex,pw91c,lhf; dftfac,0.75,1.0,0.25; asymp,shift_ne; save,$ca}
sapt;monomerA

!monomer B, perform LPBE0AC calculation
dummy,ne
{df-ks,pbex,pw91c,lhf; dftfac,0.75,1.0,0.25; start,atdens; asymp,shift_ar; save,$cb}
sapt;monomerB

!interaction contributions
{sapt,SAPT_LEVEL=3;intermol,ca=$ca,cb=$cb,icpks=0,fitlevel=3,nlexfac=0.0
dfit,basis_coul=jkfit,basis_exch=jkfit,cfit_scf=3}

!add high-order approximation to obtain the total interaction energy
eint_dftsapt=e12tot+delta_hf
```

http:

33.8 Options

SAPT_LEVEL	Set to 1 for first-order terms ($E^{(1)}_{\mathrm{pol}}$ and $E^{(1)}_{\mathrm{exch}}$), to 2 for additional second order (exchange-)induction terms ($E^{(2)}_{\mathrm{ind}}$ and $E^{(2)}_{\mathrm{exch-ind}}$) and 3 for all first- and second-order terms (including then also $E^{(2)}_{\mathrm{disp}}$ and $E^{(2)}_{\mathrm{exch-disp}}$) (default 3)
SAPT_FITLEVEL	Level of density fitting approximations in SAPT which can have values 0 to 3 (default 0)
SAPT_ICPKS	Switch between iterative (=1) and non-iterative (=0) solution of coupled-perturbed Kohn-Sham equations (default 0)
SAPT_CPKSTHR	Threshold for density matrix convergency in the coupled-perturbed Kohn-Sham program (default 1.d-6).
SAPT_CPKMAXIT	Maximum number of iterations in the coupled-perturbed Kohn-Sham program (default 50).
SAPT_FROZENA	Number of frozen electrons in the response calculations for monomer A (default 0)

The following parameters are of importance if SAPT_FITLEVEL>0:

SAPT_NFRQ_DISP	Number of frequencies for the Casimir-Polder integration (default 12)
SAPT_NORM_DISP	Norm for the density fitting which can be either COULOMB or NATURAL (default COULOMB)
SAPT_DISP_N4	Can speedup the calculation of the dispersion energy by N^4 scaling (default 1)
THR_XCKERN	Density threshold for the xc kernel matrix elements (default 1.d-8)
FIT_XCKERN	Fit both sides of the xc kernel (default 0)
SAPT_DISK	If 0 write all dimer amplitudes to file, if 1 write 3-index response propagators to file and if 2 write 3-index response propagators compressed to file. The latter two variants save disk space but need more CPU time to compute $E^{(2)}_{\mathrm{exch-disp}}$ (default 0)
COMPRESS_THR	If SAPT_DISK=2 this value determines the compression cutoff (default 1d-12)
UNCOUPLED	If SAPT_DISK>0 calculate also uncoupled (exchange-)dispersion energies (default false)
THRAO	Threshold for AO 3-index integrals (default 1.d-12)
THRMO	Threshold for MO 3-index integrals (default 1.d-8)
THROV	Threshold for AO 2-index integrals (default 1.d-10)
THRPROD	Product threshold for first half transformation (default 1.d-8)
THRSW	Threshold for Schwarz screening (default 1.d-5)
C6	Calculate dispersion coefficients for the two monomers (Note that the full dimer basis set is used in each case and that a closer distance of the monomers can perturb the result).
XCKERN_NBLOCK	number of grid points treated together as a block for $(\mathrm{aux}\|f_{\mathrm{xc}}\|\mathrm{occ}\times\mathrm{virt})$ integrals (default 128)

`CFAC`	factor for VWN correlation in ALDA xc-kernel (default 1.d0)

The last threshold values for the 2- and 3-index integrals should not be set higher in density fitting calculations as this can cause lower accuracies in the interaction terms. In addition SAPT knows the following subcommands:

`MONOMERA`	Stores informations (like number of electrons, etc.) about previous monomer A calculation
`MONOMERB`	See above
`INTERMOL`	Starts the SAPT calculation

`INTERMOL` may have the following subkeywords:

`CA`	Record number of wave function for monomer A (always needed)
`CB`	Record number of wave function for monomer B (always needed)
`SAPTLEVEL`	See above
`FITLEVEL`	See above
`ICPKS`	See above
`FROZA`	See above
`FROZB`	See above
`NLEXFAC`	Amount of nonlocal exact exchange in hybrid DFT-SAPT calculations
`CPKSTHR`	Threshold for density matrix convergency in the coupled-perturbed Kohn-Sham program.
`CPKSMAXIT`	Maximum number of iterations in the coupled-perturbed Kohn-Sham program.

34 PROPERTIES AND EXPECTATION VALUES

34.1 The property program

The property program allows the evaluation of one-electron operators and expectation values. Normally, the operators are computed automatically when using the global `GEXPEC` directive (see section 6.13) or the `EXPEC` or `TRAN` commands in the SCF, MCSCF, and CI programs. The explicit use of the property program is only necessary in the rare case that the user is interested in an orbital analysis of the properties.

34.1.1 Calling the property program (`PROPERTY`)

`PROPERTY`

invokes the property program.

34.1.2 Expectation values (DENSITY)

DENSITY [,*record.file*] [,*specifications*]

If this card is present, the density matrix will be read from record *record.file* and property expectation values will be calculated. If the specification *record.file* is omitted, the last dump record is used. Density matrices for specific states can be selected using *specifications*, as explained in section 4.11. Note that the density matrices are stored in the same record as the orbitals.

34.1.3 Orbital analysis (ORBITAL)

ORBITAL [,*record.file*] [,*specifications*]

If this card is present, the orbitals are read from record *record.file* and an orbital analysis of the expectation values is printed (the density matrix must also be provided!). If *record.file* is omitted, the last dump record is used. This is only meaningful for diagonal density matrices (SCF or natural orbitals). Natural orbitals for specific states can be selected using *specifications*, as explained in section 4.11.

34.1.4 Specification of one-electron operators

The required operators are specified by code words. Optionally, the geometry or the nuclear centre at which the operator is computed can be specified.

For each operator, an input card of the following form is required:

code,centre,x,y,z,,factor

code specifies the property. The available operators are given in section 6.13.

The other parameters have the following meaning:

centre	row number of Z–matrix or atomic symbol defining the centre at which property shall be calculated; if *centre*$\neq 0$ you need not read in coordinates.
x,y,z	cartesian coordinates of the point (only if *centre*=0).
factor	the operator is multiplied by this factor. The default is *factor*=1 except for REL. In this cases proper factors for relativistic corrections are used unless *factor* is given. The two commas before factor are needed to preserve compatibility with Molpro96.

34.1.5 Printing options

PRINT,*print*

This card is used to control output, mainly for debugging purposes.

print$= 0$	no test output (default)
print> 0	operators are printed.

34.1.6 Examples

The following example computes the dipole quadrupole moments of water and prints an orbital analysis. By default, the origin is at the centre of mass, and this is taken as origin for the quadrupole moments.

```
***,h2o properties
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
hf                                    !do scf calculation
property                              !call property program
orbital                               !read scf orbitals
density                               !read scf density matrix
dm                                    !compute dipole moments and print orbital contributions
qm                                    !compute quadrupole moments and print orbital contributi
{multi;state,2;dm                     !do full-valence CASSCF
natorb,state=1.1                      !compute natural orbitals for state 1.1
natorb,state=2.1}                     !compute natural orbitals for state 2.1

{property                             !call property program
orbital,state=1.1                     !read casscf natural orbitals for state 1.1
density,state=1.1                     !read casscf density matrix for state 1.1
dm                                    !compute dipole moments and print orbital contributions
qm}                                    !compute quadrupole moments and print orbital contribut

{property                             !call property program
orbital,state=2.1                     !read casscf natural orbitals for state 2.1
density,state=2.1                     !read casscf density matrix for state 2.1
dm                                    !compute dipole moments and print orbital contributions
qm}                                   !compute quadrupole moments and print orbital contributi
```

http://www.molpro.net/info/current/examples/h2o_property.com

Alternatively, the dipole and quadrupole moments can be computed directly in the SCF and MCSCF programs, but in this case no orbital contributions are printed:

```
***,h2o properties
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
gexpec,dm,qm                          !global request of dipole and quadrupole moments
hf                                    !do scf calculation
{multi;state,2                        !do full-valence CASSCF
natorb,state=1.1                      !compute natural orbitals for state 1.1
natorb,state=2.1}                     !compute natural orbitals for state 2.1
```

http://www.molpro.net/info/current/examples/h2o_gexpec1.com

34.2 Distributed multipole analysis

Any density matrix can be analysed using the distributed multipole analysis described by Stone, Chem. Phys. Letters (1981), 83, 233. The multipole moments arising from the overlap of each pair of primitives are calculated with respect to the overlap centre, and then shifted to the nearest of a number of *multipole sites*. By default these comprise all atoms specified in the integral input. However the list of multipole sites can be modified by deleting and/or adding sites, and also by restricting the rank of multipole which may be transferred to any given site. The atomic charges are stored in the MOLPRO variable ATCHARGE. The i'th element in ATCHARGE corresponds to the i'th row of the Z-matrix input.

Options may appear in any order, except DENSITY, which must be first if given.

The present version does not allow generally contracted AO basis sets.

34.2.1 Calling the DMA program (DMA)

DMA;

This command initializes the DMA program.

34.2.2 Specifying the density matrix (DENSITY)

DENSITY,*record.file* [,*specifications*]

The density matrix to be analysed is that found in record *record* on file *file*. If omitted, *record.file* defaults to current orbital record. If specified, DENSITY must appear first in the input. Density matrices for specific states can be selected using *specifications*, as explained in section 4.11.

34.2.3 Linear molecules (LINEAR, GENERAL)

GENERAL;

(default) invokes the normal program, which copes with any geometry.

LINEAR

invokes a faster program which can be used when all the atoms are arranged parallel to the z-axis and only the $m = 0$ components of the multipoles are required.

34.2.4 Maximum rank of multipoles (LIMIT)

LIMIT,*name,lmax*;

lmax is the highest rank of multipole that is to be calculated by the program. Default (and maximum) is 10 for the general program and 20 for the linear one. If *name* is specified, the limit applies only to multipole site *name*.

34.2.5 Omitting nuclear contributions (NONUCLEAR)

NONUCLEAR

The nuclear contributions to properties are not to be evaluated.

34.2.6 Specification of multipole sites (ADD, DELETE)

ADD,*name,x,y,z,lmax,radius*;

Add a new site at (x, y, z) with the *name* specified. The multipole rank is limited to *lmax* if a value is specified, otherwise the value of lmax specified by the LIMIT directive is used.

No account is taken of symmetry; every site in a symmetry-equivalent set must be specified explicitly. The *radius* of the site may also be specified (default 1.0).

`DELETE`,*name*

Delete all atoms with the *name* given from consideration as a multipole site. Note that original atoms from the integral program have names $1,2,3,\ldots$ as printed in integral output. `DELETE,ALL` deletes all atoms and gives the multipoles with respect to the origin only.

34.2.7 Defining the radius of multipole sites (`RADIUS`)

`RADIUS`,*name,r*;

Assign radius r to all sites with the *name* given. The program moves multipoles at an overlap centre P to the site S for which the value of $|P-S|/r(S)$ is smallest. In the absence of a `RADIUS` directive, all sites are given radius 1.

34.2.8 Notes and references

The multipoles produced by this analysis are given in their spherical harmonic definitions. Explicit formulae for translating between the cartesian and spherical harmonic definitions of the multipole moments are given in, *Explicit formulae for the electrostatic energy, forces and torques between a pair of molecules of arbitrary symmetry*, S. L. Price, A. J. Stone, and M. Alderton, Molec. Phys., 52, 987 (1984).

For examples of the use of `DMA` analysis see, Price and Stone, Chem. Phys. Lett., 98, 419 (1983); Buckingham and Fowler, J. Chem. Phys., 79, 6426 (1983).

34.2.9 Examples

The following input calculates SCF multipole moments for water.

```
***,h2o distributed multipole analysis
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
basis=6-311g**
hf                                    !do scf calculation
{dma;limit,,4}                        !results for total multipoles are
```

http://www.molpro.net/info/current/examples/h2o_dma.com

34.3 Mulliken population analysis

34.3.1 Calling the population analysis program (`POP`)

`POP`;

Invokes Mulliken analysis program, which analyses any density matrix into its contributions from s,p,d,f... basis functions on each atom. The density matrix is taken from the last dump record, unless overridden with the `DENSITY` card. The subcommands may be abbreviated by the first four characters. The atomic charges are stored in the MOLPRO variable `ATCHARGE`. The i'th element in `ATCHARGE` corresponds to the i'th row of the Z-matrix input.

34.3.2 Defining the density matrix (DENSITY)

DENSITY,*record.file* [,*specifications*]

Take density matrix to be analysed from record *record* on file *file*. Density matrices for specific states can be selected using *specifications*, as explained in section 4.11. Note that the density matrices are stored in the same record as the orbitals.

34.3.3 Populations of basis functions (INDIVIDUAL)

INDIVIDUAL;

34.3.4 Example

```
***,h2o population analysis
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
basis=6-311g**
hf                                    !do scf calculation
pop;                                  !Mulliken population analysis using mcscf density
individual                            !give occupations of individual basis functions
```

http://www.molpro.net/info/current/examples/h2o_pop.com

If specified, the Mulliken populations of each individual basis function are printed.

34.4 Natural Bond Orbital Analysis

34.4.1 Calling the Natural Bond Orbital analysis program (NBO)

NBO,[WITH_CORE=*core_option*],[LEVEL=*level*],[KEEP_WBI=*wbi_option*];

The Natural Bond Orbital Analysis of Weinhold and coworkers (J. Chem. Phys. 83 (1985) 735, J. Chem. Phys. 83 (1985) 1736 and J. Chem. Phys. 78 (1983) 4066) can be called by the use of the NBO card. It reads from a density or orbital record, and performs the necessary transformations to Natural Atomic Orbitals (NAO), Natural Bond Orbitals (NBO) and Natural Localized Molecular Orbitals (NLMO). The latter can also be saved to a record and later used in local correlation treatments (cf. Section 29). By default, the full orbital space is used. The core orbitals can, however, be left out of the procedure if *core_option*=0.

One can choose to truncate the transformation series (e.g., only compute the NAO orbitals), with help of the LEVEL keyword. If *level*=1, only the NAO transformation will be carried out. For *level*=2 the NBO transformation is performed, and for 3 the NLMO (default).

Sometimes, the NBO procedure will not converge due to a bad ordering on the 2-center bond search. The first run is based on the Wiberg bond index, but the algorithm switches to the atom ordering on the subsequent runs. This can be avoided by the use of the option KEEP_WBI. If *wbi_option*=1, the Wiberg bond index is used in all iterations.

34.4.2 Saving the NLMO orbitals (SAVE)

SAVE, *record.file*;

The NLMO orbitals are saved in the specified *record*, together with the NPA charges.

34.5 Finite field calculations

Dipole moments, quadrupole moments etc. and the corresponding polarizabilities can be obtained as energy derivatives by the finite difference approximation. This is most easily done with the DIP, QUAD, or FIELD commands. An error will result if the added perturbation is not totally symmetric (symmetry 1). Note that the orbitals must be recomputed before performing a correlation calculation.

34.5.1 Dipole fields (DIP)

DIP,*xfield,yfield,zfield*;
DIP+,*xfield,yfield,zfield*;

Add a finite dipole field to the one electron Hamiltonian and the core energy. The field strength is given by *xfield,yfield,zfield*. DIP+ adds to any existing field, otherwise any previous field is removed.

34.5.2 Quadrupole fields (QUAD)

QUAD,*xxfield,yyfield,zzfield,xyfield,xzfield,yzfield*;
QUAD+,*xxfield,yyfield,zzfield,xyfield,xzfield,yzfield*;

Exactly as the DIP command, but adds a quadrupole field.

34.5.3 General fields (FIELD)

FIELD,*oper1,fac1, oper2,fac2, . . .* ;
FIELD+,*oper1,fac1, oper2,fac2, . . .* ;

Adds one-electron operators *oper1*, *oper2*, . . . with the corresponding factors *fac1*, *fac2*, . . . to the one-electron hamiltonian. The available operators are given in section 6.13. An error will result if the added perturbation is not totally symmetric (symmetry 1).

FIELD+ adds to any existing field, otherwise any previous field is removed.

Note that FIELD does currently not modify core polarization potentials (CPP). If CPPs are present, only DIP and QUAD should be used.

34.5.4 Examples

The first examples shows various possibilities to add perturbations to the one-electron hamiltonian.

```
***,H2O finite fields
memory,4,m
R     =    0.96488518 ANG
THETA =  101.90140469
geometry={H1
          O,H1,R;
          H2,O,R,H1,THETA}
{hf;wf,10,1}           !scf without field

f=0.05

dip,,,f                !add dipole (z) field to h0
hf                     !do scf with modified h0

field,dmz,f            !add dipole (z) field to H0
                       !same result as previous example
hf                     !do scf with modified h0

quad,,,f               !add quadrupole (qmzz) field to h0
hf                     !do scf with modified h0

field,qmzz,f           !add quadrupole (qmzz) field to h0;
                       !same result as previous example
hf                     !do scf with modified h0

field,zz,f,xx,-0.5*f,yy,-0.5*f
                       !add general field; same result as quad above
hf                     !do scf with modified h0

field,zz,f           !same as before with separate field commands
field+,xx,-0.5*f
field+,yy,-0.5*f
hf                     !do scf with modified h0

field                  !remove field
hf                     !scf without field
```

http://www.molpro.net/info/current/examples/field.com

The second example shows how to compute dipole moments and polarizabilities using finite fields.

```
***,H2O finite field calculations

r=1.85,theta=104                   !set geometry parameters
geometry={O;                       !z-matrix input
          H1,O,r;
          H2,O,r,H1,theta}
basis=avtz                         !define default basis
field=[0,0.005,-0.005]             !define finite field strengths
$method=[hf,mp4,ccsd(t),casscf,mrci]

k=0
do i=1,#field                      !loop over fields
  dip,,,field(i)                   !add finite field to H
  do m=1,#method                   !loop over methods
    k=k+1
    $method(m)                     !calculate energy
    e(k)=energy                    !save energy
  enddo
enddo

k=0
n=#method
do m=1,#method
  k=k+1
  energ(m)=e(k)
  dipmz(m)=(e(k+n)-e(k+2*n))/(field(2)-field(3))   !dipole moment as first energy derivative
  dpolz(m)=(e(k+n)+e(k+2*n)-2*e(k))/((field(2)-field(1))*(field(3)-field(1)))  !polarizability
enddo

table,method,energ,dipmz,dpolz
title,results for H2O, r=$R, theta=$theta, basis=$basis
---
```

http://www.molpro.net/info/current/examples/h2o_field.com

34.6 Relativistic corrections

Relativistic corrections may be calculated within the Cowan-Griffin approach by computing expectation values of the mass-velocity and 1-electron Darwin integrals; these should be generated using the property integral program with keyword REL The expectation values can be computed within the SCF, MCSCF and CI programs in the usual way using the EXPECT command, again with the keyword REL. The mass-velocity and Darwin terms, and their sum are subsequently available through the MOLPRO variables MASSV, DARW and EREL respectively.

34.6.1 Example

```
***,ar2
geometry={ar1;ar2,ar1,r}   !geometry definition
r=2.5 ang                  !bond distance
{hf;                       !non-relativisitic scf calculation
expec,rel,darwin,massv}    !compute relativistic correction using Cowan-Griffin operator
e_nrel=energy              !save non-relativistic energy in variable enrel
show,massv,darwin,erel     !show individual contribution and their sum

dkroll=1                   !use douglas-kroll one-electron integrals
hf;                        !relativistic scf calculation
e_dk=energy                !save relativistic scf energy in variable e_dk.
show,massv,darwin,erel     !show mass-velocity and darwin contributions and their sum
show,e_dk-e_nrel           !show relativistic correction using Douglas-Kroll
```

http://www.molpro.net/info/current/examples/ar2_rel.com

34.7 CUBE — dump density or orbital values

CUBE,*filename,iflag*,n_1,n_2,n_3

calls a module which dumps the values of various properties on a spatial parallelopipedal grid to an external file. The purpose is to allow plotting of orbitals, densities and other quantities by external programs. The format of the file is intended to be the same as that produced by other programs.

filename is the unix path name of the file to be written, and its specification is mandatory.

iflag If *iflag* is negative (default), a formatted file will be written, otherwise unformatted fortran i/o will be used.

n_1,n_2,n_3 specify the number of grid points in each of three dimensions. If not specified, sensible defaults are chosen.

By default, the last density computed is evaluated on the grid, and written to *filename*. This behaviour can be modified by one or more of the following subcommands.

34.7.1 STEP — setting the point spacing

STEP,[*stepx*],[*stepy*], [*stepz*]

stepx,stepy, stepz specify the point spacing in each of three axis directions. By default, the value of *stepx,stepy, stepz* is determinated by the number of grid points, the bragg radii of the atoms, and some related parameters.

34.7.2 DENSITY — source of density

DENSITY,[*density-source*]
GRADIENT,[*density-source*]
LAPLACIAN,[*density-source*]

Compute the density and, optionally, its gradient and laplacian. *¡density-source¿* may be a record number containing the required density, and may contain further qualification, such as set number, in the usual way. By default, the last computed density is taken.

34.7.3 ORBITAL — source of orbitals

`ORBITAL`,[*orbital-list*],[`RECORD=`*orbital-source*]

¡orbital-list¿ is a list of one or more orbital numbers of the form *number.symmetry* or keywords chosen from `HOMO`, `LUMO`, `OCC` (all occupied orbitals), `ALL`. If nothing is specified, the default is `HOMO`. *¡orbital-source¿* may be a record number containing the required density, and may contain further qualification, such as set number, in the usual way. By default, the last computed orbitals are taken.

Note that the CUBE file format precludes simultaneous orbital and density dumps, but that this may be achieved in the GOPENMOL format (see 34.8).

34.7.4 AXIS — direction of grid axes

`AXIS`,*x,y,z*

x,y,z specify the unnormalised direction cosines of one of the three axes defining the grid. Up to three `AXIS` commands can be given, but none is required. Axes need not be orthogonal. By default, the first axis is the cartesian *x*, the second is orthogonal to the first and to the cartesian *z*, and the third is orthogonal to the first two.

34.7.5 BRAGG — spatial extent of grid

Based on the direction of the coordinate axes, a parallelopiped (in the usual case of orthogonal axes, a cuboid) is constructed to contain the molecule completely. The atoms are assumed to be spherical, with an extent proportional to their Bragg radii, and the constant of proportionality can be changed from the default value using

`BRAGG`,*scale*

After the parallelopiped has been constructed, the grid is laid out with equal spacing to cover it using the number of points specified on the `CUBE` command.

34.7.6 ORIGIN — centroid of grid

`ORIGIN`,*x,y,z*

x,y,z specify the centroid of the grid. It is usually not necessary to use this option, since the default should suffice for most purposes.

34.7.7 TITLE — user defined title

`TITLE`,*title*

Set a user defined title in the cube file.

34.7.8 DESCRIPTION — user defined description

`DESCRIPTION`,*description*

Set a user defined description in the cube file.

34.7.9 Format of cube file

The formatted cube file contains the following records

`(A)`	job title.
`(A)`	brief description of the file contents.
`(I5,3F12.6)`	number of atoms, coordinates of grid origin (bohr).
`(I5,3F12.6)`	number of grid points n_1, step vector for first grid dimension.
`(I5,3F12.6)`	number of grid points n_2, step vector for second grid dimension.
`(I5,3F12.6)`	number of grid points n_3, step vector for third grid dimension.
`(I5,4F12.6)`	atomic number, charge and coordinates; one such record for each atom.
`(6E13.5)`	$n_1 \times n_2$ records of length n_3 containing the values of the density or orbital at each grid point. In the case of a number of orbitals m, the record length is $m \times n_3$, with the data for a single grid point grouped together. In the case of the density gradient, there is first a record of length n_3 containing the density, then one of length $3n_3$ containing the gradient, with the three cartesian components contiguous. For the laplacian, there is a further record of length n_3.

34.8 GOPENMOL — calculate grids for visualization in gOpenMol

`GOPENMOL`,*filename,iflag*,n_1,n_2,n_3

The syntax and sub-options are exactly the same as for CUBE, except that the files produced are in a format that can be used directly in the gOpenMol visualization program. The following should be noted.

- Only the base name (up to the last '.') in *filename* is used, and is appended by different suffices to create several different files:

.crd	A CHARMm CRD-format file containing the coordinates is always produced, and may be used in the invocation of gOpenMol: `rungOpenMol -i`*filename*`.crd`
`_density.plt`	If `DENSITY` is given, then the file *filename*`_density.plt` is produced and contains the density grid in gOpenMol internal format.
`_orbital_`*number*`.`*symmetry*`.plt`	If `ORBITAL` is given, then for each orbital *number*`.`*symmetry* specified, the file *filename*`_orbital_`*number*`.`*symmetry*`.plt` is produced and contains the orbital grid in gOpenMol internal format.

- The default is not to produce any orbitals or densities, and so only the atomic coordinates are dumped.
- The default is to use unformatted binary files, and this should not normally be changed.
- The `ORIGIN` and `AXIS` commands should not be used.
- If

  `INTERACT`

  is given in the input, when all the grids have been calculated, an attempt is made to start gOpenMol by executing the Unix command `rungOpenMol`. If `rungOpenMol` is not in `$PATH`, then nothing happens. Otherwise, gOpenMol should start and display the molecule. Any `.plt` files produced can be added to the display by following the `Plot`;`Contour` menu item. The name of the Unix command may be changed from the default `rungOpenMol` by specifying it as the first argument to the `INTERACT` directive. By default, gOpenMol is not started, and this is equivalent to giving the command `BATCH`.

35 RELATIVISTIC CORRECTIONS

There are three ways in MOLPROto take into account scalar relativistic effects:

1. Use the Douglas-Kroll relativistic one-electron integrals.
2. Compute a perturbational correction using the Cowan-Griffin operator (see section 6.13).
3. Use relativistic effective core potentials (see section 12).

35.1 Using the Douglas–Kroll–Hess Hamiltonian

For all-electron calculations, the prefered way is to use the Douglas–Kroll–Hess (DKH) Hamiltonian, which is available up to arbitrary order in MOLPRO. It is activated by setting

```
SET,DKROLL=1
```

somewhere in the input before the first energy calculation. If no further input is specified, the standard second-order Douglas–Kroll–Hess Hamiltonian (DKH2) is used.

Starting with version 2006.1, MOLPRO does, however, also provide the DKH Hamiltonian up to (in principle) any arbitrary order of decoupling (DKH*n*). The desired DKH order (DKHO) and the chosen parametrization for the unitary transformations have to be specified by

`SET,DKHO=`n, $(n = 2,\dots,12)$,
`SET,DKHP=`m, $(m = 1,\dots,5)$

below the `DKROLL=1` statement in the input file. Alternatively, these values can be given as options on the `INT` command:

`INT,DKROLL=1,DKHO=`n`,DKHP=`m.

The possible parametrizations supported by MOLPRO are:

`DKHP=1`:	Optimum parametrization (OPT, default)
`DKHP=2`:	Exponential parametrization (EXP)

DKHP=3:	Square-root parametrization (SQR)
DKHP=4:	McWeeny parametrization (MCW)
DKHP=5:	Cayley parametrization (CAY)

Example:

```
SET,DKROLL=1   !   activate Douglas–Kroll–Hess one-electron integrals
SET,DKHO=8     !   DKH order = 8
SET,DKHP=4     !   choose McWeeny parametrization for unitary transformations
```

(Note: For DKHO $\geq$ 13 the values of some parameters in the file src/common/dkhparameters.inc have to be suitably increased. Only recommended for experts who do exactly know what they are doing!! For most cases DKHO=10 is sufficient.)

Up to fourth order (DKHO=4) the DKH Hamiltonian is independent of the chosen paramterization. Higher-order DKH Hamiltonians depend slightly on the chosen paramterization of the unitary transformations applied in order to decouple the Dirac Hamiltonian.

For details on the infinite-order DKH Hamiltonians see

M. Reiher, A. Wolf, JCP **121**, 2037–2047 (2004),
M. Reiher, A. Wolf, JCP **121**, 10945–10956 (2004).

For details on the different parametrizations of the unitary transformations see
A. Wolf, M. Reiher, B. A. Hess, JCP **117**, 9215–9226 (2002).

35.2 Example for computing relativistic corrections

```
***,ar2
geometry={ar1;ar2,ar1,r}    !geometry definition
r=2.5 ang                   !bond distance
{hf;                        !non-relativisitic scf calculation
expec,rel,darwin,massv}     !compute relativistic correction using Cowan-Griffin operator
e_nrel=energy               !save non-relativistic energy in variable enrel
show,massv,darwin,erel      !show individual contribution and their sum

dkroll=1                    !use douglas-kroll one-electron integrals
hf;                         !relativistic scf calculation
e_dk=energy                 !save relativistic scf energy in variable e_dk.
show,massv,darwin,erel      !show mass-velocity and darwin contributions and their sum
show,e_dk-e_nrel            !show relativistic correction using Douglas-Kroll
```

http://www.molpro.net/info/current/examples/ar2_rel.com

36 DIABATIC ORBITALS

In order to construct diabatic states, it is necessary to determine the mixing of the diabatic states in the adiabatic wavefunctions. In principle, this mixing can be obtained by integration of the non-adiabatic coupling matrix elements. Often, it is much easier to use an approximate method, in which the mixing is determined by inspection of the CI coefficients of the MCSCF or CI wavefunctions. This method is applicable only if the orbital mixing is negligible. For CASSCF wavefunctions this can be achieved by maximizing the overlap of the active orbitals with those of a reference geometry, at which the wavefunctions are assumed to be diabatic (e.g. for symmetry reasons). The orbital overlap is maximized using using the new DIAB command in the MCSCF program.

This procedure works as follows: first, the orbitals are determined at the reference geometry. Then, the calculations are performed at displaced geometries, and the "diabatic" active orbitals, which have maximum overlap with the active orbitals at the reference geometry, are obtained by adding a `DIAB` directive to the input:

Old form (`Molpro96`, obsolete):

`DIAB`,*orbref, orbsav, orb1,orb2,pri*

New form:

`DIAB`,*orbref* [,`TYPE`=*orbtype*] [,`STATE`=*state*] [,`SPIN`=*spin*] [,*MS2*=*ms2*] [,`SAVE`=*orbsav*] [,`ORB1`=*orb1*, `ORB2`=*orb2*] [,*PRINT*=*pri*]

Here *orbref* is the record holding the orbitals of the reference geometry, and *orbsav* is the record on which the new orbitals are stored. If *orbsav* is not given (recommended!) the new orbitals are stored in the default dump record (2140.2) or the one given on the `ORBITAL` directive (see section 19.5.3). In contrast to earlier versions of MOLPRO it is possible that *orbref* and *orbsav* are the same. The specifications `TYPE`, `STATE`, `SPIN` can be used to select specific sets of reference orbitals, as described in section 4.11. *orb1, orb2* is a pair of orbitals for which the overlap is to be maximized. These orbitals are specified in the form *number.sym*, e.g. 3.1 means the third orbital in symmetry 1. If *orb1, orb2* are not given, the overlap of all active orbitals is maximized. *pri* is a print parameter. If this is set to 1, the transformation angles for each orbital are printed for each jacobi iteration.

Using the defaults described above, the following input is sufficient in most cases:

`DIAB`,*orbref*

Using `Molpro98` is is not necessary any more to give any `GEOM` and `DISPL` cards. The displacements and overlap matrices are computed automatically (the geometries are stored in the dump records, along with the orbitals).

The diabatic orbitals have the property that the sum of orbital and overlap contributions in the non-adiabatic coupling matrix elements become approximately zero, such that the adiabatic mixing occurs only through changes of the CI coefficients. This allows to determine the mixing angle directly from the CI coefficients, either in a simple way as described for instance in J. Chem. Phys. **89**, 3139 (1988), or in a more advanced manner as described by Pacher, Cederbaum, and Köppel in J. Chem. Phys. **89**, 7367 (1988).

Below we present an example for the first two excited states of H_2S, which have B_1 and A_2 symmetry in C_{2v}, and A'' symmetry in C_S. We first perform a reference calculation in C_{2v} symmetry, and then determine the diabatic orbitals for displaced geometries in C_S symmetry. Each subsequent calculation uses the previous orbitals as reference. One could also use the orbitals of the C_{2v} calculation as reference for all other calculations. In this case one would have to take out the second-last input card, which sets `reforb=2141.2`.

```
***,H2S diabatic A" states

basis=VDZ                              !use cc-pVDZ basis set
symmetry,x,planeyz                     !use Cs symmetry & fix orientation of the molecule
orient,noorient                        !dont allow automatic reorientation
geometry={s;h1,s,r1;h2,s,r2,h1,theta}  !Z-matrix geometry input

gprint,orbitals,civector               !global print options

text,reference calculation for C2V
theta=92.12,r1=2.3,r2=2.3              !reference geometry

{hf;occ,7,2;wf,18,1}                   !scf calculation for ground state

{multi;occ,9,2;closed,4,1;             !define active and inactive spaces
wf,18,2;state,2;                       !two A" states (1B1 and 1A2 in C2v)
orbital,2140.2}                        !save orbitals to 2140.2
reforb=2140.2

text,calculations at displaced geometries

rd=[2.4,2.5,2.6]                       !define a range of bond distances

do i=1,#rd                             !loop over displaced geometries

r2=rd(i)                               !set r2 to current distance

{multi;occ,9,2;closed,4,1;             !same wavefunction definition as at reference geom.
wf,18,2;state,2;
orbital,2141.2                         !save new orbitals to record
diab,reforb}                           !compute diabatic orbitals using reference orbitals
                                       !stored on record reforb
reforb=2141.2                          !set variable reforb to the new orbitals.
enddo
```

http://www.molpro.net/info/current/examples/h2s_diab.com

37 NON ADIABATIC COUPLING MATRIX ELEMENTS

Non-adiabatic coupling matrix elements can be computed by finite differences for MCSCF or CI wavefunctions using the DDR program. For state-averaged MCSCF wavefunctions, they can also computed analytically (cf. section 19.9.2).

Note that present numerical procedure has been much simplified relative to Molpro96. No GEOM and DISPL input cards are needed any more, and the three necessary calculations can be done in any order.

37.1 The DDR procedure

In order to compute the coupling matrix elements by finite differences, one has to compute and store the wavefunctions at two (first-order algorithm) or three (second-order algorithm) slightly displaced geometries. The order of these calculations is arbitrary.

The typical strategy is as follows:

1.) Compute the wavefunction at the reference geometry. The wavefunctions for both states have to be stored using the SAVE command of the CI program. If the matrix elements are computed for MCSCF wavefunctions, it is necessary to recompute the wavefunction with the CI program, using the NOEXC option. The transition density matrix is stored using the DM directive of the CI program.

2.) Compute the wavefunctions at the (positively) displaced geometry and store the CI wavefunction in a second record.

3.) If the second-order (three-point) method is used, step (2) is repeated at a (negatively) displaced geometry.

4.) Compute the transition density matrices between the states at the reference geometry and the displaced geometr(ies). This is done with the TRANS directive of the CI program.

5.) Finally, the DDR program is used to assemble the matrix element. Using the first-order two-point method, only a single input line is needed:

DDR, *dr, orb1, orb2, trdm2*

where *dr* is the geometry increment used as denominator in the finite difference method, *orb1* is the record holding the orbitals of the reference geometry, *orb2* is the record holding the orbitals of the displaced geometry, and *trdm2* is the record holding the transition density matrix computed from the CI-vectors at *R* and *R*+*DR*.

If central differences (three points) are used, the input is as follows:

DDR,*2*dr*
ORBITAL,*orb1,orb2,orb3*
DENSITY,*trdm1,trdm2,trdm3*

where *dr, orb1, orb2* are as above, and *orb3* is the record holding the orbitals at the negatively displaced geometry.

trdm1, trdm2, trdm3 are the records holding the transition densities $\gamma(R|R)$, $\gamma(R|R+DR)$, and $\gamma(R|R-DR)$, respectively.

If more than two states are computed simultaneously, the transition density matrices for all pairs of states will be stored in the same record. In that case, and also when there are just two states whose spatial symmetry is not 1, it is necessary to specify for which states the coupling is to be computed using the STATE directive:

STATE,$state_1$, $state_2$

where $state_i$ is of the form *istate.isym* (the symmetries of both states must be the same, and it is therefore sufficient to specify the symmetry of the first state).

As an example the input for first-order and second-order calculations is given below. The calculation is repeated for a range of geometries, and at the end of the calculation the results are printed using the TABLE command.

In the calculation shown, the "diabatic" CASSCF orbitals are generated in the two CASSCF calculations at the displaced geometries by maximizing the overlap with the orbitals at the reference geometry. This is optional, and (within the numerical accuacy) does not influence the final results. However, the relative contributions of the orbital, overlap and CI contributions to the NACME are modified. If diabatic orbitals are used, which change as little as possible as function of geometry, the sum of overlap and orbital contribution is minimized, and to a very good approximation the NACME could be obtained from the CI-vectors alone.

```
***,lif non-adiabatic coupling
memory,1,m

basis,f=avdz,li=vdz              !define basis
r=[10.0,10.5,11.0,11.5,12.0]     !define bond distances
dr=0.01                          !define increment
geometry={li;f,li,rlif}          !define geometry

rlif=3                           !first calculation at R=3
{hf;occ,4,1,1}                   !SCF
{multi;closed,3;                 !CASSCF, 3 inactive orbitals
wf,12,1;state,2;                 !Two 1A1 states
orbital,2140.2}                  !dump orbitals to record 2140.2

do i=1,#r                        !loop over geometries
rlif=r(i)                        !set bond distance
{multi;closed,3;                 !CASSCF, 3 inactive orbitals
wf,12,1;state,2;                 !Two 1A1 states
orbital,2140.2}                  !Overwrite previous orbitals by present ones

{ci;state,2;noexc;               !CI for 2 states, no excitations
save,6000.2;                     !save wavefunction to record 6000.2
dm,8000.2}                       !save (transition) densities to record 8000.2

rlif=r(i)+dr                     !increment bond distance by dr

{multi;closed,3;                 !same CASSCF as above
wf,12,1;state,2;                 !Two 1A1 states
start,2140.2;                    !start with orbitals from reference geometry
orbital,2141.2;                  !save orbitals to record 2141.2
diab,2140.2}                     !generate diabatic orbitals by maximizing the
                                 !overlap with the orbitals at the reference geometry

{ci;state,2;noexc;save,6001.2}    !CI for 2 states, wavefunction saved to record 6001.2

{ci;trans,6000.2,6001.2;         !Compute overlap and transition density <R|R+DR>
dm,8100.2}                       !Save transition density to record 8100.2

rlif=r(i)-dr                     !repeat at r-dr

{multi;closed,3;                 !same CASSCF as above
wf,12,1;state,2;                 !Two 1A1 states
start,2140.2;                    !start with orbitals from reference geometry
orbital,2142.2;                  !save orbitals to record 2142.2
diab,2140.2}                     !generate diabatic orbitals by maximizing the
                                 !overlap with the orbitals at the reference geometry

{ci;state,2;noexc;save,6002.2}   !CI for 2 states, wavefunction saved to record 6002.2

{ci;trans,6000.2,6002.2;         !Compute overlap and transition density <R|R-DR>
dm,8200.2}                       !Save transition density to record 8200.2

{ddr,dr,2140.2,2141.2,8100.2}    !compute NACME using 2-point formula (forward difference)
nacme1p(i)=nacme                 !store result in variable nacme1p
{ddr,-dr,2140.2,2142.2,8200.2}   !compute NACME using 2-point formula (backward difference)
nacme1m(i)=nacme                 !store result in variable nacme1m

{ddr,2*dr                        !compute NACME using 3-point formula
orbital,2140.2,2141.2,2142.2;    !orbital records for R, R+DR, R-DR
density,8000.2,8100.2,8200.2}    !transition density records for R, R+DR, R-DR
nacme2(i)=nacme                  !store result in variable nacme2

end do                           !end of loop over differend bond distances

nacmeav=(nacme1p+nacme1m)*0.5    !average the two results forward and backward differences
table,r,nacme1p,nacme1m,nacmeav,nacme2   !print a table with results
title,Non-adiabatic couplings for LiF    !title for table
```

http://www.molpro.net/info/current/examples/lif_nacme.com

This calculation produces the following table:

```
 Non-adiabatic couplings for LiF

      R       NACME1P       NACME1M       NACMEAV        NACME2
   10.0   -0.22828936   -0.22328949   -0.22578942   -0.22578942
   10.5   -0.51777034   -0.50728914   -0.51252974   -0.51252974
   11.0    0.76672943    0.76125391    0.76399167    0.76399167
   11.5    0.42565202    0.42750263    0.42657733    0.42657733
   12.0    0.19199878    0.19246799    0.19223338    0.19223338
```

Note that the sign changes because of a phase change of one of the wavefunctions. In order to keep track of the sign, one has to inspect both the orbitals and the ci-vectors.

38 QUASI-DIABATIZATION

The DDR procedure can also be used to generate quasi-diabatic states and energies for MRCI wavefucntions (CASSCF case can be treated as special case using the `NOEXC` directive in the MRCI). The quasi-diabatic states have the propery that they change as little as possible relative to a reference geometry; with other words, the overlap between the states at the current geometry with those at a reference geometry is maximized by performing a unitary transformation among the given states. Preferably, the adiabatic and diabatic states should be identical at the reference geometry, e.g., due to symmetry. For instance, in the examples given below for the $^1B_1$ and $^1A_2$ states of H_2S, C_{2v} geomtries are used as reference, and at these geometries the states are unmixed due to their different symmetry. At the displaced geometries the molecular symmetry is reduced to C_S. Both states now belong to the $^1A''$ irreducible representation and are strongly mixed. For a description and application of the procedure described below, see D. Simah, B. Hartke, and H.-J. Werner, J. Chem. Phys. **111**, 4523 (1999).

This diabatization can be done automatically and requires two steps: first, the active orbitals of a CASSCF calculation are rotated to maximize the overlap with the orbitals at the reference geometry. This is achieved using the `DIAB` procedure described in section 19.5.8. Secondly, the `DDR` procedure can be used to find the transformation among the CI vectors.

The following input is required:

`DDR` calls the DDR procedure.

`ORBITAL`,*orb1, orb2* — *orb1* and *orb2* are the (diabatic) orbitals at the current and reference geometry, respectively.

`DENSITY`,*trdm1,trdm2* — *trdm1* are the transition densities computed at the current geometry, *trdm2* are transition densities computed using the wavefunctions of the current (bra) and reference (ket) geometries.

`MIXING`,*state1, state2, ...* — The given states are included in the diabatization.

`ENERGY`,*e1, e2, ...* — Adiabatic energies of the states. If this input card is present, the Hamiltonian in the basis of the diabatic states is computed and printed. Alternatively, the energies can be passed to `DDR` using the Molpro variable `EADIA`.

The results are printed and stored in the following Molpro variables, provided the ENERGY directive or the EADIA variable is found:

Results including the first-order orbital correction:

SMAT	The first $nstate \times nstate$ elements contain the state overlap matrix (bra index rans fastest).
UMAT	The first $nstate \times nstate$ elements contain the transformation matrix.
HDIA	The first $nstate \cdot (nstate+1)/2$ elements contain the lower triangle of the diabatic hamiltonian.
MIXANG	Non-adiabatic mixing angle in degree. This is available only in the two-state case.

The corresponding results obtained from the CI-vectors only (without orbital correction) are stored in the variables [SMATCI], UMATCI, HDIACI, and MIXANGCI.

The way it works is most easily demonstrated for some examples. In the following input, the wavefunction is first computed at the C_{2v} reference geometry, and then at displaced geometries.

```
***,h2s Diabatization
memory,3,m

gprint,orbitals,civector

symmetry,x
orient,noorient             !noorient should always be used for diabatization
geometry={
         s;
         h1,s,r1;
         h2,s,r2,h1,theta}

basis=avdz                  !This basis is too small for real application

r1=2.5                      !Reference geometry
theta=[92]

r=[2.50,2.55,2.60]          !Displaced geometries

reforb=2140.2               !Orbital dumprecord at reference geometry
refci=6000.2                !MRCI record at reference geometry
savci=6100.2                !MRCI record at displaced geometries

text,compute wavefunction at reference geometry (C2v)
r2=r1

{hf;occ,9,2;wf,18,2,4;
orbital,2100.2}

{multi;occ,9,2;closed,4,1;
wf,18,2;state,2;            !1B1 and 1A2 states
natorb,reforb               !Save reference orbitals on reforb
noextra}                    !Dont use extra symmetries

{ci;occ,9,2;closed,4,1;      !MRCI at reference geometry
wf,18,2,0;state,2;          !1B1 and 1A2 states
orbital,reforb              !Use orbitals from previous CASSCF
save,refci}                 !Save MRCI wavefunction

Text,Displaced geometries

do i=1,#r                   !Loop over different r values
data,truncate,savci+1       !truncate dumpfile after reference
r2=r(i)                     !Set current r2

{multi;occ,9,2;closed,4,1;
wf,18,2,0;state,2;          !Wavefunction definition
start,reforb                !Starting orbitals
orbital,3140.2;             !Dump record for orbitals
diab,reforb                 !Generate diabatic orbitals relative to reference geometry
noextra}                    !Dont use extra symmetries

{ci;occ,9,2;closed,4,1;
wf,18,2,0;state,2;          !1B1 and 1A2 states
orbital,diabatic            !Use diabatic orbitals
save,savci}                 !Save MRCI for displaced geometries

e1(i)=energy(1)             !Save adiabatic energies
e2(i)=energy(2)

{ci;trans,savci,savci        !Compute transition densities at R2
dm,7000.2}                  !Save transition densities on this record
{ci;trans,savci,refci;       !Compute transition densities between R2 and R1
dm,7100.2}                  !Save transition densities on this record

{ddr
density,7000.2,7100.2       !Densities for <R2||R2> and <R2||R1>
orbital,3140.2,2140.2       !Orbitals for <R2||R2> and <R2||R1>
energy,e1(i),e2(i)          !Adiabatic energies
mixing,1.2,2.2}             !Compute mixing angle and diabatic energies
```

This calculation produces the following results:

```
 Diabatic energies for H2S, obtained from CI-vectors

       R       E1             E2             H11CI          H22CI          H21CI
     2.50 -398.64296319 -398.63384782 -398.64296319 -398.63384782    0.00000
     2.55 -398.64572746 -398.63666636 -398.64509901 -398.63729481   -0.00230
     2.60 -398.64911752 -398.63771802 -398.64662578 -398.64020976   -0.00471

 Diabatic energies for H2S, obtained from CI-vectors and orbital correction

       R       E1             E2             H11            H22            H21
     2.50 -398.64296319 -398.63384782 -398.64296319 -398.63384782    0.00000
     2.55 -398.64572746 -398.63666636 -398.64509941 -398.63729441   -0.00230
     2.60 -398.64911752 -398.63771802 -398.64662526 -398.64021027   -0.00471
```

The results in the first table are obtained from the CI-contribution to the state-overlap matrix only, while the ones in the second table include a first-order correction for the orbitals. In this case, both results are almost identical, since the `DIAB` procedure has been used to minimize the change of the active orbitals. This is the recommended procedure. If simply natural orbitals are used without orbital diabatization, the following results are obtained from the otherwise unchanged calculation:

```
 Diabatic energies for H2S, obtained from CI-vectors

       R       E1             E2             H11CI          H22CI          H21CI
     2.50 -398.64296319 -398.63384782 -398.64296319 -398.63384782    0.00000
     2.55 -398.64572742 -398.63666630 -398.64475612 -398.63763760   -0.00280
     2.60 -398.64911746 -398.63771803 -398.64521031 -398.64162518   -0.00541

 Diabatic energies for H2S, obtained from CI-vectors and orbital correction

       R       E1             E2             H11            H22            H21
     2.50 -398.64296319 -398.63384782 -398.64296319 -398.63384782    0.00000
     2.55 -398.64572742 -398.63666630 -398.64509146 -398.63730226   -0.00231
     2.60 -398.64911746 -398.63771803 -398.64648358 -398.64035190   -0.00480
```

It is seen that the mixing obtained from the CI vectors only is now very different and meaningless, since the orbitals change significantly as function of geometry. However, the second calculations, which accounts for this change approximately, still gives results in quite good agreement with the calculation involving diabatic orbitals.

The final examples shows a more complicated input, which also computes the non-adiabatic coupling matrix elements. In a two-state model, the NACME should equal the first derivative of the mixing angle. In the example, the NACME is computed using the 3-point `DDR` method (NACMECI), and also by finite difference of the mixing angle (DCHI).

```
***,h2s Diabatization and NACME calculation
memory,3,m

gprint,orbitals,civector

symmetry,x
orient,noorient              !noorient should always be used for diabatization
geometry={
         s;
         h1,s,r1;
         h2,s,r2,h1,theta}

basis=avdz                   !This basis is too small for real application

r1=2.5                       !Reference geometry
theta=[92]

r=[2.55,2.60]                !Displaced geometries
dr=[0,0.01,-0.01]            !Samll displacements for finite difference NACME calculation

reforb1=2140.2               !Orbital dumprecord at reference geometry
refci=6000.2                 !MRCI record at reference geometry
savci=6100.2                 !MRCI record at displaced geometries

text,compute wavefunction at reference geometry (C2v)
r2=r1

{hf;occ,9,2;wf,18,2,4;orbital,2100.2}

{multi;occ,9,2;closed,4,1;
wf,18,2;state,2;             !1B1 and 1A2 states
natorb,reforb1               !Save reference orbitals on reforb1
noextra}                     !Dont use extra symmetries

{ci;occ,9,2;closed,4,1;       !MRCI at reference geometry
wf,18,2,0;state,2;           !1B1 and 1A2 states
orbital,reforb1              !Use orbitals from previous CASSCF
save,refci}                  !Save MRCI wavefunction

Text,Displaced geometries

do i=1,#r                    !Loop over different r values
data,truncate,savci+1        !truncate dumpfile after reference
reforb=reforb1

do j=1,3                     !Loop over small displacements for NACME
r2=r(i)+dr(j)                !Set current r2

{multi;occ,9,2;closed,4,1;
wf,18,2,0;state,2;           !Wavefunction definition
start,reforb                 !Starting orbitals
orbital,3140.2+j;            !Dumprecord for orbitals
diab,reforb                  !Generate diabatic orbitals relative to reference geometry
noextra}                     !Dont use extra symmetries

reforb=3141.2                !Use orbitals for j=1 as reference for j=2,3

{ci;occ,9,2;closed,4,1;
wf,18,2,0;state,2;
orbital,diabatic             !Use diabatic orbitals
save,savci+j}                !Save MRCI for displaced geometries

eadia=energy                 !Save adiabatic energies for use in ddr
if(j.eq.1) then
e1(i)=energy(1)              !Save adiabatic energies for table printing
e2(i)=energy(2)
end if

{ci;trans,savci+j,savci+j;   !Compute transition densities at R2+DR(j)
dm,7000.2+j}                 !Save transition densities on this record
{ci;trans,savci+j,refci;     !Compute transition densities between R2+DR(j) and R1
```

The calculation produces the following table

```
 Mixing angles and non-adiabatic coupling matrix elements for H2S

       R           MIXCI          MIXTOT            DCHI         NACMECI
     2.55        15.2694        15.2644         -5.2226         -5.2365
     2.60        27.8740        27.8772         -3.4702         -3.4794

 Diabatic energies for H2S, obtained from CI-vectors

       R        E1             E2             H11CI          H22CI          H21CI
     2.55 -398.64572746 -398.63666636 -398.64509901 -398.63729481   -0.002302
     2.60 -398.64911752 -398.63771802 -398.64662578 -398.64020976   -0.004711

 Diabatic energies for H2S, obtained from CI-vectors and orbital correction

       R        E1             E2             H11            H22            H21
     2.55 -398.64572746 -398.63666636 -398.64509941 -398.63729441   -0.002301
     2.60 -398.64911752 -398.63771802 -398.64662526 -398.64021027   -0.004711
```

As expected the coupling matrix elements obtained from the 3-point `DDR` calculation (NACMECI) and by differentiating the mixing angle (DCHI) are in close agreement.

39 THE VB PROGRAM CASVB

CASVB is a general program for valence bond calculations
written by T. Thorsteinsson and D. L. Cooper (1996–2005).

This program can be used in two basic modes:

a) variational optimization of quite general types of nonorthogonal MCSCF or modern valence bond wavefunctions

b) representation of CASSCF wavefunctions in modern valence form, using overlap- (*relatively inexpensive*) or energy-based criteria.

Bibliography:

T. Thorsteinsson, D. L. Cooper, J. Gerratt, P. B. Karadakov and M. Raimondi, Theor. Chim. Acta **93**, 343–66 (1996).
D. L. Cooper, T. Thorsteinsson and J. Gerratt, Int. J. Quant. Chem. **65**, 439–51 (1997).
D. L. Cooper, T. Thorsteinsson and J. Gerratt, Adv. Quant. Chem. **32**, 51–67 (1998).
T. Thorsteinsson and D. L. Cooper, in *Quantum Systems in Chemistry and Physics. Volume 1: Basic problems and models systems*, eds. A. Hernández-Laguna, J. Maruani, R. McWeeny, and S. Wilson (Kluwer, Dordrecht, 2000); pp 303–26.

All publications resulting from use of this program should acknowledge relevant publications. There is a more complete bibliography at http://www.liv.ac.uk/ dlc/CASVB.html

39.1 Structure of the input

All *CASVB* sub-commands may be abbreviated by four letters. The general input structure can be summarized as follows:

a) For generating representations of CASSCF wavefunctions, the program is invoked by the command `CASVB`. For variational optimization of wavefunctions it is normally invoked inside *MULTI* by the sub-command `VB` (see 19.10).

b) Definition of the CASSCF wavefunction (not generally required).

c) Definition of the valence bond wavefunction.

d) Recovery and/or storage of orbitals and vectors.

e) Manual input of starting guess (optional).

g) Optimization control.

f) Definition of molecular symmetry and possible constraints on the VB wavefunction.

h) Wavefunction analysis.

i) Further general options.

Items a) and b) should precede everything else in the input; apart from this, commands may come in any order.

39.2 Defining the CASSCF wavefunction

CASVB is interfaced with the determinant part of *MULTI* (i.e., `CONFIG,CSF;` must *not* be specified). When this program is run prior to *CASVB*, the CI vector must dumped using one of the directives `SAVE, NATORB, CANONICAL,` or `LOCALI` (see section 19.5.4). The three latter are recommended.

39.2.1 The VBDUMP directive

`VBDUMP`[,*vbdump*];

If present, the `VBDUMP` card must occur first in the *CASVB* input. It is *not* required for variational calculations.

Note that in the majority of cases (*e.g.*, if a *CASVB* run occurs immediately after *MULTI*, or for variational calculations), explicit specification of dump records with *vbdump* is not required.

Wavefunction definitions may be restored here using `VBDUMP` cards (see also Section 19.8.6). The default record name (*vbdump*) is 4299.2. If a `VBDUMP` card is not present and record 4299.2 does not exist, then *CASVB* will attempt to generate the wavefunction information automatically based on the latest MCSCF calculation (however, `STATE` and `WEIGHT` information will not be restored in such a case).

39.3 Other wavefunction directives

The definitions of the CASSCF wavefunction may also be specified manually using some or all of the directives:

`OCC`	Occupied orbitals.
`CLOSED`	Closed-shell orbitals.
`FROZEN`	Frozen-core orbitals.
`WF`	Wavefunction card.
`STATE`	Number of states for this wavefunction symmetry.
`WEIGHT`	Weights of states.

For the exact definition of these cards see sections 19.2 and 19.3. These commands may also be used to modify the values defined in `VBDUMP`. The information given on these cards should correspond to the CI vector saved in the CASSCF calculation. The cards, and their ordering, should therefore coincide with those used in *MULTI*, except for the `WEIGHT` cards which may differ. At present, the VB wavefunction must correspond to a well-defined number of electrons and total spin. Other states may be present, but an error condition will occur if non-zero weights are specified for wavefunction symmetries with varying values of *elec* or *spin*.

39.4 Defining the valence bond wavefunction

39.4.1 Specifying orbital configurations

The number of core and active orbitals (*mcore*, *mact*), active electrons (*Nact*), and the value of the total spin will be identical to that defined for the CASSCF wavefunction. The spatial VB configurations are defined in terms of the active orbitals only, and may be specified using one or more `CON` cards (note that the `RESTRICT` and `SELECT` keywords are not used in *CASVB*):

`CON`,$n_1,n_2,n_3,n_4,\ldots$;

The configurations can be specified by occupation numbers, exactly as in *MULTI* (see section 19.4.3), so that n_i is the occupation of the ith valence bond orbital. Alternatively a list of *Nact* orbital numbers (in any order) may be provided – the program determines which definition applies. The two cards `CON,1,0,1,2;` and `CON,1,3,4,4;` are thus equivalent.

If no configurations are specified the single covalent configuration $\phi_1\phi_2\cdots\phi_{Nact}$ is assumed.

39.4.2 Selecting the spin basis

`SPINBASIS`,*key*;

key may be chosen from `KOTANI` (default), `RUMER`, `PROJECT` or `LTRUMER`, specifying the basis of spin eigenfunctions used in the definition of valence bond structures. `PROJECT` refers to spin functions generated using a spin projection operator, `LTRUMER` to Rumer functions with the so-called "leading term" phase convention.

39.5 Recovering CASSCF CI vector and VB wavefunction

The appropriate MOLPRO records may be specified explicitly using the `START` directive (an alternative is the *vbdump* mechanism described in section 39.2.1):

`START`,*ci,vb,orb,trnint*;

ci: record name for the CASSCF CI vector. The CI vector must have been dumped previously using either of the `SAVE, NATORB, CANONICAL,` or `LOCALI` directives (see section 19.5.4). A default value for *ci* is determined from the most recent *vbdump* record(s).

Note that if the *ci* record is not found, only an energy-based optimization of the VB wavefunction can be carried out.

vb: record name for the valence bond orbitals and structure coefficients, as saved by a previous CASVB calculation. If the VB wavefunction was previously saved in the AO basis the orbitals will be projected onto the present active space (note that it is necessary to specify a record name for the molecular orbitals (*orb* below) for this to be possible).

orb: record name for the molecular orbitals defining the CASSCF wavefunction. This information is necessary if one wants to output the valence bond orbitals in the atomic orbital basis.

trnint: record name for the transformed CASSCF integrals. These are required for the energy-based criteria (i.e., if `CRIT,ENERGY` is specified), and can be saved inside *MULTI* by the `TRNINT` sub-command (see 19.8.7). The default record name, both here and in *MULTI*, is 1900.1.

39.6 Saving the VB wavefunction

`SAVE`,*vb,civb,vbao*;

vb: record name for VB wavefunction (default is first available record after 3200.2), i.e., orbitals and structure coefficients.

civb: record name for valence bond full CI vector defined in terms of the CASSCF MOs (default is 3300.2). Saving this vector is necessary for the calculation of further properties, geometry optimization, etc.

vbao: record name for valence bond wavefunction in the AO basis. Note that specifying *orb* in the `START` directive is a precondition for this keyword. It may be useful for plotting of orbitals, or for providing a guess to be used in the interpretation of a CASSCF solution employing a different active space.

It is normally advisable to use records on file 2 for *vb*, *civb*, and *vbao*.

39.7 Specifying a guess

`GUESS`;*key-1,...*;*key-2,...*;...

The `GUESS` keyword initiates the input of a guess for the valence bond orbitals and structure coefficients. *key-i* can be either `ORB`, `STRUC` or `READ`. These keywords modify the guess provided by the program, or specified by the `START` directive. It is thus possible to modify individual orbitals in a previous solution to construct the starting guess.

39.7.1 Orbital guess

`ORB`,*i*, c_1, c_2,... c_{mact};

Specifies a starting guess for valence bond orbital number *i*. The guess is specified in terms of the *mact* active MOs defining the CASSCF wavefunction. (Note that the definition of these MOs will depend on how the CI vector was dumped – i.e. which of the `SAVE, NATORB, CANONICAL,` or `LOCALI` directives was used (see section 19.5.4). Use of one of the three latter keywords is recommended.)

39.7.2 Guess for structure coefficients

`STRUC`,c_1, c_2,... c_{NVB};

Specifies a starting guess for the *NVB* structure coefficients. If this card is not provided, and no guess specified by `START`, the perfect-pairing mode of spin coupling is assumed for the spatial configuration having the least number of doubly occupied orbitals. Note that the definition of structures depends on the value of `SPINBASIS`. Doubly occupied orbitals occur first in all configurations, and the spin eigenfunctions are based on the singly occupied orbitals being in ascending order.

39.7.3 Read orbitals or structure coefficients

The `READ` keyword can take one of the following forms:

`READ`,`ORB`,*iorb1*[,`TO`,*iorb2*] [,`AS`,*jorb1*[,`TO`,*jorb2*]] [,`FROM`,*record*];

`READ`,`STRUC`,*istruc1*[,`TO`,*istruc2*] [,`AS`,*jstruc1*[,`TO`,*jstruc2*]] [,`FROM`,*record*];

`READ`,`ALL` [,`FROM`,*record*];

In this way a subset of orbitals and/or structure coefficients may be picked out from a previous calculation. Renumbering of orbitals or structures can be done using the “`AS`” construct as outlined above. If the VB wavefunction was previously saved in the AO basis, the orbitals will be projected onto the present active space (note that it is necessary to specify a record name for the molecular orbitals (*orb* in the `START` commmand) for this to be possible).

Default for *record* is the *vb* record name specified in keyword `START` (if applicable).

39.8 Permuting orbitals

`ORBPERM`,$i_1,\ldots,i_{mact}$;

Permutes the orbitals in the valence bond wavefunction and changes their phases according to $\phi'_j = \mathrm{sign}(i_j)\phi_{\mathrm{abs}(i_j)}$. The guess may be further modified using the `GUESS` keyword. Also the structure coefficients will be transformed according to the given permutation (note that the configuration list must be closed under the orbital permutation for this to be possible).

39.9 Optimization control

39.9.1 Optimization criterion

`CRIT`,*method*;

Specifies the criterion for the optimization. *method* can be `OVERLAP` or `ENERGY` (`OVERLAP` is default). The former maximizes the normalized overlap with the CASSCF wavefunction:

$$\max\left(\frac{\langle\Psi_{CAS}|\Psi_{VB}\rangle}{(\langle\Psi_{VB}|\Psi_{VB}\rangle)^{1/2}}\right)$$

and the latter simply minimizes the energy:

$$\min\left(\frac{\langle\Psi_{VB}|\hat{H}|\Psi_{VB}\rangle}{\langle\Psi_{VB}|\Psi_{VB}\rangle}\right).$$

39.9.2 Number of iterations

`MAXITER`,N_{iter};

Specifies the maximum number of iterations in the second order optimizations. Default is N_{iter}=50.

39.9.3 CASSCF-projected structure coefficients

`(NO)CASPROJ`;

With this keyword the structure coefficients are picked from the transformed CASSCF CI vector, leaving only the orbital variational parameters. For further details see the bibliography. This option may be useful to aid convergence.

39.9.4 Saddle-point optimization

`SADDLE`,*n*;

Defines optimization onto an n^{th}-order saddle point. See also T. Thorsteinsson and D. L. Cooper, Int. J. Quant. Chem. **70**, 637–50 (1998).

39.9.5 Defining several optimizations

More than one optimization may be performed in the same *CASVB* deck, by the use of OPTIM keywords:

OPTIM[;...;FINOPTIM];

The subcommands may be any optimization declarations defined in this section, as well as any symmetry or constraints specifications described in section 39.10. Commands given as arguments to OPTIM will be particular to this optimization step, whereas commands specified outside will act as default definitions for all subsequent OPTIM keywords.

If only one optimization step is required, the OPTIM keyword need not be specified.

When only a machine-generated guess is available, *CASVB* will attempt to define a sequence of optimization steps chosen such as to maximize the likelihood of successful convergence and to minimize CPU usage. To override this behaviour, simply specify one or more OPTIM cards.

39.9.6 Multi-step optimization

A loop over two or more optimization steps may be specified using:

ALTERN,*Niter*;...;FINALTER

With this specification the program will repeat the enclosed optimization steps until either all optimizations have converged, or the maximum iteration count, *Niter*, has been reached.

39.10 Point group symmetry and constraints

The problems associated with symmetry-adapting valence bond wavefunctions are considered, for example, in: T. Thorsteinsson, D. L. Cooper, J. Gerratt and M. Raimondi, Theor. Chim. Acta **95**, 131 (1997).

39.10.1 Symmetry operations

SYMELM,*label*,*sign*;

Initiates the definition of a symmetry operation referred to by *label* (any three characters). *sign* can be + or −; it specifies whether the total wavefunction is symmetric or antisymmetric under this operation, respectively. A value for *sign* is not always necessary but, if provided, constraints will be put on the structure coefficients to ensure that the wavefunction has the correct overall symmetry (note that the configuration list must be closed under the orbital permutation induced by *label* for this to be possible).

The operator is defined in terms of its action on the active MOs as specified by one or more of the keywords IRREPS, COEFFS, or TRANS (any other keyword will terminate the definition of this symmetry operator). If no further keyword is supplied, the identity is assumed for *label*. The alternative format SYMELM,*label*,*sign*;*key-1*,...;*key-2*,...;... may also be used.

39.10.2 The IRREPS keyword

IRREPS,i_1, i_2,...;

The list i_1, i_2,... specifies which irreducible representations (as defined in the CASSCF wavefunction) are antisymmetric with respect to the *label* operation. If an irreducible representation is not otherwise specified it is assumed to be symmetric under the symmetry operation.

39.10.3 The COEFFS keyword

`COEFFS`,i_1, i_2,...;

The list i_1, i_2,... specifies which individual CASSCF MOs are antisymmetric with respect to the *label* operation. If an MO is not otherwise specified, it is assumed to be symmetric under the symmetry operation. This specification may be useful if, for example, the molecule possesses symmetry higher than that exploited in the CASSCF calculation.

39.10.4 The TRANS keyword

`TRANS`,n_{dim}, $i_1, \ldots i_{n_{dim}}$, $c_{11}, c_{12}, \ldots c_{n_{dim}n_{dim}}$;

Specifies a general $\mathrm{n}_{dim} \times n_{dim}$ transformation involving the MOs $i_1, \ldots i_{n_{dim}}$, specified by the c coefficients. This may be useful for systems with a two- or three-dimensional irreducible representation, or if localized orbitals define the CASSCF wavefunction. Note that the specified transformation must always be orthogonal.

39.10.5 Symmetry relations between orbitals

In general, for a VB wavefunction to be symmetry-pure, the orbitals must form a representation (not necessarily irreducible) of the symmetry group. Relations between orbitals under the symmetry operations defined by `SYMELM` may be specified according to:

`ORBREL`,i_1, i_2, *label1, label2,*...;

Orbital i_1 is related to orbital i_2 by the sequence of operations defined by the *label* specifications (defined previously using `SYMELM`). The operators operate right to left. Note that i_1 and i_2 may coincide. Only the minimum number of relations required to define all the orbitals should be provided; an error exit will occur if redundant `ORBREL` specifications are found.

39.10.6 The SYMPROJ keyword

As an alternative to incorporating constraints, one may also ensure correct symmetry of the wavefunction by use of a projection operator:

`(NO)SYMPROJ`[,$irrep_1$,$irrep_2$,...];

The effect of this keyword is to set to zero coefficients in unwanted irreducible representations. For this purpose the symmetry group defined for the CASSCF wavefunction is used (always a subgroup of D_{2h}). The list of irreps in the command specifies which components of the wavefunction should be kept. If no irreducible representations are given, the current wavefunction symmetry is assumed. In a state-averaged calculation, all irreps are retained for which a non-zero weight has been specified in the wavefunction definition. The `SYMPROJ` keyword may also be used in combination with constraints.

39.10.7 Freezing orbitals in the optimization

FIXORB,i_1, i_2,. . . ;

This command freezes the orbitals specified in the list i_1, i_2,. . . to that of the starting guess. Alternatively the special keywords ALL or NONE may be used. These orbitals are eliminated from the optimization procedure, but will still be normalized and symmetry-adapted according to any ORBREL keywords given.

39.10.8 Freezing structure coefficients in the optimization

FIXSTRUC,i_1, i_2,. . . ;

Freezes the coefficients for structures i_1, i_2,. . . . Alternatively the special keywords ALL or NONE may be used. The structures are eliminated from the optimization procedure, but may still be affected by normalization or any symmetry keywords present.

39.10.9 Deleting structures from the optimization

DELSTRUC,i_1, i_2,. . . ,[ALL],[NONE];

Deletes the specified structures from the wavefunction. The special keywords ALL or NONE may be used. A structure coefficient may already be zero by symmetry (as defined by SYMELM and ORBREL), in which case deleting it has no effect.

39.10.10 Orthogonality constraints

ORTHCON;*key-1,*. . . ;*key-2,*. . . ;. . .

The ORTHCON keyword initiates the input of orthogonality constraints between pairs of valence bond orbitals. The sub-keywords *key-i* can be one of ORTH, PAIRS, GROUP, STRONG or FULL as described below. Orthogonality constraints should be used with discretion. Note that orthogonality constraints for an orbital generated from another by symmetry operations (using the ORBREL keyword) cannot in general be satisfied.

ORTH,i_1, i_2, . . . ;

Specifies a list of orbitals to be orthogonalized. All overlaps between pairs of orbitals in the list are set to zero.

PAIRS,i_1, i_2, . . . ;

Specifies a simple list of orthogonalization pairs. Orbital i_1 is made orthogonal to i_2, i_3 to i_4, etc.

GROUP,*label*,i_1, i_2, . . . ;

Defines an orbital group to be used with the ORTH or PAIRS keyword. The group is referred to by *label* which can be any three characters beginning with a letter a–z. Labels defining different groups can be used together or in combination with orbital numbers in ORTH or PAIRS. i_1, i_2, . . . specifies the list of orbitals in the group. Thus the combination GROUP,AZZ,1,2; GROUP,BZZ,3,4; ORTH,AZZ,BZZ; will orthogonalize the pairs of orbitals 1-3, 1-4, 2-3 and 2-4.

STRONG;

This keyword is short-hand for strong orthogonality. The only allowed non-zero overlaps are between pairs of orbitals ($2n-1$, $2n$).

`FULL`;

This keyword is short-hand for full orthogonality. This is mainly likely to be useful for testing purposes.

39.11 Wavefunction analysis

39.11.1 Spin correlation analysis

`(NO)SCORR`;

With this option, expectation values of the spin operators $(\hat{s}_\mu + \hat{s}_\nu)^2$ are evaluated for all pairs of μ and ν. Default is `NOSCORR`. The procedure is described by: G. Raos, J. Gerratt, D. L. Cooper and M. Raimondi, Chem. Phys. **186**, 233–250 (1994); ibid, 251–273 (1994); D. L. Cooper, R. Ponec, T. Thorsteinsson and G. Raos, Int. J. Quant. Chem. **57**, 501–518 (1996).

At present this analysis is only implemented for spin-coupled wavefunctions.

39.11.2 Printing weights of the valence bond structures

For further details regarding the calculation of weights in CASVB, see T. Thorsteinsson and D. L. Cooper, J. Math. Chem. **23**, 105-26 (1998).

`VBWEIGHTS`,*key1*,*key2*,...

Calculates and outputs weights of the structures in the valence bond wavefunction Ψ_{VB}. *key* specifies the definition of nonorthogonal weights to be used, and can be one of:

`CHIRGWIN`	Evaluates Chirgwin-Coulson weights (see: B. H. Chirgwin and C. A. Coulson, Proc. Roy. Soc. Lond. **A201**, 196 (1950)).
`LOWDIN`	Performs a symmetric orthogonalization of the structures and outputs the corresponding weights.
`INVERSE`	Outputs “inverse overlap populations” as in G. A. Gallup and J. M. Norbeck, Chem. Phys. Lett. **21**, 495–500 (1973).
`ALL`	All of the above.
`NONE`	Suspends calculation of structure weights.

The commands `LOWDIN` and `INVERSE` require the overlap matrix between valence bond structures, and some computational overhead is thus involved.

39.11.3 Printing weights of the CASSCF wavefunction in the VB basis

For further details regarding the calculation of weights in *CASVB*, see T. Thorsteinsson and D. L. Cooper, J. Math. Chem. **23**, 105-26 (1998).

`CIWEIGHTS`,*key1*,*key2*,... [,$N_{\rm conf}$];

Prints weights of the CASSCF wavefunction transformed to the basis of nonorthogonal VB structures. For the *key* options see `VBWEIGHTS` above. Note that the evaluation of inverse overlap weights involves an extensive computational overhead for large active spaces. Weights are given for the total CASSCF wavefunction, as well as the orthogonal complement to Ψ_{VB}. The default for the number of configurations requested, N_{conf}, is 10. If $N_{\text{conf}}=-1$ all configurations are included.

39.12 Controlling the amount of output

`PRINT`,i_1, i_2,...;

Each number specifies the level of output required at various stages of the execution, according to the following convention:

-1	No output except serious, or fatal, error messages.
0	Minimal output.
1	Standard level of output.
2	Extra output.

The areas for which output can be controlled are:

i_1	Print of input parameters, wavefunction definitions, etc.
i_2	Print of information associated with symmetry constraints.
i_3	General convergence progress.
i_4	Progress of the 2nd order optimization procedure.
i_5	Print of converged solution and analysis.
i_6	Progress of variational optimization.
i_7	Usage of record numbers on file 2.

For all, the default output level is +1. If $i_5 \geq 2$ VB orbitals will be printed in the AO basis (provided that the definition of MOs is available); such output may be especially useful for plotting of orbitals.

39.13 Further facilities

Calculations can also be performed for various types of direct product wavefunctions and/or with strictly localized orbitals. Details are available from the authors. These facilities will be documented in a later release.

39.14 Service mode

`SERVICE`;
This keyword takes precedence over any others previously defined to *CASVB*. It provides simple facilities for retrieving orbital coefficients and VB structure coefficients. It should not be used during a run of *CASVB* that has been invoked from inside *MULTI*.

`START`,*record.file*;
Coefficients are taken from *record.file*. The default value is *2100.2*.

`WRITE`,*iwrite*;
Vectors in the symmetry orbital basis are written to channel *iabs(iwrite)*. The default action is

to write these vectors to the standard output. If *iwrite* is negative, then the vectors are instead written to a binary file as a single record.

`SPECIAL`,*idim1,idim2,idim3,idim4*;
If present, this keyword must come last. The program attempts to retrieve from *record.file* a vector of length *idim1*idim2+idim3*, after first skipping *idim4* elements. The vector is written according to the setting of *iwrite*. (Default *idim* values are zero.)

39.15 Examples

```
***, ch2                        ! A1 singlet state
geometry={angstrom
c
h1,c,1.117
h2,c,1.117,h1,102.4}
int
hf
{multi;occ,4,1,2;closed,1       ! 6 in 6 CASSCF
natorb,,ci,save=3500.2;vbdump}
{casvb                          ! Overlap-based VB using
save,3200.2}                    ! the spin-coupled wavefunction
{casvb                          ! Energy-based VB calculation
start,,3200.2;save,3220.2
crit,energy}
{multi;occ,4,1,2;closed,1       ! Fully variational VB calculation
{vb;start,,3220.2;save,3240.2;print,,,,,2}}
---
```

```
memory,4,m
***,n2s2 (model a)              ! Variational calculation for N2S2.
geometry={x,y,z;
a1,n,-2.210137753,0,0;          ! NOTE: other choices of active space
a2,n,+2.210137753,0,0;          ! give alternative (competing) models.
a3,s,0,-2.210137753,0;
a4,s,0,+2.210137753,0}
basis=VTZ;
cartesian
{hf;wf,46,1}
{multi;occ,7,4,5,2,4,2,2,0;closed,7,4,5,2,1,0,1,0; natorb,,ci,save=3500.2}
{multi;occ,7,4,5,2,4,2,2,0;closed,7,4,5,2,1,0,1,0; vb}
---
```

```
***, lih                        ! Fully variational VB calculation
r=2.8,bohr                      ! and geometry optimization.
basis={
s,1,921.300000,138.700000,31.940000,9.353000,3.158000,1.157000;
k,1.6,0.001367,0.010425,0.049859,0.160701,0.344604,0.425197;
s,1,0.444600,0.076660,0.028640;
p,1,1.488000,0.266700,0.072010,0.023700;
k,1.2,0.038770,0.236257;
s,2,13.36,2.013,0.4538,.1233;
```

```
k,1.2,0.032828,0.231204;}
geometry={li;h,li,r}
int;
{hf;wf,4,1}
{multi
occ,4,0,0,0
closed,0,0,0,0
natorb,,ci,save=3500.2}
{multi;maxiter,20
vb}
optg
---
```

40 SPIN-ORBIT-COUPLING

40.1 Introduction

Spin-orbit matrix elements and eigenstates can be computed using either the Breit-Pauli (BP) operator or spin-orbit pseudopotentials (ECPs). The *state-interacting* method is employed, which means that the spin-orbit eigenstates are obtained by diagonalizing $\hat{H}_{el} + \hat{H}_{SO}$ in a basis of eigenfunctions of $\hat{H}_{el}$. The full Breit-Pauli SO-operator can be used only for MCSCF wavefunctions. For MRCI wavefunctions, the full BP operator is used for computing the matrix elements between *internal configurations* (no electrons in external orbitals), while for contributions of external configurations a mean-field one-electron fock operator is employed. The error caused by this approximation is usually smaller than 1 cm^{-1}.

The program allows either the computation of individual spin-orbit matrix elements for a given pair of states, or the automatic setting-up and diagonalization of the whole matrix for a given set of electronic states. In the latter case, matrix elements over one-electron operators are also computed and transformed to the spin-orbit eigenstates (by default, the dipole matrix elements are computed; other operators can be specified on the `GEXPEC` or `EXPEC` cards, see section 6.13). Since it may be often sufficient to compute the spin-orbit matrix elements in a smaller basis than the energies, it is possible to replace the energy eigenvalues by precomputed values, which are passed to the spin-orbit program by the MOLPRO variable `HLSDIAG`.

40.2 Calculation of SO integrals

The one-and two-electron spin-orbit integrals over the BP Hamiltonian can be precomputed and stored on disk using the command

`LSINT` [,`X`] [,`Y`] [,`Z`] [,`ONECENTER`] [;`TWOINT`,*twoint*;] [;`PREFAC`,*prefac*;]

`X`, `Y`, and `Z` specify the components to be computed. If none of these is given, all three are evaluated. The advantage of precomputing the integrals is that they can then be used in any number of subsequent SO calculations, but this may require a large amount of disk space (note that there are 6 times as many integrals as in an energy calculation). If the `LSINT` card is not given, the integrals are computed whenever needed. The keyword `ONECENTER` activates the one-center approximation for one- and two- electron spin-orbit integrals. This can reduce drastically the computing time for large molecules. `TWOINT` and `PREFAC` can be used to control the accuracy of spin-orbit integrals. These thresholds are similar to `TWOINT` and `PREFAC` for standard integrals. The default value for PREFAC is TWOINT/100, and the default value for TWOINT is 10^{-7}. In the case when no integrals are precomputed, these thresholds can be specified as options for HLSMAT or TRANLS cards, see below.

The input for spin-orbit ECPs is described in section 12. Of course, in ECP-LS calculations the LSINT card is not needed.

40.3 Calculation of individual SO matrix elements

Individual spin-orbit matrix elements can be computed within the MRCI program using

TRANLS,*record1.file, record2.file, bra2ms, ket2ms, lsop*;

where

record1.file	Record holding the bra-wavefunction.
record2.file	Record holding the ket-wavefunction. Both records must have been generated using the SAVE directive of the MRCI program.
bra2ms	$2 \times M_S$ value of the bra-wavefunction.
ket2ms	$2 \times M_S$ value of the ket-wavefunction.
lsop	Cartesian component of the Spin-orbit Hamiltonian. This can be one of LSX, LSY, or LSZ in all electron calculations, and ECPLSX, ECPLSY, or ECPLSZ in ECP calculations. Starting from the MOLPRO version 2008.1, more types are available which control the approximation level. These are described in section 40.4.

Since the spin-orbit program is part of the MRCI program, the TRANLS card must be preceded by a [MR]CI card. For the case that the matrix elements are computed for MCSCF wavefunctions, one has to recompute and save the CI-vectors using the MRCI program (see chapter 20), using the NOEXC directive to avoid inclusion of any further excitations out of the MCSCF reference function. If in the MRCI step several states of the same symmetry are computed simultaneously using the STATE directive, the matrix elements are computed for all these states. Note that the OCC and CLOSED cards must be the same for all states used in a TRANLS calculation.

The selection rules for the M_S values are $\Delta M_S = \pm 1$ for the LSX and LSY operators, and $\Delta M_S = 0$ for the LSZ operator. Note that $2M_S$ has to be specified, and so the selection rules applying to the difference of the input values are 0 or 2.

In all-electron SO calculations the value of the calculated spin-orbit matrix element is saved (in atomic units) in the MOLPRO variables TRLSX, TRLSY and TRLSZ for the x, y, and z components respectively. For ECP-LS calculations the variables TRECPLSX, TRECPLSY, and TRECPLSZ are used. Note that for imaginary matrix elements (i.e., for the x and z components of the SO Hamiltonian) the matrix elements are imaginary and the stored real values have to be multiplied by i. If matrix elements for several states are computed, all values are stored in the respective variable-arrays with the bra-states running fastest.

40.4 Approximations used in calculating spin-orbit integrals and matrix elements

Recently, more sophisticated approximations were introduced to simplify spin-orbit calculations for larger molecules. These are controlled by specifying the spin-orbit operator type *lsop* as follows (we omit suffixes X, Y, Z which specify the component):

LS	Standard spin-orbit calculations.
ALS	The one-center approximation is used for one- and two-electron spin-orbit integrals.
FLS	The effective Fock-matrix approximation is used for the internal part too.
AFLS|AMFI	The one-center approximation is used for one- and two-electron spin-orbit integrals, and the effective Fock-matrix approximation for the internal part.
ECPLS	Effective core potentials are used for all atoms at which they are defined; contributions of all other atoms are neglected (see below).

In case that the effective Fock matrix is used for all contributions, and no spin-orbit integrals are pre-calculated and stored on disk (i.e., the LSINT command is not given), the Fock matrices are evaluated in direct mode and no integrals are stored on disk. When this is combined with the one-center approximation (AMFI), the computing and I/O times are drastically reduced, and this makes spin-orbit calculations quite fast even for larger molecules.

Also, the treatment of ECP-type of spin-orbit interaction has been changed and now allows for treating both ECP and non-ECP atoms in one calculation. Thus, in molecules containing both heavy and light atoms, the heavy atoms can be described using ECPs and the light atoms using all-electron basis sets. If the operator type is LS, ALS, FLS, or AFLS, then for the atoms having an ECP spin-orbit operator defined in the basis input the ECP operator is used, while the full BP-operator is used for all other atoms (couplings are neglected). Both one-center and AMFI approximations can be used in this case. If, on the other hand, one specifies the operator type as ECPLS, then the behavior is the same as in the previous versions, i.e., only the ECP contributions are considered and the contributions from all other atoms are neglected.

40.5 Calculation and diagonalization of the entire SO-matrix

HLSMAT,*type*, *record1, record2, record3,* . . .

Computes the entire SO matrix and diagonalizes it using all states which are contained in the records *record1, record2, record3,* . . . . All records must have been generated using the SAVE directive of the MRCI program. *type* may be either LS for Breit-Pauli calculations, or ECP for ECP-LS calculations. By default, the eigenvalues and dipole transition matrix elements between the ground and excited states are printed.

As with the TRANLS card, the HLSMAT is recognized only by the MRCI program and must be preceded by a CI card. Also, the OCC and CLOSED cards must be the same for all states used in a HLSMAT calculation.

40.6 Modifying the unperturbed energies

Often it may be sufficient to compute the spin-orbit matrix elements in a smaller basis or at a lower computational level than the energies. It is therefore possible to replace the energy eigenvalues by precomputed values, which are passed to the spin-orbit program by the MOLPRO variable HLSDIAG. The energy values in HLSDIAG must be in exactly the same order as the states in the records given on the HLSMAT card. Before any spin-orbit calculation, the variable HLSDIAG must either be undefined or cleared (then the original energies are used), or must contain exactly the number of energies as the number of states treated in the subsequent spin-orbit calculation (use CLEAR,HLSDIAG to clear any previous values in the variable). It is the user's responsibility that the order of the energies in HLSDIAG is correct!

40.6.1 Print Options for spin-orbit calculations

PRINT,*option*$_1$=*value*$_1$, *option*$_2$=*value*$_2$,...

where option can be

HLS	HLS=-1 only the SO energies and transition matrix elements between ground and excited states are printed (default). HLS$\geq$ 0: The SO matrix is printed. HLS$\geq$ 1: The property matrices are printed. HLS$\geq$ 2: The individual matrix elements are printed (same as OPTION,MATEL). HLS$\geq$ 3: Debugging information is printed.
VLS	VLS=-1: No print of eigenvectors (default). VLS$\geq$ 0: The eigenvectors are printed.

40.6.2 Options for spin-orbit calculations

Some options can be set using the OPTION directive (in any order)

OPTIONS [,WIGNER=*value*] [,HLSTRANS=*value*] [,MATEL=*value*]

where

WIGNER	This option determines whether the Wigner-Eckart theorem should be used when the SO matrix is determined. WIGNER=1 (default) uses the theorem, WIGNER=0 calculates each SO matrix element individually. This option is needed for test purposes only.
HLSTRANS	This option determines whether a SO matrix calculation should be performed in the not spin-symmetry adapted basis set (HLSTRANS=0), in the spin-symmetry adapted basis set (HLSTRANS=1, default) or with both basis sets (HLSTRANS=2). At present, symmetry adaption can only be performed for triplet states, where the following notation is used to indicate the symmetry adapted spin functions: $\|S,M_S\rangle_+ = \frac{1}{\sqrt{2}}(\|S,M_S\rangle + \|S,-M_S\rangle)$, $\|S,M_S\rangle_- = \frac{1}{\sqrt{2}}(\|S,M_S\rangle - \|S,-M_S\rangle)$. If only singlet and triplet states are considered, the spin-orbit matrix is blocked according to double-group symmetry and the eigenvalues for each each block are printed separately. In all other cases the HLSTRANS option is ignored.
MATEL	If the entire SO matrix is calculated using HLSMAT, the individual matrix elements are normally not shown. When the option MATEL=1 is given, the individual matrix elements and the contributions of the internal and external configuration classes are printed.

40.7 Examples

40.7.1 SO calculation for the S-atom using the BP operator

```
***,SO calculation for the S-atom
geometry={s}
basis={spd,s,vtz}                                   !use uncontracted basis

{rhf;occ,3,2,2,,2;wf,16,4,2}                        !rhf for 3P state

{multi                                              !casscf
wf,16,4,2;wf,16,6,2;wf,16,7,2;wf,16,1,0;state,3;  !1D and 1S states
wf,16,4,0;wf,16,6,0;wf,16,7,0}                      !3P states

{ci;wf,16,1,0;save,3010.1;state,3;noexc}            !save casscf wavefunctions using mrci
{ci;wf,16,4,0;save,3040.1;noexc}
{ci;wf,16,6,0;save,3060.1;noexc}
{ci;wf,16,7,0;save,3070.1;noexc}
{ci;wf,16,4,2;save,3042.1;noexc}
{ci;wf,16,6,2;save,3062.1;noexc}
{ci;wf,16,7,2;save,3072.1;noexc}

{ci;wf,16,1,0;save,4010.1;state,3}                  !mrci calculations for 1D, 1S states
ed=energy(1)                                        !save energy for 1D state in variable ed
es=energy(3)                                        !save energy for 1S state in variable es
{ci;wf,16,4,2;save,4042.1}                          !mrci calculations for 3P states
ep=energy                                           !save energy for 3P state in variable ep
{ci;wf,16,6,2;save,4062.1}                          !mrci calculations for 3P states
{ci;wf,16,7,2;save,4072.1}                          !mrci calculations for 3P states
text,only triplet states, casscf

lsint                                               !compute so integrals

text,3P states, casscf
{ci;hlsmat,ls,3042.1,3062.1,3072.1}                 !Only triplet states, casscf

text,3P states, mrci
{ci;hlsmat,ls,4042.1,4062.1,4072.1}                 !Only triplet states, mrci

text,3P, 1D, 1S states, casscf
{ci;hlsmat,ls,3010.1,3040.1,3060.1,3070.1,3042.1,3062.1,3072.1}     !All states, casscf

text,only triplet states, use mrci energies and casscf SO-matrix elements
hlsdiag=[ed,ed,es,ed,ed,ed,ep,ep,ep]              !set variable hlsdiag to mrci energies
{ci;hlsmat,ls,3010.1,3040.1,3060.1,3070.1,3042.1,3062.1,3072.1}
```

http://www.molpro.net/info/current/examples/s_so.com

40.7.2 SO calculation for the I-atom using ECPs

```
***,I
memory,5,M;
gprint,orbitals,civector,basis;
gthresh,energy=1.d-8,coeff=1.d-8;
geometry={I};

basis={
!
! Iodine-ECP (Dirac-Fock) with SO-coupling
!
ecp,I,46,4,3;
1; 2,   1.00000000,       0.00000000;                                   ! lokal term = 0
2; 2,   3.50642001,      83.09814545; 2,   1.74736492,       5.06370919;    ! s-terme
4; 2,   2.99860773, 1/3* 81.88444526; 2,   3.01690894, 2/3* 83.41280402;    ! p-terms with wei
   2,   1.59415934, 1/3*  2.32392477; 2,   1.19802939, 2/3*  2.72079843;
4; 2,   1.03813792, 2/5*  6.40131754; 2,   1.01158599, 3/5*  6.21328827;    ! d-terms with wei
   2,   2.04193864, 2/5* 19.11604172; 2,   1.99631017, 3/5* 19.08465909;
4; 2,   2.64971585,-3/7* 24.79106489; 2,   2.75335574,-4/7* 24.98147319;    ! f-terms with wei
   2,   0.49970082,-3/7*  0.27936581; 2,   0.79638982,-4/7*  0.70184261;
4; 2,   2.99860773,-2/3* 81.88444526; 2,   3.01690894, 2/3* 83.41280402;    ! ECP-SO for p-ter
   2,   1.59415934,-2/3*  2.32392477; 2,   1.19802939, 2/3*  2.72079843;
4; 2,   1.03813792,-2/5*  6.40131754; 2,   1.01158599, 2/5*  6.21328827;    ! ECP-SO for d-ter
   2,   2.04193864,-2/5* 19.11604172; 2,   1.99631017, 2/5* 19.08465909;
4; 2,   2.64971585, 2/7* 24.79106489; 2,   2.75335574,-2/7* 24.98147319;    ! ECP-SO for f-ter
   2,   0.49970082, 2/7*  0.27936581; 2,   0.79638982,-2/7*  0.70184261;
!
! Iodine-basis
!
s,I,0.2027624,0.4080619,0.8212297,1.6527350,3.3261500;
c,1.5,-0.4782372,-0.5811680,0.2617769,0.4444120,-0.1596560;
s,I,0.05,0.1007509;
p,I,0.2027624,0.4080619,0.8212297,1.6527350,3.3261500;
c,1.5,0.4251859,0.2995618,0.0303167,-0.2064228,0.0450858;
p,I,0.05,0.1007509,0.01;  ! diffuse p-Funktion wegen evt. neg. Part.Ldg
d,I,0.2,0.4;
f,I,0.3;
}


{hf;occ,1,1,1,,1;wf,7,5,1}       !scf for 2Pz
{multi;occ,1,1,1,,1;             !casscf with minmal active space
wf,7,2,1;wf,7,3,1;wf,7,5,1}      !average 2P states
{ci;wf,7,2,1;noexc;save,5000.2}  !save casscf vector for 2Px state
{ci;wf,7,3,1;noexc;save,5100.2}  !save casscf vector for 2Py state
{ci;wf,7,5,1;noexc;save,5200.2}  !save casscf vector for 2Pz state
{ci;wf,7,2,1;save,6000.2}        !mrci for 2Px state
{ci;wf,7,3,1;save,6100.2}        !mrci for 2Py state
{ci;wf,7,5,1;save,6200.2}        !mrci for 2Pz state

{multi;occ,1,2,2,,2              !casscf with larger active space
wf,7,2,1;wf,7,3,1;wf,7,5,1}      !average 2P states
{ci;wf,7,2,1;noexc;save,5010.2}
{ci;wf,7,3,1;noexc;save,5110.2}
{ci;wf,7,5,1;noexc;save,5210.2}
{ci;wf,7,2,1;save,6010.2}
{ci;wf,7,3,1;save,6110.2}
{ci;wf,7,5,1;save,6210.2}

text,casscf,  occ,1,1,1,,1
{ci;hlsmat,ecp,5000.2,5100.2,5200.2}   !do spin-orbit calculations
text,casscf,  occ,1,2,2,,2
{ci;hlsmat,ecp,5010.2,5110.2,5210.2}

text,mrci, occ,1,1,1,,1
{ci;hlsmat,ecp,6000.2,6100.2,6200.2}
text,mrci, occ,1,2,2,,2
{ci;hlsmat,ecp,6010.2,6110.2,6210.2}
```

41 ENERGY GRADIENTS

41.1 Analytical energy gradients

MOLPRO uses two different gradient programs:

The CADPAC gradient program is based on the CADPAC integral routines by R. D. Amos. Currently, this program works for closed shell SCF, high spin RHF, and (state averaged) MCSCF. In the MCSCF case the wavefunction must either be fully optimized, or frozen core orbitals must be taken from a closed-shell SCF calculation (but this does not work in the case of state-averaged MCSCF). Note that CADPAC does not work with generally contracted basis functions.

The ALASKA gradient program is based on the SEWARD integral routines by R. Lindh. It allows the calculation of gradients of generally contracted basis functions for closed shell SCF, open shell RHF, UHF, RKS, UKS, MCSCF, MP2, LMP2, DF-LMP2, QCISD, QCISD(T), and RS2 (CASPT2). Gradients for state averaged MCSCF wave functions can be evaluated using the RS2 gradient program, see section 41.1.5. For details about CASPT2 gradients, see section 21.7.

By default, the program uses ALASKA gradients whenever possible. However, it is possible to force the use of a particular gradient program by defining the variable `GRADTYP` before calling the gradient program:

```
GRADTYP=ALASKA
GRADTYP=CADPAC
```

The gradient program is called using the `FORCE` command:

```
FORCE
```

Normally, the `FORCE` command is not needed, since geometry optimizations should be performed using the `OPTG` procedure. An exception is the optimization of counterpoise corrected energies, which requires several force calculations (cf. section 42.4.7).

If no further data cards are given, the default is to evaluate the gradient for the last optimized wavefunction. In this case no further input is needed for ordinary gradient cases (the program remembers the records on which the wavefunction information is stored). An exception is the unusual case that several different `CPMCSCF` calculations have been formed in a previous MCSCF calculation. In this case the `SAMC` directive must be used to select the desired record. If analytical gradients are not available for the last wavefunction, the gradient is computed numerically. For more details regarding numerical energy gradients see section 41.2.

41.1.1 Adding gradients (`ADD`)

`ADD`,*factor*,[`NOCHECK`];

If this card is present, the current gradient and energy are added to the previous ones using the given factor. This is useful for the optimization of counterpoise corrected energies (cf. 42.4.7). By default, the program will stop with an error message unless `NOORIENT` has been specified in the geometry input. This behaviour can be disabled by the `NOCHECK` option. This option should only be given if all gradients which are added are evaluated at exactly the same nuclear geometry; otherwise wrong results could result due to unintended rotations of the system.

41.1.2 Scaling gradients (`SCALE`)

`SCALE`,*factor*;

If this card is present, the current gradient and energy are scaled by the give factor. This is sometimes useful for the optimization of counterpoise corrected energies (cf. 42.4.7).

41.1.3 Defining the orbitals for SCF gradients (ORBITAL)

ORBITAL,*record.file*;

In the SCF case, *record.file* specifies the location of the orbitals, which are used for constructing density matrices, etc. This card is only needed if the SCF for which the gradient is to be computed was not the most recent energy calculation.

For MCSCF wavefunctions, the ORBITAL card is not needed, because the location of the orbitals is stored in the MCSCF dump record.

41.1.4 MCSCF gradients (MCSCF)

MCSCF,*record.file*;

Triggers code for MCSCF gradient. *record.file* specifies the location of information dumped from the MCSCF program, using a SAVE,GRD=*recmc.filmc* card. This card is not needed if the FORCE command appears directly after the corresponding MCSCF input, since the program automatically remembers where the MCSCF information was stored. The same is true if OPTG is used.

41.1.5 State-averaged MCSCF gradients with SEWARD

SA-MCSCF gradients can be computed using segmented or generally contracted basis sets using SEWARD and the RS2 gradient program. The NOEXC directive has to be used in the RS2 input, but no CPMCSCF card is required in MULTI. The RS2 gradient program does the CP-MCSCF automatically.

Example: compute SA-CASSCF gradients for $^2\Pi$ and $^2\Sigma^+$ state of OH.

```
geometry={o;h,o,r}
r=1.83
{multi;wf,9,2,1;wf,9,3,1;wf,9,1,1}       !state averaged casscf for X(2PI) and A(2SIGMA)
{rs2;noexc;wf,9,1,1}                     !compute A(2SIGMA) energy
forces                                   !energy gradient for A(2SIGMA) state
{rs2;noexc;wf,9,2,1}                     !compute A(2PI) energy
forces                                   !energy gradient for A(2PI) state
```

http://www.molpro.net/info/current/examples/oh_samcforce.com

Without the NOEXC directive, the RS2 (CASPT2) gradient would be evaluated, using the state-averaged orbitals.

41.1.6 State-averaged MCSCF gradients with CADPAC

Normally, no further input is required for computing gradients for state-averaged MCSCF when CADPAC is used. Note, however, that a CPMCSCF,GRAD,*state* directive is required in the SA-MCSCF calculation (see Section 19.9). The gradients are then computed automatically for the *state* specified on the CPMCSCF card. The same is true for difference gradients (CPMCSCF,DGRAD,*state1,* *state2*) and non-adiabatic coupling matrix elements (CPMCSCF,NACM,*state1, state2*). It is possible to do several coupled-perturbed MCSCF calculations one after each other in the same MCSCF. In this case FORCE would use the last solution by default. The information from the CPMCSCF is passed to the FORCE program in a certain records (default 5101.1, 5102.1, ...). If several CPMCSCF calculations are performed in the same MCSCF, several such records may be present, and a particular one can be accessed in the FORCE program using the SAMC directive:

SAMC,*record*.

An alias for SAMC is CPMC. For compatibility with earlier versions one can also use

NACM,*record*

for non-adiabatic couplings or

DEMC,*record*

for difference gradients.

Example:

```
multi;
....
state,3
cpmcscf,nacm,1.1,2.1,save=5101.1   !do cpmcscf for coupling of states 1.1 - 2
cpmcscf,nacm,1.1,3.1,save=5102.1   !do cpmcscf for coupling of states 1.1 - 3
cpmcscf,nacm,2.1,3.1,save=5103.1   !do cpmcscf for coupling of states 2.1 - 3

force;samc,5101.1;                 !compute NACME for states 1.1 - 2.1
force;samc,5102.1;                 !compute NACME for states 1.1 - 3.1
force;samc,5103.1;                 !compute NACME for states 2.1 - 3.1
```

See also test job lif_nacme.test.

41.1.7 Non-adiabatic coupling matrix elements (NACM)

see Section 41.1.6.

41.1.8 Difference gradients for SA-MCSCF (DEMC)

see Section 41.1.6.

41.1.9 Example

```
***, Calculate Gradients for Water
alpha=104 degree           !set geometry parameters
r=1 ang
geometry={O;               !define z-matrix
          H1,o,r;
          H2,o,r,H1,alpha}
basis=vdz                  !basis set
hf                         !do scf
forces                     !compute gradient for SCF
mp2                        !mp2 calculation
forces                     !mp2 gradients
multi                      !casscf calculation
forces                     !casscf gradient
```

http://www.molpro.net/info/current/examples/h2o_forces.com

41.2 Numerical gradients

It is possible to compute gradients by finite differences using

`FORCE,NUMERICAL,`*options*

Numerical gradients are computed automatically if no analytical gradients are available for the last energy calculation. By default, no further input are needed, and the gradient will be computed for the last energy calculation. The following options can be given on the `FORCE` command or on subsequent directives (see subsequent sections):

`STARTCMD=`*command* The input between *command* and the current `FORCE` command defines the energy calculation for which the gradient is computed. This input section is executed for each displacement.

`PROC=`*procname* specifies a procedure to be executed for each displacement. This must define a complete energy calculation and must not contain gradient or Hessian calculations.

`VARIABLE=`*varname* Compute the gradient of the value of variable *varname*. This implies numerical gradients. The variable must be set in the corresponding energy calculation.

`COORD=ZMAT|CART|3N` coordinates with respect to which the gradient is evaluated. See section 41.2.1 for more information.

`DISPLACE=ZMAT|SYM|UNIQUE|CART`
Displacement coordinates to be used for numerical gradient. The default is `ZMAT` if the geometry is given as a zmatrix which depends on variables, and `SYM` (symmetrical displacement coordinates) otherwise. See section 41.2.1 for more information.

`SYMMETRY=AUTO|NOSYM` Symmetry to be used in wavefunction calculations of numerical gradients. This option is only relevant if `DISPLACE=UNIQUE|CART`. If `AUTO` is given, the maximum possible symmetry is used for each displacement. This implies that the energy is independent of the symmetry used. Note that this often not the case in MRCI or CASPT2 calculations. The option can also not be used in local correlation calculations.

`AUTO` (logical). Same as `SYMMETRY=AUTO`

ZMAT	(logical). Same as COORD=ZMAT
OPT3N	(logical). Same as COORD=3N
RSTEP=*rstep*	Step length for distances in numerical gradient calculations (in bohr). The default is 0.01.
DSTEP=*dstep*	Step length for symmetrical displacements (in bohr). The default is 0.01.
ASTEP=*astep*	Step length for angles in numerical gradient calculations (in degree). The default is 1.
CENTRAL	(logical). Use 2-point central formula; needs $2M$ energy calculations for M degrees of freedom.
FORWARD	(logical). Use forward gradients (needs only $M+1$ energy calculations, but less accurate)
FOURPOINT	(logical). Use 4-point formula for accurate numerical gradient; needs $4M$ energy calculations.
NUMERICAL	(logical). Force the use of numerical gradients, even if gradients are available.
VARSAV	(logical). Save gradient in variables GRADX, GRADY, GRADZ.

Example

```
hf
ccsd(t)
forces,numerical
```

The program will then automatically repeat HF and CCSD(T) at as many geometries as needed for evaluating the gradient. This is equivalent to

```
hf
ccsd(t)
forces,numerical,startcmd=hf
```

or, using a procedure

```
forces,numerical,proc=runccsdt
...
runccsdt={
hf
ccsd(t)}
```

41.2.1 Choice of coordinates (COORD)

By default, the numerical gradients are computed relative to all variables on which the z-matrix depends. If the z-matrix depends on no variables or on $3N$ variables, the gradient is computed for all $3N$ coordinates and symmetrical displacement coordinates are used to evaluate the gradient. This yields the minimum computational effort.

These defaults can be modified using the COORD directive:

COORD,*coord_type*,[*displacement_type*]

where *coord_type* can be one of the following:

`ZMAT` Compute the numerical gradients for all variables on which the geometry depends (default).

`3N` or `CART` Compute the gradients for all 3*N* nuclear coordinates. This is the default if the z-matrix does not depend on variables or if the `xyz` input format is used. If this option is used and the original geometry is given in z-matrix form, the z-matrix is lost.

The specification of *displacement_type* is optional and only affects the numerical calculation of the gradient for 3*N* coordinates. It can also be given using

`DISPLACE`,*displacement_type*

displacement_type can be one of the following:

`SYM` Use symmetrical displacements. This yields the minimum number of displacements and always preserves the symmetry of the wavefunction. This is the default and only recommended option.

`CART` Displacements are generated for all 3*N* Cartesian coordinates. This is normally not recommended, since in cases in which molecular symmetry is present it generates far more displacements than needed. Also, the wavefunction symmetry is not preserved, and the calculation must be done in C1 symmetry.

`UNIQUE` As `CART`, but symmetry-equivalent displacements are eliminated. Not recommended either.

41.2.2 Numerical derivatives of a variable

Numerical derivatives of the value of a variable can be computed using

`VARIABLE,`*name*

The default is to compute the gradient of the current energy.

41.2.3 Step-sizes for numerical gradients

By default, the numerical step sizes are 0.01 bohr for distances or Cartesian coordinates, and 1 degree for angles. These defaults can be changed using

`RSTEP,`*dr*
`ASTEP,`*da*

where *dr* is the displacement for distances (or Cartesian coordinates) in bohr, and *da* is the displacement for angles in degree. The value of `RSTEP` is used for symmetrical displacements. The step sizes for individual variables can be modified using

`VARSTEP,`*varname*=*value*,...

where the *value* must be in atomic units for distances and in degree for angles.

41.2.4 Active and inactive coordinates

By default, numerical gradients are computed with respect to all variables on which the Z-matrix depends, or for all 3*N* coordinates if there are no variables or `XYZ` inputstyle is used. One can define subsets of active variables using

ACTIVE,*variables*

If this card is present, all variables which are not specified are inactive. Alternatively,

INACTIVE,*variables*

In this case all variables that are not given are active.

41.3 Saving the gradient in a variables

If the directive

VARSAV

is given, the gradient is saved in variables GRADX, GRADY, GRADZ. GRADX(n) is the derivative with respect to *x* for the *n*-th atom. The atoms are in the order as printed. This order can be different from the order in the input z-matrix, since the centres are reordered so that all atoms of the same type follow each other.

42 GEOMETRY OPTIMIZATION (OPTG)

Automatic geometry optimization is invoked using the OPTG command. The OPT command available in previous MOLPRO versions is no longer needed and not available any more.

OPTG[, *key1=value, key2=value,......*]

The OPTG command can be used to perform automatic geometry optimizations for all kinds of wavefunctions. For minimum searches, it is usually sufficient to give just the OPTG command without further options or directives, but many options are available which are described in the following sections.

Various optimization methods can be selected as described in section 42.2.1. MOLPRO allows minimization (i.e. search for equilibrium geometries), transition state optimization (i.e. search for saddle points on energy surfaces), and reaction path following. The standard algorithms are based on the *rational function* approach and the *geometry DIIS* approach. Also available is the *quadratic steepest descent following* method of Sun and Ruedenberg (see J. Sun and K. Ruedenberg, *J. Chem. Phys.* **99**, 5257 (1993)). This method is often advantageous in Transition State searches. For a detailed discussion of the various minimization algorithms see F. Eckert, P. Pulay and H.-J. Werner, *J. Comp. Chem* **18**, 1473 (1997). Reaction path following is described in F. Eckert and H.-J. Werner, *Theor. Chem. Acc.* **100**, 21, (1998). Please refer to the references section for citations of the analytic gradient methods.

When analytical gradients are available for the optimized energy these will be used. Otherwise the gradient will be computed numerically from finite energy differences. Normally, the last computed ground-state energy is used. But the VARIABLE directive or option can be used to optimize, e.g., Davidson corrected energies, excited states, or counterpoise corrected energies.

By default the program repeats in each geometry optimization step those commands in the input that are needed to compute the last energy. For example, for MP2 gradients the commands HF and MP2 are needed. The MP2 gradients will then be computed automatically. It is also possible to define procedures for the energy calculation, or to specify the first command from which the input should be repeated in each step (see section 42.1.1). The section of the input which is needed for the geometry optimization must not modify variables that are used in the geometry definition (changes of such variables are ignored, and a warning message is printed).

42.1 Options

Most parameters can be given as options on the `OPTG` command line, as described in this section. Alternatively, directives can be used, which will be described in section 42.2.

42.1.1 Options to select the wavefunction and energy to be optimized

By default, the last computed energy is optimized, and all commands on which the last energy calculation depends are automatically executed. For certain purposes, e.g., optimization of counter-poise corrected energies or Davidson corrected energies, the following options can be used to alter the default behaviour.

`STARTCMD=`*command* Specifies a start command. In each geometry optimization step all input beginning with *command* to the current `OPTG` is processed. This input must not include numerical gradient or Hessian calculations. If numerical gradients are needed, these will be computed for the final energy (or specified variable) by `OPTG`. It is assumed that these commands have been executed before entering the `OPTG` program.

`PROC=`*procname* specifies a procedure to be executed in each geometry optimization step. This must define a complete energy calculation (orbital optimization and correlation treatment), and must not include numerical gradient of Hessian calculations (numerical gradients will be computed automatically for the optimized energy or variable). However, the procedure can include the calculation of analytical gradients, for instance for counter-poise corrected optimizations in which a linear combination of several gradient calculations is needed.

`VARIABLE=`*varname* Optimize the value of variable *varname*. This implies numerical gradients.

42.1.2 Options for optimization methods

`METHOD=RF|AH|DIIS|QSD|QSTPATH|SRMIN|SRTRANS|STSTEEP`
Optimization method to be used. See section 42.2.1 for details.

`ROOT=`1|2 Minimum search (1, default) or transition state search (2).

`DIRECTION=`idir Determines step length and direction in reaction path following, see section 42.2.16.

`STEPMAX=`*value* Max step length in one optimization step. For more detailed specifications see section 42.2.12.

`TRUST=`*value* Trust ratio for Augmented Hessian method (default 0.5).

`AHMAX=`*value* Maximum step size allowed in the Augmented Hessian procedure. This refers to the scaled parameter space (default 0.5).

`CUT=`*value* Threshold for ortho-normalization used in conjugate gradient update of Hessian (default 1.d-3).

`ROTATE` (logical). If .true., the Cartesian coordinates are transformed to minimize rotations (default=.true.)

42.1.3 Options to modify convergence criteria

The standard MOLPRO convergency criterion requires the maximum component of the gradient to be less then $3 \cdot 10^{-4}$ [a.u.] and the maximum energy change to be less than $1 \cdot 10^{-6}$ [H] *or* the maximum component of the gradient to be less then $3 \cdot 10^{-4}$ [a.u.] and the maximum component of the step to be less then $3 \cdot 10^{-4}$ [a.u.].

It is also possible to use the convergency criterion of the Gaussian program package. It is somewhat weaker than the MOLPRO criterion and requires the maximum component of the gradient to be less then $4.5 \cdot 10^{-4}$ [a.u.] and the root mean square (RMS) of the gradient to be less then $3 \cdot 10^{-4}$ [a.u.] as well as the maximum component of the optimization step to be less then 0.0018 [a.u.] and the RMS of the optimization step to be less then 0.0012 [a.u.].

`MAXIT`=*maxit*	maximum number of optimization cycles. The default is 50.
`GRADIENT`=*thrgrad*	required accuracy of the optimized gradient. The default is $3 \cdot 10^{-4}$.
`ENERGY`=*threnerg*	required accuracy of the optimized energy. The default is $1 \cdot 10^{-6}$.
`STEP`=*thrstep*	convergence threshold for the geometry optimization step. The default is $3 \cdot 10^{-4}$.
`BAKER`	(logical). Use Baker's convergency criteria (see J. Baker, *J. Comp. Chem.* **14**,1085 (1993)).
`GAUSSIAN`	(logical). Use Gaussian convergency criteria.
`SRMS`=*thrsrms*	sets (for Gaussian convergency criterion) the required accuracy of the RMS of the optimization step. The default is 0.0012.
`GRMS`=*thrgrms*	sets (for Gaussian convergency criterion) the required accuracy of the RMS of the gradient. The default is $3 \cdot 10^{-4}$.
`FREEZE`=*thrfreez*	Freeze DFT grid and domains in local calculations if the step length is smaller than *thrfreez* (default 0.01).

Note: The defaults for the convergence parameters can also be changed by using a global `GTHRESH` directive, i.e.

`GTHRESH`, `OPTSTEP`=*step*, `OPTGRAD`=*grad*, `ENERGY`=*energy*;

42.1.4 Options to specify the optimization space

If the geometry is given as Z-matrix, the default is to optimize the variables on which the Z-matrix depends. In case of `xyz` input, always all 3N coordinates are optimized, even if the `xyz` input depends on fewer variables. If Cartesian z-matrix input is used, optimization in full space is only enforced if automatic orientation is requested using the `MASS`, or `CHARGE` options on the `ORIENT` directive. See *opt_space* in section 42.2.2 for details.

`SPACE=ZMAT|3N`	Specifies the coordinates to be used in the optimization. Z-matrix optimization is only possible if the geometry is given as Z-matrix.
`OPT3N|3N`	(logical). Same as `SPACE=3N`
`ZMAT`	(logical). Same as `SPACE=ZMAT`

42.1.5 Options to specify the optimization coordinates

These options specify the coordinates in which the optimization takes place. The default is to use local normal coordinates. See *opt_coord* in section 42.2.2 for details.

COORD=NORMAL|NONORMAL|BMAT

NORMAL (logical). Same as COORD=NORMAL.

NONORMAL (logical). Same as COORD=NONORMAL.

BMAT (logical). Same as COORD=BMAT.

42.1.6 Options for numerical gradients

Numerical gradients can be computed with respect to variables on which the Z-matrix depends or with respect to Cartesian coordinates. In the latter case, it is most efficient to use symmetrical displacement coordinates. These do not change the symmetry of the molecule and the number of displacements is minimal. Alternatively (mainly for testing purpose) the gradients can be computed using symmetry unique Cartesian displacements or all 3N Cartesian displacements. In these cases the symmetry of the molecule can be reduced by the displacements and using such displacements is normally not recommended.

DISPLACE=ZMAT|SYMM|UNIQUE|CART
Displacement coordinates to be used for numerical gradient. The default is ZMAT if the geometry is given as a zmatrix which depends on variables, and SYMM (symmetrical displacement coordinates) otherwise. The use of UNIQUE or CART is not recommended.

SYMMETRY=AUTO|NOSYM Symmetry to be used in wavefunction calculations of numerical gradients. This option is only relevant if DISPLACE=UNIQUE|CART. If AUTO is given, the maximum possible symmetry is used for each displacement. This implies that the energy is independent of the symmetry used. Note that this often not the case in MRCI or CASPT2 calculations. The option can also not be used in local correlation calculations.

AUTO (logical). Same as SYMMETRY=AUTO

NOSYM (logical). Same as SYMMETRY=NOSYM

RSTEP=*rstep* Step length for distances in numerical gradient calculations (in bohr). The default is 0.01.

DSTEP=*dstep* Step length for symmetrical displacements (in bohr). The default is 0.01.

ASTEP=*astep* Step length for angles in numerical gradient calculations (in degree). The default is 1.

FOURPOINT (logical). Use 4-point formula for accurate numerical gradient.

NUMERICAL (logical). Force the use of numerical gradients, even if gradients are available.

42.1.7 Options for computing Hessians

By default, an approximate Hessian (model Hessian) is used. Optionally, a Hessian can be computed in the optimization or read from a previous Hessian or frequency calculation.

`NUMHESS`=*hstep*	If given, a numerical Hessian is computed in each *hstep'th* iteration. If *hstep=0* or not given, only an initial Hessian is computed.
`HESSREC`=*record*	Read initial Hessian from the given record. If *record* is not given or zero, the last computed Hessian is used.
`READHESS`	(logical). Same as `HESSREC=0`.
`HESSPROC`=*procname*	specifies a procedure to be used for computing the Hessian. This procedure must be define a complete energy calculation (orbital optimization and correlation treatment). A different method can be used than for the optimized energy. For instance, an MP2 Hessian can be used for CCSD(T) optimizations, or a CASPT2 Hessian for MRCI optimizations. By default, the same procedure is used for the Hessian as for the optimized energy.
`HESSVAR`=*varname*	Compute Hessian for variable *varname*. This implies numerical calculation of the Hessian from energies. The default is to use the same variable as for the energy and gradient.
`HESSCENT`	Use central gradient differences for computing Hessian (only effective if gradients are available)
`HESSFORW`	Use forward gradient differences for computing Hessian (only effective if gradients are available). This effectively computes the Hessian at a slightly displaced geometry, but needs only half the number of displacements. This is the default.
`UPDATE=BFGS|IBFGS|CGRD|PMS|POWELL|MS|NONE`	Hessian update method to be used. See section 42.2.9 for details.
`MAXUPD`=*maxupd*	Max number of Hessian updates. The count is reset to zero each time a Hessian is computed.
`NUMDIAG`	If true, replace diagonal elements of model hessian by diagonal numerical hessian (if available). This sometimes improves convergence, but since it may lead to symmetry breaking it is no the default.

Note that there are restrictions for computing Hessians for multireference methods (MCSCF, MRCI, ACPF,AQCC,RS2). For these methods the symmetry must not change by any displacements, since this could change the occupations and states and may lead to non-contiguous potential energy surfaces. One of the following three options can be used in these cases:

- Use no symmetry from the beginning (`NOSYM`).
- Use symmetric displacement coordinates. This is the default if the optimization is done in 3N cartesian coordinates. One can use `OPTG,DISPLACE=SYMM` to force the use of symmetrical displacements (this creates 3N cartesian coordinates if a Z-matrix is used in the geometry input).
- Use a Z-matrix with the restriction that no variable in the Z-matrix may change the symmetry. For example, `geometry={O;H1,O,r;H2,O,r,H1,theta}` would work, but `geometry={O;H1,O,r1;H2,O,r2,H1,thetai}` would not work. In this case the program prints a warning message. If an incorrect Z-matrix is used and the symmetry changes, the program will crash.

42.1.8 Miscellaneous options:

`VARSAVE`	Save Cartesian gradients in variables `GRADX, GRADY, GRADZ`.
`NONUC`	Do not compute gradients at lattice points.
`DEBUG`	Set debug print options.
`PRINT`=*iprint*	Print option for optimization.
`SAVEXYZ`[=*file*]	Save the optimized coordinates in an xyz-file. One file is written for each step. The filename is *file_nn*`.xyz`, where *nn* is the iteration number. For the final geometry, *_nn* is omitted. If filename is not given, *file* is taken to be the root name of the input, i.e. `test.inp` creates `test_1.xyz` in the first iteration and `test.xyz` for the converged geometry. By default, the xyz information is written to the log file in each step.
`SAVEACT`[=*file*]	Save optimized variables in each step. The file name is *file*`.act`. If *file* is not given the root name of the input is used. The file can be read later using `the READVAR` command or copied into new input.
`SAVEGRD`[=*file*]	Write in each step the Cartesian coordinates and gradients. The file name is *file*`.grd`. If file is not given, the root name of the input appended by `.grd` is used.
`APPEND`	(logical). If given, existing `SAVEACT` and/or `SAVEGRD` files are appended.
`REWIND`	(logical). If given, the `SAVEACT` and/or `SAVEGRD` files are rewound at each step, i.e. only the last geometry or gradient is saved, previous values are overwritten.

42.2 Directives for OPTG

An alternative way to specify options is to use directives, as described in this section. In some cases this allows more detailed specifications than with the options on the `OPTG` command. In particular, directives `ACTIVE` or `INACTICE` can be used to define the optimization space in more detail.

42.2.1 Selecting the optimization method (`METHOD`)

`METHOD`,*key*;

key defines the optimization method.

For *minimization* the following options are valid for *key*:

`RF`	Rational Function method (default).
`AH`	Augmented Hessian method. This is similar to `RF` algorithm but uses a more sophisticated step restriction algorithm.
`DIIS`	Pulay's Geometry DIIS method. As an an additional option you may add the number of geometries to be used in GDIIS interpolation (default 5) and the interpolation type (i.e. the subspace in which the GDIIS interpolation is made. `METHOD,DIIS,` *number*, *type*

type may be GRAD interpolation using the gradients (default), working good for rigid molecules, STEP interpolation using Quasi-Newton steps which could be advantageous in dealing with very floppy molecules, ENER interpolation using energies, which is an intermediate between the above two.

QSD — Quadratic steepest descent method of Sun and Ruedenberg.

SRMIN — Old version of QSD.

For *transition state* searches (invoked with the ROOT option, see section 42.2.11) *key* can be

RF — Rational Function method (default).

DIIS — Pulay's Geometry DIIS method (see above).

QSD — Quadratic Steepest Descent Transition State search using the image Hessian method (see J. Sun and K. Ruedenberg, *J. Chem. Phys.* **101**, 2157 (1994)) The use of this option is recommended for transition state searches – especially in complicated cases. The optimization step is checked and the Hessian is recalculated when approaching a troublesome region of the PES. Thus **this method is somewhat safer (and often faster) in reaching convergence than the RF or DIIS method**. The Hessian recalculation safeguard may be turned off using the METHOD,QSD,NOHESS input card.

SRTRANS — Old version of QSD.

For *reaction path following* the input *key* is

QSDPATH — Quadratic Steepest Descent reaction path following. This methods determines reaction paths (intrinsic reaction coordinates, IRCs) by following the exact steepest descent lines of subsequent quadratic approximations to the potential energy surface. The Hessian matrix is calculated numerically at the first optimization step and subsequently updated by Powell or BFGS update. If a given arc length of the steepest descent lines is exceeded, the Hessian is recalculated numerically (see OPTION section 42.2.16). For details see J. Sun and K. Ruedenberg, *J. Chem. Phys.* **99**, 5269 (1993) It is also possible to recalculate the Hessian after each *m* steps using the NUMHES,*m* command (see section 42.2.7). If the Hessian matrix is recalculated in every optimization step (NUMHES,1) a algorithm different to the one with updated Hessians is used, which is very accurate. Using the PRINT,OPT card, this algorithm prints in every optimization step a *reaction path point r* which is different from the point where the energy and the gradient is calculated but closer to the real reaction path (for further details of the algorithm see J. Sun and K. Ruedenberg, *J. Chem. Phys.* **99**, 5257 (1993)). For further input options of the QSD reaction path following see OPTION section 42.2.16.

SRSTEEP — Old Version of QSDPATH.

42.2.2 Optimization coordinates (COORD)

It is possible to use various coordinate types and algorithms for the optimization. This can be controlled by additional subcommands as described in this and the following subsections.

COORD,[*opt_space*],[*opt_coord*],[NOROT]

These options choose the optimization space and the coordinate system in which the optimization takes place.

opt_space defines the parameters to be optimized. By default, if the geometry input is given in Z-matrix format, all variables on which the Z-matrix depends are optimized. Subsets of the variables on which the Z-matrix depends can be chosen using the ACTIVE or INACTIVE subdirectives. If the Z-matrix depends on no variables or xyz input is used, all 3*N* cartesian coordinates are optimized.

opt_space can be one of the following:

ZMAT	Optimize all variables on which the Z-matrix depends (default if the geometry is given as Z-matrix).
3N	Optimize all 3*N* cartesian coordinates (default if the Z-matrix depends on no variables, or if xyz-input is used). Z-Matrix input coordinates will be destroyed if 3N is used..

opt_coord determines the coordinates in which the optimization takes place. By default, local normal coordinates are used. Optionally cartesian coordinates or natural internal coordinates can be used.

opt_coord can be one of the following:

NORMAL	Optimization in local normal coordinates. This is default if the Model Hessian is used to approximate the Hessian.
NONORM	Don't use local normal coordinates.
BMAT[=*filename*]	Use Pulay's *natural internal coordinates*, see G. Fogarasi, X. Zhou, P. W. Taylor and P. Pulay *J. Am. Chem. Soc.* **114**, 8191 (1992); P. Pulay, G. Fogarasi, F. Pang, J. E. Boggs *J. Am. Chem. Soc.* **101**, 2550 (1979)). Optionally, the created coordinates as well as additional informations about this optimization are written to the specified file. These coordinates resemble in part the valence coordinates used by vibrational spectroscopist, and have the advantage of decreasing coupling between different modes. This often increases the speed of convergence. The use of this option is highly recommended, especially in minimization of large organic molecules with rings. Nevertheless you should keep in mind that these coordinates are constructed automatically, and there exist exotic bond structures which might not be treated properly (e.g. weakly bonded species as in transition state optimizations). In such a case, if the BMAT optimization converges slowly or leads to symmetry-breaking errors, you should try another optimization method and/or cartesian or Z-Matrix coordinates.

If the option [NOROT] is given, the cartesian coordinates are not transformed to minimize rotations.

42.2.3 Displacement coordinates (DISPLACE)

DISPLACE,*displacement_type*

see section 41.2.1 for details.

42.2.4 Defining active geometry parameters (ACTIVE)

ACTIVE,*param*;

Declares variable name *param* to be active in the optimization. By default, initially all variables on which the geometry depends are active; inclusion of an ACTIVE card makes all parameters inactive unless explicitly declared active (see also INACTIVE).

42.2.5 Defining inactive geometry parameters (INACTIVE)

INACTIVE,*param*;

Declares variable name *param* to be inactive in the optimization. If any ACTIVE card appears in the input, this card is ignored! (see also ACTIVE)

42.2.6 Hessian approximations (HESSIAN)

By default, the MOLPRO geometry optimization utilizes a force field approximation to the hessian ("Model Hessian", see R. Lindh, A. Bernhardsson, G. Karlström and P. Malmqvist *Chem. Phys. Lett.* **241**, 423 (1995)), which speeds up convergence significantly. The Model Hessian is parameterized for the elements up to the third row. Alternatively, the model Hessian of Schlegel can be used, or the Hessian can be computed numerically (see also section 42.2.7).

HESSIAN,*options*

where *options* can be

MODEL	Use Lindh's Model Hessian in optimization (default).
MODEL=SCHLEGEL	Use Schlegel's Model Hessian.
MODEL=VDW	Add vdW terms to Lindh's Model Hessian.
SCHLEGEL	same as MODEL=SCHLEGEL.
VDW	same as MODEL=VDW.
NOMODEL	Don't use Model Hessian approximation to the hessian.
NUMERICAL=*hstep*	Recompute Hessian after *hstep* iterations. This disables the use of a model hessian. If *hstep*=0, the Hessian is only computed in the first iteration. Default parameters are used for computing the numerical Hessian, unless modified using options as described for the NUMHESS directive, see Sect. 42.2.7. Any option valid for the NUMHESS directive may also follow the NUMERICAL option on the HESSIAN directive.
READ|RECORD|HESSREC=*record*	Read Hessian from given *record*. If *record* is not given or zero, the last computed hessian will be read. See section 42.2.7 for more details about numerical Hessians.
UPDATE=*type*	Method used for hessian update. See section 42.2.9 for possibilities and details.
MAXUPD=*maxupd*	Max number of hessian updates. The count is reset to zero each time a hessian is computed.

If the Model Hessian is disabled (NOMODEL) and no Hessian is read or computed, the initial hessian is assumed to be diagonal, with values 1 hartree*bohr**(-2) for all lengths, 1 hartree*radian**(-2) for all angles. Additional matrix elements of the hessian can be defined using the HESSELEM directive, see section 42.2.8.

In transition state searches the Hessian is evaluated numerically in the first iteration by default. Alternatively, if READ is specified, a previously computed hessian is used.

42.2.7 Numerical Hessian (NUMHESS)

NUMHESS,*options*

or

NUMHESS,*hstep,options*

If this directive is present a numerical Hessian is computed using finite differences. If analytical gradients are available, one can use forward gradient differences (needs one gradient calculation for each coordinate) or central differences (more accurate, needs two gradient calculations for each coordinate). For transition state optimizations it is usually sufficient to use forward differences. If analytical gradients are not available for the optimized method, the energy is differentiated twice. In this case only central differences are possible.

The following options can be given:

HSTEP=*hstep*	*hstep*=-1: Don't calculate numerical hessian (default for minimization); *hstep*=0 Calculate numerical hessian only once at the start of the optimization (default for transition state searches). *hstep*=*n* Calculate numerical hessian after each *n* optimization steps. This is useful for difficult transition state optimizations (e.g. if the eigenvalue structure of the hessian changes during the optimization).
FORWARD	Use forward differences (default).
CENTRAL	Use the more accurate central differences.
RSTEP=*rstep*	Step length for distances (in bohr). The default is 0.01.
ASTEP=*astep*	Step length for angles (in degree). The default is 0.5 or 1 for angles below and above 90 degree, respectively.
DSTEP=*dstep*	Step length for symmetrical displacements (in bohr). The default is 0.01.
VARIABLE=*varname*	Use given variable for numerical calculation of the Hessian. Note that this disables the use of gradients, and Hessian evaluation can be very expensive.
PROCEDURE=*procname*	Procedure to be used for computing Hessian. This procedure must be define a complete energy calculation (orbital optimization and correlation treatment). A different method can be used than for the optimized energy. For instance, an MP2 hessian can be used for CCSD(T) optimizations, or a CASPT2 hessian for MRCI optimizations. By default, the same procedure is used for the hessian as for the optimized energy.
DISPLACE=*type*	*type* can be one of the following:

SYMM	Use symmetric displacement coordinates (default). This is the only recommended option.
CART	Use 3*N* cartesian displacements (not recommended). This requires many more energy calculations than necessary and does not preserve the molecular symmetry.
UNIQUE	Use symmetry-unique cartesian displacements (not recommended)

Note that the displacement type for gradient and hessian must be the same.

CALC=*icalc* — *icalc*=0: Recalculate the complete Hessian matrix numerically after each *hstep* optimization steps (default).
icalc=1: Recalculate selected Hessian matrix elements if the relative deviation of this element before and after update (see UPDATE, section 42.2.9) is larger than *thresh*. If *thresh* is not specified, a default value of $thresh = 0.05$ (i.e. a maximum deviation of 5%) is used.
icalc=2: Recalculate complete Hessian matrix if the RMS deviation of the Hessian matrix before and after update is larger than *thresh*. If *thresh* is not specified a default value of

THRESH=*thresh* — Threshold for partial or dynamical update of hessian, see above

42.2.8 Hessian elements (HESSELEM)

HESSELEM,*value*, *active1*,*active2*,...

sets the starting value for hessian matrix element between active variables *active1*, *active2* to *value*. If *active2* is omitted it defaults to *active1* (diagonal element). As many HESSELEM directives as needed may be given.

42.2.9 Hessian update (UPDATE)

UPDATE,[TYPE=]*type*,MAX=*maxupd*

This directive chooses the update type and limits the number of points used for the hessian update to *maxupd*. The default number of steps used in hessian update procedures is 5. If there are symmetry constraint in the coordinates of the optimization, the default number may be lower than five.

In minimizations *type* may be

BFGS	Use BFGS update of hessian (default).
IBFGS	Use BFGS update of the inverse hessian.
CGRD	Use Conjugate Gradient update (see also CUT,TRUST).
NONE	Don't do any update.

In transition state optimizations *type* may be

PMS	Combined Powell/Murtagh-Sargent update of hessian (default).
POWELL	Use Powell's update of the hessian.
MS	Use update procedure of Murtagh and Sargent.
NONE	Don't do any update.

42.2.10 Numerical gradients (NUMERICAL)

NUMERICAL,*options*,$active_1$=$step_1$, $active_2$=$step_2$ ... ;

With this directive the gradients are computed by finite differences. $step_i$ is the increment for the active geometry parameter $active_i$. For active parameters which are not specified, the default values are used. By default, the increment is 0.01 bohr for bond distances and 0.5 or 1 degree for angles less than or greater than 90 degrees, respectively. These defaults can be modified by specifying RSTEP or ASTEP. DSTEP is the length of symmetrical displacements, which are used if the optimization is performed in 3N coordinates.

For each active variable, two energy calculations are necessary in each geometry optimization step – so numerical optimizations may be expensive! In optimizations of 3N coordinates symmetrical displacement coordinates are normally used to minimize the number of energy calculations. (see section 41.2.1).

For optimization of special energies see VARIABLE section 42.2.17.

The following options can be given:

RSTEP=*rstep*	Step length for distances (in bohr). The default is 0.01.
ASTEP=*astep*	Step length for angles (in degree). The default is 0.5 or 1 for angles below and above 90 degree, respectively.
DSTEP=*dstep*	Step length for symmetrical displacements (in bohr). The default is 0.01.
CENTRAL	Use central differences for gradient (default)
FORWARD	Use forward differences (not recommended for gradient).
FOURPOINT	Use four-point formula for very accurate numerical gradients.
PROCEDURE=*procname*	Use given procedure for numerical calculation of the gradient. This procedure must define a complete energy calculation (orbital optimization and correlation treatment).
VARIABLE=*varname*	Use given variable for numerical calculation of the gradient.
DISPLACE=*type*	The displacement type. Note that the displacement type for gradient and hessian must be the same. *type* can be one of the following:

SYMM	Use symmetric displacement coordinates (default). This is the only recommended option.
CART	Use 3*N* cartesian displacements (not recommended). This requires many more energy calculations than necessary and does not preserve the molecular symmetry.
UNIQUE	Use symmetry-unique cartesian displacements (not recommended)

42.2.11 Transition state (saddle point) optimization (ROOT)

ROOT,*root*

Specifies the eigenvector of the hessian to be followed.

root=1	specifies a minimization (default).
root=2	specifies a transition state (saddle point) optimization.

In the present implementation a saddle point search is possible with the rational function method (`METHOD,RF`), the geometry DIIS method (`METHOD,DIIS`) and the quadratic steepest descent method of Sun and Ruedenberg (`METHOD,SRTRANS`).

Note that convergence is usually much more difficult to achieve than for minimizations. In particular, a good starting geometry and a good approximation to the hessian is needed. The latter is achieved by evaluating the hessian numerically (see section 42.2.7) or using a precomputed hessian (see section 42.2.6).

42.2.12 Setting a maximum step size (`STEP`)

`STEP`,*steplength*,*drmax*,*damax*,*drmax1*,*damax1*

steplength is the initial step length in the scaled parameter space (default 0.3). In the AH-method this is dynamically adjusted, and can have a maximum value *ahmax* (see `TRUST`).

drmax is the initial max change of distances (in bohr, default 0.3). In the AH-method this is dynamically adjusted up to a maximum value of *drmax1* (default 0.5 bohr).

damax is the initial max change of angles (in degree, default 2). In the AH-method this is dynamically adjusted up to a maximum value of *damax1* (default 10 degrees).

42.2.13 Redefining the trust ratio (`TRUST`)

`TRUST`,*ratio*,*ahmax*

ratio determines the radius around the current minimum in which points are used to update the Hessian with the conjugate gradient method (default 0.5; see also `UPDATE`).

ahmax is the maximum step size allowed in the Augmented Hessian procedure. This refers to the scaled parameter space (default 0.5). The initial step size is *stepmx* (see `STEP` card).

42.2.14 Setting a cut parameter (`CUT`)

`CUT`,*threshold*

Specifies a threshold for ortho-normalization used in conjugate gradient update of hessian (default 1.d-3; see also `UPDATE`).

42.2.15 Line searching (`LINESEARCH`)

`LINESEARCH`,*iflag*,*thrlmin*,*thrlmax*

Interpolate the geometry of the stationary point (minimum or saddle point) by a quartic polynomial between the current and the previous geometry. If *iflag*=0 or no *iflag* is set, the next optimization step will be taken from the interpolated geometry using the interpolated energy and gradient. If *iflag*=1 the energy and gradient will be recalculated at the interpolated geometry before taking the new optimization step. Note though, that the additional effort of recalculating the energy and gradient is usually not met by the increase of the convergence rate of the optimization. *thrlmin* and *thrlmax* are min and max thresholds for the recalculation of the energy and the gradient in case *iflag*=1. I.e. the recalculation just takes place if the interpolated geometry isn't too close to the actual geometry *thrlmin* and isn't too remote from the actual geometry *thrlmax*. Default values are *thrlmin*=0.001 and *thrlmax*=0.05 in the scaled parameter space of the optimization.

42.2.16 Reaction path following options (OPTION)

OPTION,*key=param*;

where *key* can be

IDIR	If starting at a transition state (or near a transition state) determine where to take the first step. If IDIR=0 is chosen, the first step will be towards the transition state. This is the default. If IDIR=1 is given in the input the first optimization step will be along the "transition vector" i.e. the hessian eigenvector to the smallest eigenvalue which points down towards the minimum. If using a larger IDIR parameter, the first step will be larger; if using a negative value, the first step will be in the opposite direction.
STPTOL	If using an updated hessian matrix, this parameter determines what update to take. If the step size between two subsequent points on which the steepest decent lines are puzzled together is smaller than *stptol* (i.e. if we are close to a minimum) the BFGS update is used, otherwise it is Powell update. The default value of *stptol* is $1.d-6$.
SLMAX	This option is only valid with the old version of the reaction path following algorithm (i.e. METHOD,SRSTEEP). In this algorithm slmax determines the frequency of the recalculation of the numerical hessian. If the total step size of the last steps exceeds *slmax* the hessian will be recalculated, otherwise it will be updated. By default *slmax* is two times the maximum step size of the optimization step *steplength* (see STEP section 42.2.12). If you are using METHOD,QSD, the SLMAX option is obsolete and the NUMHES command (see above) should be used instead.

42.2.17 Optimizing energy variables (VARIABLE)

VARIABLE,*name*;

Defines a variable *name* which holds the energy value to be optimized in using finite differences. By default, this is ENERGY(1) as set by the most recent program. Other variables which can be used are

ENERGY(*i*)	holds last energy for state *i*.
ENERGR(*i*)	holds last reference energy for state *i*.
ENERGD(*i*)	holds last Davidson corrected energy for state *i*.
ENERGP(*i*)	holds last Pople corrected energy for state *i*.
ENERGC	holds CCSD (QCI, BCCD) energy in CCSD(T) [QCI(T), BCCD(T)] calculations (single state optimization).
ENERGT(1)	holds CCSD(T) energy in CCSD(T) calculations (single state)
ENERGT(2)	holds CCSD[T] energy in CCSD(T) calculations (single state).
ENERGT(3)	holds CCSD-T energy in CCSD(T) calculations (single state).

These variables are set automatically by the CI and/or CCSD programs. It is the user's responsibility to use the correct variable name; an error exit occurs if the specified variable has not been defined by the last program or the user.

Note: The use of the `VARIABLE` option triggers `NUMERICAL`, so optimization can be very inefficient!

42.2.18 Printing options (`PRINT`)

`PRINT`,*code*=*level*,... ;

Enables printing options. Usually level should be omitted or 0; values of *level* > 0 produce output useful only for debugging. *code* can be

`HESSIAN`	prints the updated hessian matrix. Note that its diagonal elements are printed anyway.
`HISTORY`	prints the complete set of previous geometries, gradients and energies.
`GRADIENT`	prints extended gradient information
`OPT`	prints detailed information about the optimization process (mainly for debugging).

Several print options can be specified with one `PRINT` command.

42.2.19 Conical Intersection optimization (`CONICAL`)

To optimize a conical intersection between two electronic states having the same spin, three vectors must be evaluated at SA-CPMCSCF level:

1) Non-Adiabatic Derivative Coupling (DC).

2) Gradient of the lower state (LSG).

3) Gradient of the upper state (USG).

This requires three different `CPMCSCF` directives in the MULTI input:

`CPMCSCF, NACM`, S_i, S_j, `ACCU`=*1.0d-7*, `record`=*record1.file*
`CPMCSCF, GRAD`, S_i, `SPIN`=*Spin of state* S_i, `ACCU`=*1.0d-7*, `record`=*record2.file*
`CPMCSCF, GRAD`, S_j, `SPIN`=*Spin of state* S_j, `ACCU`=*1.0d-7*, `record`=*record3.file*

where S_i,S_j are the electronic states in the usual format *istate.istsym*, and *record[n].file* specifies the name and the file number where CPMCSCF solutions should be stored. Parameter `SPIN` is half of the value in the `WF` card used to define the electronic state.

Things to remember:

i) Specify always three different *record.file* on the `CPMCSCF` directives.

ii) Evaluate the CPMCSCF for USG always last.

iii) Skip the DC evaluation if the conical intersection involves states with different spin (e.g., a Singlet/Triplet crossing) because the coupling is then zero.

Three sets of FORCE commands (only two for Singlet/Triplet intersection) follow the MULTI input. They will be like:

FORCE
SAMC,*record[n].file*
CONICAL,*record4.file*[,NODC]

where *record.file* is one of the records containing CPMCSCF info and *record4.file* points to a free record used for internal storage by the CONICAL code. *record4.file* must be the same on all the CONICAL directives. Furthermore, the present implementation works properly only if *file=1* on the CONICAL directive. The optional keyword NODC must be used in case of different spins (e.g., S/T crossing) when DC is not needed.

The actual optimization is performed using OPTG,STARTCMD=MULTI The example below optimizes the conical intersection in LiH_2 (ground and excited states are both doublets).

```
***, LiH2

basis=sto-3g
print,orbitals,civector

symmetry,x              !use only molecular plane. Both states must be in the same symmetry.
geometry={
          Li;
          h1,Li,r;
          h2,Li,r,h1,theta}

r=3.0
theta=35

{hf;wf,4,1,0}

{multi;occ,6,1;wf,5,1,1;state,2                        !state averaged casscf
CPMCSCF,NACM,1.1,2.1,accu=1.0d-7,record=5100.1         !cpmcscf for non-adiabatic couplings
CPMCSCF,GRAD,1.1,spin=0.5,accu=1.0d-7,record=5101.1    !gradient for state 1
CPMCSCF,GRAD,2.1,spin=0.5,accu=1.0d-7,record=5102.1}   !gradient for state 2

{Force
SAMC,5100.1             !compute coupling matrix element
CONICAL,6100.1}         !save information for optimization of conical intersection

{Force
SAMC,5101.1             !compute gradient for state 1
CONICAL,6100.1}         !save information for optimization of conical intersection

{Force
SAMC,5102.1             !compute gradient for state 2
CONICAL,6100.1}         !save information for optimization of conical intersection

optg,startcmd=multi     !find conical intersection
```

http://www.molpro.net/info/current/examples/lih2_D0D1.com

This second example optimizes the singlet-triplet intersection in $LiH_2(+)$ (ground state is Singlet, excited state is Triplet).

```
***, LiH2

basis=sto-3g

symmetry,nosym
geometry={
          Li;
          H1,Li,r;
          H2,Li,r,H1,theta}

r=3.7
theta=160

{hf;wf,4,1,0}

{multi;
 occ,7;
 wf,4,1,0;   !singlet state
 wf,4,1,2;   !triplet state
 CPMCSCF,GRAD,1.1,spin=0,accu=1.0d-7,record=5101.1   !cpmcscf for gradient of singlet state
 CPMCSCF,GRAD,1.1,spin=1,accu=1.0d-7,record=5100.1   !cpmcscf for gradient of triplet state
}

{Force
 SAMC,5101.1                !state averaged gradient for singlet state
 CONICAL,6100.1,NODC}       !save information for OPTCONICAL

{Force
 SAMC,5100.1                !state averaged gradient for triplet state
 CONICAL,6100.1,NODC}       !save information for OPTCONICAL

optg,startcmd=multi,gradient=1.d-6  !find singlet-triplet crossing point
```

http://www.molpro.net/info/current/examples/lih2+_S0T0.com

42.3 Using the SLAPAF program for geometry optimization

It is optionally possible to use the SLAPAF program written by Roland Lindh for geometry optimizations. This is done by prepending the optimization method with 'SL'. The following methods are supported:

SLRF	Use the rational function approximation;
SLNR	Use the Newton-Raphson method;
SLC1	Use the C1-DIIS method;
SLC2	Use the C2-DIIS method.

When using DIIS methods (SLC1 or SLC2), the DIIS parameters are specified in the same way as in standard molpro optimizer.

There are some differences when using the SLAPAF program:

1) It is not possible to use Z-matrix coordinates in the optimization.

2) Instead, one can explicitly define internal coordinates to be varied or fixed.

3) Additional constraints can be imposed on the converged geometry in a flexible way.

42.3.1 Defining constraints

Constraints and internal coordinates (see below) can be linear combinations of bonds, angles etc. The latter, called here primitive internal coordinates, can be specified before the constraints definition, or directly inside. The general definition of a primitive coordinate is:

`PRIMITIVE`,[`NAME=`] *symbolic name*, *explicit definition;*

or

`PRIM`,[`NAME=`] *symbolic name*, *explicit definition;*

Here *symbolic name* is the name given to the primitive coordinate (if omitted, it will be generated automatically). This name is needed for further reference of this primitive coordinate.

explicit definition has the form:

type,atoms

type can be one of the following:

`BOND`	Bond length, defined by 2 atoms.
`ANGLE`	Bond angle, defined by 3 atoms (angle 1–2–3).
`DIHEDRAL`	Dihedral angle, defined by 4 atoms (angle between the planes formed by atoms 1,2,3 and 2,3,4, respectively).
`OUTOFPLANE`	Out-of-plane angle, defined by 4 atoms (angle between the plane formed by atoms 2,3,4 and the bond 1–4).
`DISSOC`	A dissociation coordinate, defined by two groups of atoms.
`CARTESIAN`	Cartesian coordinates of an atom.

For all types except `DISSOC` and `CARTESIAN`, atoms are given as:

`ATOMS=`[*a1,a2,a3,...*]

where the number of atoms required varies with *type* as specified above, and the atomic names *a1,a2,a3,...* can be either atomic tag names from the Z-matrix input, or integers corresponding to Z-matrix rows. Note that the square brackets are required here and do not indicate optional input.

For `DISSOC` the specification is as follows:

`DISSOC,GROUP1=`[*a1,a2,...*],`GROUP2=`[*b1,b2,...*];

The corresponding internal coordinate is the distance between the centres of mass of the two groups.

For `CARTESIAN` the definition is

`CARTESIAN`, *I, atom;*

where *I* can be one of `X,Y,Z` or 1,2,3 and *atom* can be a z-matrix atom name or an integer referring to the z-matrix row.

With this definition, the constraints are defined as

`CONSTRAINT`,[`VALUE=`]*value*,[*unit*],[[`FACTOR=`]*fac*,*prim*,[[`FACTOR=`]*fac*],*prim,...;*

where *value* is the value imposed to the constraint, and *prim* is either the name of the primitive defined before this constraint, or an explicit definition; and *fac* is a factor of the corresponding primitive in the constraint. If *fac* is omitted it is taken to be 1.

If *value* is specified in `Angstrom` or `Radian`, *unit* must be given.

Examples for H_2O in C_s symmetry:

Constraining the bond angle to 100 degrees:

```
constraint,100,deg,angle,atoms=[h1,o,h2];
```

which is equivalent to

```
primitive,a1,angle,atoms=[h1,o,h2];
constraint,100,a1;
```

Keeping the two OH distances equal:

```
constraint,0,bond,atoms=[h1,o],-1.,bond,atoms=[h2,o];
```

which is equivalent to

```
primitive,b1,bond,atoms=[h1,o];
primitive,b2,bond,atoms=[h2,o];
constraint,0,b1,-1.,b2;
```

42.3.2 Defining internal coordinates

By default `SLAPAF` optimizes in force-constant weighted normal coordinates that are determined automatically. However, the user can define his own coordinates. The definition of internal coordinates, similar to constraints, is based on primitive coordinates. The input is:

`INTERNAL`,[[`NAME=`]*name*],[[`FACTOR=`]*fac*],*prim*,[[`FACTOR=`]*fac*],*prim,...;*

`FIX`, [[`NAME=`]*name*],[[`FACTOR=`]*fac*],*prim*,[[`FACTOR=`]*fac*],*prim,...;*

Internal coordinates that are specified using `INTERNAL` are varied and those using `FIX` are fixed to their initial values.

An important point for the definition of internal coordinates is that their total number must be equal to the number of degrees of freedom of the molecule . Otherwise an error message is generated. Only symmetry independent coordinates need to be given.

42.3.3 Additional options for SLAPAF

Some options can be passed to the SLAPAF program. Options are specified with SLOPT subdirective:

{`opt;method=slnr;`{`slopt;opt1;opt2,`*par1,par2*`;opt3;...`}}

The available options are

`CART`	Use eigenvectors of the approximate Hessian, expressed in cartesian coordinates, as the definition of internal coordinates;
`NOMA`	Don't impose any restrictions on the step size;
`UORD`	Order the gradients and displacement vectors according to Schlegel prior to the update of the Hessian. Default is no reordering;
`HWRS`	Use force field weighted internal coordinates (default);

`RS-P`	Activate RS-P-RFO as default for transition state search; default is RS-I-RFO;
`NOHW`	Use unweighted internal coordinates;
`PRBM`	Print B-matrix;
`RTHR,`*Thra,Thrb,Thrt*	Thresholds for redundant coordinate selection for bonds, bends and torsions, respectively. Default 0.2, 0.2, 0.2
`MODE,`*index*	Hessian vector index for mode following when calculating transition states.
`FIND`	Enable unconstrained optimization for constrained cases, when looking for transition states (see MOLCAS manual).
`GNRM,`*thr*	Threshold for FIND, default 0.2 (see MOLCAS manual).

For more information, please consult the MOLCAS manual.

42.4 Examples

42.4.1 Simple HF optimization using Z-matrix

```
***, Allene geometry optimization using Z-Matrix
memory,1,m
basis=sto-3g

rcc=1.32 ang
rch=1.08 ang
acc=120  degree
Geometry={C1                          !Z-matrix input
          C2,c1,rcc
          Q1,c1,rcc,c2,45
          C3,c2,rcc,c1,180,q1,0
          h1,c1,rch,c2,acc,q1,0
          h2,c1,rch,c2,acc,h1,180
          h3,c3,rch,c2,acc,h1,90
          h4,c3,rch,c2,acc,h2,90}
hf
optg,saveact='allene.act',savexyz='allene.xyz'  !default optimization using model hessian.
                                                !Save optimized variables in file allene.act
                                                !Save optimized geometry in xyz style in in fi
```

http://www.molpro.net/info/current/examples/allene_optscf.com

42.4.2 Optimization using natural internal coordinates (BMAT)

```
***, Allene geometry optimization using natural internal coordinates
memory,1,m
basis=sto-3g

rcc=1.32 ang
rch=1.08 ang
acc=120  degree
symmetry,nosym
Geometry={C1;                       !Z-matrix input
          C2,c1,rcc
          Q1,c1,rcc,c2,45
          C3,c2,rcc,c1,180,q1,0
          h1,c1,rch,c2,acc,q1,0
          h2,c1,rch,c2,acc,h1,180
          h3,c3,rch,c2,acc,h1,90
          h4,c3,rch,c2,acc,h2,90}
hf;
optg                 !default optimization using model hessian
coord,bmat           !use natural internal coordinates

optg,coord=bmat      !same as above
```

http://www.molpro.net/info/current/examples/allene_opt_bmat.com

42.4.3 MP2 optimization using a procedure

```
***, Allene geometry optimization using Z-Matrix
memory,2,m
basis=vdz

rcc=1.32 ang
rch=1.08 ang
acc=120  degree
Geometry={C1                       !Z-matrix input
          C2,c1,rcc
          Q1,c1,rcc,c2,45
          C3,c2,rcc,c1,180,q1,0
          h1,c1,rch,c2,acc,q1,0
          h2,c1,rch,c2,acc,h1,180
          h3,c3,rch,c2,acc,h1,90
          h4,c3,rch,c2,acc,h2,90}

optg,procedure=runmp2     !use procedure optmp2

runmp2={hf;mp2}         !procedure definition
```

http://www.molpro.net/info/current/examples/allene_optmp2.com

42.4.4 Optimization using geometry DIIS

```
***, CAFFEINE cartesian coordinates (XYZ format)
memory,1,m
basis=sto-3g
geomtyp=xyz
geometry={
24
           CAFFEINE CARTESIAN COORDINATES
C          0.8423320060      -0.3654865620       0.0000000000
C         -0.2841017540      -1.1961236000       0.0000000000
N          2.0294818880      -1.1042264700       0.0000000000
N          0.0774743850      -2.5357317920       0.0000000000
N         -1.6472646000      -0.6177952290       0.0000000000
C          1.4531962870      -2.3678913120       0.0000000000
C          0.6373131870       1.1735112670       0.0000000000
C         -1.7812691930       0.7688916330       0.0000000000
N         -0.6771444680       1.6306355000       0.0000000000
O          1.6106752160       1.9349693060       0.0000000000
O         -2.9202890400       1.2510058880       0.0000000000
C         -0.9202462430       3.1094501020       0.0000000000
C         -2.8623938560      -1.4824503660       0.0000000000
C          3.4552156930      -0.6811094280       0.0000000000
H          2.0878150460      -3.2451913360       0.0000000000
H         -1.4989252090       3.4222116470      -0.8897886280
H         -1.4989252090       3.4222116470       0.8897886280
H          0.0071905670       3.7148499490       0.0000000000
H         -3.4903070930      -1.2888938190      -0.8907763360
H         -3.4903070930      -1.2888938190       0.8907763360
H         -2.6289534570      -2.5638654230       0.0000000000
H          4.1360211370      -1.5529079440       0.0000000000
H          3.6817059520      -0.0685850980       0.8931597470
H          3.6817059520      -0.0685850980      -0.8931597470
}

hf
optg,savexyz=caffeine.xyz   !save optimized geometry in file caffeine.xyz
coord,bmat                  !Optimization in natural internal coordinates
method,diis                 !Optimization method: Geometry DIIS

optg,coord=bmat,method=diis,savexyz=caffeine.xyz   !same as above
```

http://www.molpro.net/info/current/examples/caffeine_opt_diis.com

42.4.5 Transition state of the HCN – HNC isomerization

The first example shows how to do a MP2 transition state optimization. The initial Hessian is taken from a previous HF frequency calculation.

```
***, HCN <--> NHC Isomerization - Transition State Optimization and Frequencies

l1=1.18268242 ang
l2=1.40745082 ang
a1=55.05153416 degree

basis=3-21G

symmetry,nosym
geometry={
          C
          N,1,l1
          H,2,l2,1,a1}

hf                         ! HF-SCF

frequencies,analytical     ! Vibrational frequencies for HF-SCF (analytical Hessian)

mp2                        ! MP2

optg,root=2,method=rf,readhess     ! Transition State Search using Rational Function Optimizer

frequencies                ! Vibrational frequencies for MP2 (numerical Hessian)
---
```

http://www.molpro.net/info/current/examples/hcn_mp2_ts.com

The second example shows how to do a CCSD(T) optimization with an MP2 hessian. Note that currently the CCSD(T) gradient is computed numerically using finite energy differences, and this can take long time for larger molecules. The calculation of the MP2 hessian finite differences of analytical gradients.

```
***, HCN <--> NHC Transition State Optimization and Frequencies

rcn=1.18 ang
rnh=1.40 ang
alpha=55 degree

basis=vtz

geometry={
          C
          N,1,rcn
          H,2,rnh,1,alpha}

hf
ccsd(t)
optg,root=2,hessproc=runmp2  !Transition state optimization for ccsd(t) using mp2 hessian
frequencies                  !CCSD(T) frequencies (using numerical second derivatives)

runmp2={hf;mp2}    !procedure definition
---
```

http://www.molpro.net/info/current/examples/hcn_ccsd_ts.com

The last example shows how to do a MRCI+Q (MRCI with Davidson correction) optimization with an CASPT2 hessian. As for CCSD(T), the MRCI+Q gradient as computed numerically, while the CASPT2 hessian is obtained using finite differences of analytical CASPT2 gradients.

```
***, HCN <-> NHC Isomerization - Transition State Optimization and Frequencies
print,orbitals,civector
rcn=1.18 ang
rnh=1.40 ang
alpha=55 degree

basis=vtz

geometry={
          C
          N,1,rcn
          H,2,rnh,1,alpha}

closed,4       ! global setting for casscf inactive space

hf             ! HF-SCF
multi          ! CASSCF
mrci           ! MRCI
optg,root=2,variable=energd,hessproc=runrs2   !optimize mrci+q transition state and caspt2 for

runrs2={multi;rs2}       !procedure definition for caspt2
---
```

http://www.molpro.net/info/current/examples/hcn_mrci_ts.com

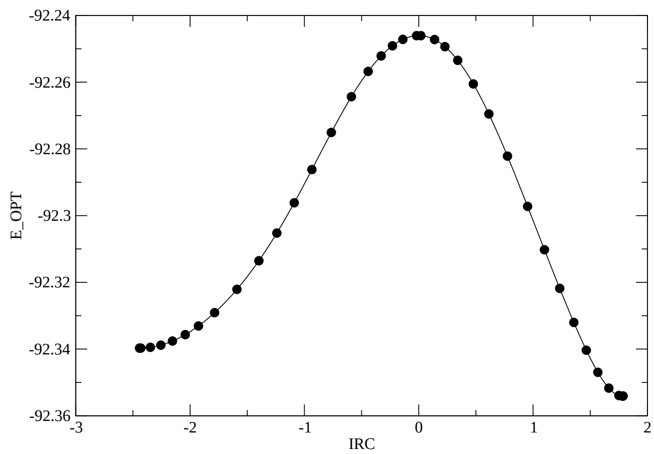
42.4.6 Reaction path of the HCN – HNC isomerization

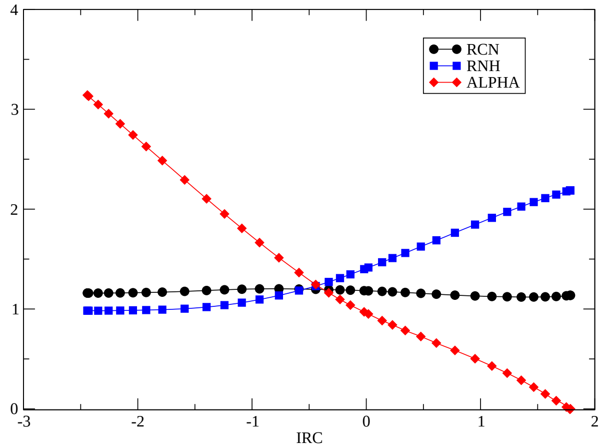
The following input first optimizes the transition state, and then performs reaction path calculations in both directions. The results are plotted.

```
***, HCN <---> NHC Isomerization Reaction Path
memory,1,m
basis=3-21G

rcn=1.18282 ang       ! Starting geometry is transition state
rnh=1.40745 ang
alpha=55.05 degree

symmetry,x            ! Cs Symmetry
geometry={
          C
          N,1,rcn
          H,2,rnh,1,alpha}

int
rhf
optg,root=2,saveact=hcn_ts,rewind            ! Find and store the TS
{optg,method=qsdpath,dir=1, numhess=5,hesscentral,saveact=hcn_path}         ! find IRC in pos

readvar,hcn_ts.act                           ! Reset geometry to TS
{optg,method=qsdpath,dir=-1,numhess=5,hesscentral,saveact=hcn_path,append} !find IRC in negat

readvar,hcn_path.act

alpha=alpha*pi/180    !convert angle to radian

table,irc,rcn,rnh,alpha,e_opt   !tabulate results

{table,irc,e_opt                !plot energy profile as function of irc
 plot,file='hcn_eopt.plot'}

{table,irc,rcn,rnh,alpha        !plot distances and angle as function of irc
 plot,file='hcn_dist.plot'}
```

http://www.molpro.net/info/current/examples/hcn_isomerization.com

This produces the plots





42.4.7 Optimizing counterpoise corrected energies

Geometry optimization of counterpoise corrected energies is possible by performing for the total system as well as for each individual fragment separate `FORCE` calculations. The gradients and energies are added using the `ADD` directive. This requires that `NOORIENT` has been specified in the geometry input, in order to avoid errors due to unintended rotation of the system. This default can be disabled using the `NOCHECK` option, see `ADD` above.

The way a counterpoise corrected geometry optimization works is shown in the following example. Note that the total counterpoise corrected energy must be optimized, not just the interaction energy, since the interaction energy depends on the monomer geometries and has a different minimum than the total energy. The interaction energy could be optimized, however, if the monomer geometries were frozen. In any case, the last calculation before calling `OPTG` must be the calculation of the total system at the current geometry (in the example below the dimer calculation), since otherwise the optimizer gets confused.

```
***,HF dimer MP2/CP optimization with relaxed monomers

basis=avtz
gthresh,energy=1.d-8

! INITIAL VALUES OF GEOMETRY VARIABLES

RFF=      5.3
R1=       1.76
R2 =      1.75
THETA1 =  7.0
THETA2 = 111

symmetry,x
orient,noorient
geometry={
      f1
      f2  f1  rff
      h1  f1  r1   f2  theta1
      h2  f2  r2   f1  theta2  h1  180.}

label:

text, CALCULATION AT LARGE SEPARATION

rff_save=rff            !save current rff distance
rff=1000                !dimer calculation at large separation

text, HF1

dummy,f2,h2;            !second hf is now dummy
{hf;accu,16}            !scf for first monomer
mp2;                    !mp2 for first monomer
ehf1inf=energy          !save mp2 energy in variable
forces;                 !compute mp2 gradient for first monomer

text, HF2

dummy,f1,h1;            !first hf is now dummy
{hf;accu,16}            !scf for second monomer
mp2;                    !mp2 for second monomer
ehf2inf=energy          !save mp2 energy in variable
forces;                 !compute mp2 gradient for second monomer
add,1                   !add to previous gradient
einf=ehf1inf+ehf2inf    !total energy of unrelaxed momomers

rff=rff_save            !reset HF - HF distance to current value

text, CP calculation for HF1 MONOMER

dummy,f2,h2;            !second hf is now dummy
{hf;accu,16}            !scf for first monomer
mp2;                    !mp2 for first monomer
ehf1=energy             !save mp2 energy in variable
forces;                 !compute mp2 gradient for first monomer
add,-1                  !subtract from previous gradient

text, CP calculation for HF2 MONOMER

dummy,f1,h1;            !first hf is now dummy
{hf;accu,16}            !scf for second monomer
mp2;                    !mp2 for second monomer
ehf2=energy             !save mp2 energy in variable
forces;                 !compute mp2 gradient for first monomer
add,-1                  !subtract from previous gradient

text, DIMER CALCULATION
dummy                   !reset dummies
{hf;accu,16}            !scf for dimer
mp2;                    !mp2 for dimer
edimer=energy           !save mp2 energy in variable
```

The next example shows how the same calculations can be done using numerical gradients. In this case, first the total counter-poise corrected energy is formed and then optimized. Note that the `ADD` command does not work for numerical gradients.

```
***,HF dimer MP2/CP optimization with relaxed monomers

basis=avtz
gthresh,energy=1.d-8

! INITIAL VALUES OF GEOMETRY VARIABLES

RFF=      5.3
R1=       1.76
R2 =      1.75
THETA1 =  7.0
THETA2 = 111

symmetry,x
orient,noorient
geometry={
      f1
      f2  f1  rff
      h1  f1  r1   f2  theta1
      h2  f2  r2   f1  theta2  h1  180.}

label:

text, CALCULATION AT LARGE SEPARATION

rff_save=rff            !save current rff distance
rff=1000                !dimer calculation at large separation

text, HF1

dummy,f2,h2;            !second hf is now dummy
{hf;accu,16}            !scf for first monomer
mp2;                    !mp2 for first monomer
ehf1inf=energy          !save mp2 energy in variable

text, HF2

dummy,f1,h1;            !first hf is now dummy
{hf;accu,16}            !scf for second monomer
mp2;                    !mp2 for second monomer
ehf2inf=energy          !save mp2 energy in variable
einf=ehf1inf+ehf2inf    !total energy of unrelaxed momomers

rff=rff_save            !reset HF - HF distance to current value

text, CP calculation for HF1 MONOMER

dummy,f2,h2;            !second hf is now dummy
{hf;accu,16}            !scf for first monomer
mp2;                    !mp2 for first monomer
ehf1=energy             !save mp2 energy in variable

text, CP calculation for HF2 MONOMER

dummy,f1,h1;            !first hf is now dummy
{hf;accu,16}            !scf for second monomer
mp2;                    !mp2 for second monomer
ehf2=energy             !save mp2 energy in variable

text, DIMER CALCULATION
dummy                   !reset dummies
{hf;accu,16}            !scf for dimer
mp2;                    !mp2 for dimer
edimer=energy           !save mp2 energy in variable
etot=edimer-ehf2-ehf1+ehf1inf+ehf2inf  !total BSSE corrected energy

optg,numerical,variable=etot,gradient=1.d-4,startcmd=label:   !optimize geometry


text, compute optimized monomer energy
```

In the last example the monomer structures are kept fixed, and the interaction energy is optimized.

```
***,HF dimer MP2/CP optimization without monomer relaxation

basis=avtz
gthresh,energy=1.d-8

! INITIAL VALUES OF GEOMETRY VARIABLES

RFF=      5.3
THETA1 =  7
THETA2 = 111

symmetry,x
orient,noorient
geometry={
      f1
      f2  f1  rff
      h1  f1  1.74764059   f2  theta1
      h2  f2  1.74764059   f1  theta2  h1  180.}   !using fixed HF distances of isolated HF

label:

text, CP calculation for HF1 MONOMER

dummy,f2,h2;             !second hf is now dummy
{hf;accu,16}             !scf for first monomer
mp2;                     !mp2 for first monomer
ehf1=energy              !save mp2 energy in variable
forces;                  !compute mp2 gradient for first monomer
scale,-1                 !multiply gradient by -1

text, CP calculation for HF2 MONOMER

dummy,f1,h1;             !first hf is now dummy
{hf;accu,16}             !scf for second monomer
mp2;                     !mp2 for second monomer
ehf2=energy              !save mp2 energy in variable
forces;                  !compute mp2 gradient for first monomer
add,-1                   !subtract from previous gradient

text, DIMER CALCULATION
dummy                    !reset dummies
{hf;accu,16}             !scf for dimer
mp2;                     !mp2 for dimer
edimer=energy            !save mp2 energy in variable
forces;                  !compute mp2 gradient for dimer
add,1                    !add to previous gradient

optg,gradient=.d-5,startcmd=label:   !find next energy

text,optimized geometry parameters
show,rhf,rff,theta1,theta2

text,computed interaction energies
de=(edimer-ehf1-ehf2)*tocm     !CPC corrected interaction energy with fixed monomers
```

http://www.molpro.net/info/current/examples/hfdimer_cpcopt2.com

43 VIBRATIONAL FREQUENCIES (FREQUENCIES)

FREQUENCIES,*options*,

Calculate harmonic vibrational frequencies and normal modes. The hessian is calculated analytically or numerically by finite differences in 3N cartesian coordinates (Z-Matrix coordinates will be destroyed on entry). If analytic gradients are available these are differentiated once to build the hessian, otherwise the energy is differentiated twice. If for the wavefunction method dipole moments are available, the dipole derivatives and the IR intensities are also calculated. Note that numerical hessians cannot be computed when dummy atoms holding basis functions are present. To get reasonable results it is necessary to do a geometry optimization before using the frequency calculation.

43.1 Options

The following *options* are available:

ANALYTICAL Use analytical second derivatives of the energy. At present, analytical second derivatives are only possible for closed shell Hartree-Fock (HF) and MCSCF wavefunctions without symmetry. It is not yet possible to calculate IR-intensities analytically. Note that, due to technical reasons, the analytical MCSCF second derivatives have to be computed in the MCSCF-program using e.g. `multi; cpmcscf,hess` (see MULTI) before they can be used in FREQUENCIES. If analytical MCSCF second derivatives have been computed using `multi; cpmcscf,hess`, FREQUENCIES will use them by default.

CENTRAL Use central differences/high quality force constants (default).

NUMERICAL Differentiate the energy twice, using central differences.

FORWARD Use forward differences/low quality force constants.

SYMM=AUTO|NO During the numerical calculation of the hessian, the symmetry of the molecule may be lowered. Giving SYMM=AUTO the program uses the maximum possible symmetry of the molecular wavefunction in each energy/gradient calculation, and this option therefore minimizes the computational effort. With SYMM=NO no symmetry is used during the frequency calculation (default). For single reference calculations like HF, MP2, CCSD, RCCSD the AUTO option can be safely used and is recommended. However, the AUTO option cannot be used for multireference methods (MCSCF/MRCI/ACPF/AQCC/RS2). If given, the option is disabled in these cases. For these methods frequency calculations are only possible without symmetry. Symmetry is turned off atomatically if the state symmetry is 1. Note that this may fail if there are lower states in other symmetries. Use of RESTRICT, SELECT, REF, PROJECT, LOCAL, state-averaged MCSCF will lead on errors unless the calculation is performed in C_1 symmetry In such cases the whole calculation must be done without symmetry.

AUTO Same as SYMM=AUTO, see above.

NOAUTO|NOSYM Same as SYMM=NO, see above.

HESSREC|SAVE=*record* Save hessian to *record*. By default the hessian is saved on record 5300.2.

FREQREC=*record*	Save frequencies and normal modes to *record* (default 5400.2) This information is used, e.g., by the VSCF/VCI program.
TASKREC=*record*	Save task information in numerical hessian calculation to the given record. This information is required for a restart of a numerical hessian calculation. By default, the information is saved on record 5500.2.
READ	Read hessian from default hessian record.
READ|START=*record*	Use hessian from previously saved on *record*. If the hessian has been computed for the current method and geometry already, it is used by default.
RESTART	Attempt to restart a previous numerical frequency/hessian calculation using default task record.
RESTART=*record*	Attempt to restart a previous numerical frequency/hessian calculation using task record *record*.
LOW=*value*	Threshold for printing low frequencies in cm^{-1}. If this option is given, frequencies below the given value are not printed. By default, all frequencies are printed.
STEP=*value*	Determines the step size of the numerical differentiation of the energy or the gradient. The default step size is 0.01 a.u.
MAXTASK=*value*	Stop the calculation if the number of tasks is exceeded (the calculation can be restarted later).
MAXCPU=*value*	Stop the calculation if the given CPU-time (in sec) exceeded (the calculation can be restarted later).
NEW	Recompute hessian, even if a hessian is already available.
PROJECT	Project rotations and translations out of the hessian (default).
NOPROJECT	Don't project rotations and translations out of the hessian.
PRINT=*value*	Print option. If *value* is greater or equal to zero, the hessian and other information is printed (default -1).
DEBUG	Print Debug information, same as PRINT=1.
SCALE=*value*	Scaling factor for frequencies. The scaled frequencies are used to compute the ZPE and thermodynamic properties.

For compatibility with older MOLPRO versions many of the options can also be set using directives, as described in the following sections.

43.2 Printing options (PRINT)

PRINT,*options*

This directive can be used to control the output:

The following *options* can be given:

HESSIAN	Print the force constant matrix (hessian) i.e. the second derivative matrix of the energy and the mass weighted hessian matrix.
LOW	Print low vibrational frequencies (i.e. the 5 or 6 frequencies belonging to rotations and translations) and their normal modes (default; LOW=-1 suppresses the print).

LOW=*value*	Threshold for printing low vibrations in cm^{-1} (default 150). If a value > 0 is given, frequencies below this value are not printed.

43.3 Saving the hessian and other information (SAVE)

SAVE,*options*

The following *options* can be given:

hessian=*record*	Save hessian to *record* (same effect as option HESSREC). By default the hessian is saved on record 5300.2.
FREQ=*record*	Save frequencies and normal modes to *record* (same effect as option FREQREC). By default the frequencies are saved on record 5400.2.
TASK=*record*	Save task information for possible restart of hessian calculation to *record* (same effect as option TASKREC). By default the frequencies are saved on record 5500.2.

43.4 Restarting a hessian/Frequency calculation (START)

START,*options*

The following options can be given:

HESSIAN=*record*	Read hessian from record *record* (same effect as option READHESS).
TASK=*record*	Read task information from record *record* and restart numerical hessian calculation (same effect as option RESTART).

43.5 Coordinates for numerical hessian calculations (COORD)

COORD,*type*

type can be one of the following:

UNIQUE	Use symmetry-unique displacements in the numerical calculation of the hessian (default).
3N	Don't use symmetry-unique displacements (not recommended).

43.6 Stepsizes for numerical hessian calculations (STEP)

[STEP,*rstep*]

determines the step size of the numerical differentiation of the energy or the gradient. The default step size is *rstep*=0.01 a.u.

43.7 Numerical hessian using energy variables (VARIABLE)

VARIABLE,*name*;

Defines a variable *name* which holds the energy value to be used for computing the hessian using finite differences. By default, this is ENERGY(1) as set by the most recent program. For other other variables which can be used see section 42.2.17. Note that numerical hessians cannot be computed when dummy atoms holding basis functions are present.

43.8 Thermodynamical properties (THERMO)

It is also possible to calculate the thermodynamical properties of the molecule. Since MOLPRO can only handle Abelian point groups it is necessary to give the point group of the molecule in the input file:

THERMO,[SYM=*pointgroup*],[TEMP=*value*], [PRESS=*value*], [TMIN=*value*, TMAX=*value*, TSTEP=*value*]

pointgroup has to be the Schoenflies Symbol (e.g. C3v for ammonia; linear molecules have to be C*v or D*h respectively). If no point group is given, the point group is determined automatically, but only Abelian groups (D2H and subgroups) are recognized. If the molecule has higher symmetry this may eventually cause deviations in the rotational entropy.

The temperature (in K), pressure (in atm) or a range of temperatures (in K) can be given as options.

If no temperature or pressure is specified the zero-point vibrational energy and the enthalpy $H(T)-H(0)$ [kJ/mol], heat capacity C_v [J/mol K] and entropy S [J/mol K] are calculated for standard temperature and pressure ($T = 298.150$ [K], $p = 1$ [atm]).

The FREQUENCIES program sets the variable ZPE containing the zero-point-energy of the harmonic vibrations in atomic units. If the THERMO option is used, the variables HTOTAL and GTOTAL, containing the enthalpy and the free enthalpy of the system in atomic units, are also set.

43.9 Examples

```
***,formaldehyde freqency calculation
memory,8,m

basis=vdz
gthresh,energy=1.d-8

geomtyp=xyz
symmetry,nosym
geometry={
   4
FORMALDEHYDE
C          0.0000000000        0.0000000000       -0.5265526741
O          0.0000000000        0.0000000000        0.6555124750
H          0.0000000000       -0.9325664988       -1.1133424527
H          0.0000000000        0.9325664988       -1.1133424527
}

hf;accu,14
optg;coord,3n;

{frequencies,analytic
thermo,sym=c2v
print,thermo}

mp2
optg;coord,3n
{frequencies
thermo,sym=c2v
print,thermo}
```

http://www.molpro.net/info/current/examples/form_freq.com

```
***, Phosphorous-pentafluoride Vibrational Frequencies
memory,1,m
basis=3-21G

geomtyp=xyz        ! use cartesian coordinates xmol style
symmetry,nosym     ! don't use symmetry
geometry={         ! geometry input
6
  PF5
  P       0.00000       0.00000       0.00000
  F       0.00000       1.11100      -1.12400
  F       0.00000      -1.52800      -0.40100
  F       0.00000       0.41700       1.52500
  F      -1.60400       0.00000       0.00000
  F       1.60400       0.00000       0.00000}

rhf
optg              ! optimize geometry

frequencies       ! calculate vibrational frequencies
print,low         ! print frequencies+modes of zero frequencies
thermo,sym=d3h    ! calculate thermodynamical properties
temp,200,400,50   ! temperature range 200 - 400 [K]
---
```

http://www.molpro.net/info/current/examples/pf5_freq.com

44 MAGNETIC PROPERTIES OF MOLECULES

Bibliography:

S. Loibl, F.R. Manby, M. Schütz, Mol. Phys. 108, 1362 (2010)

All publications resulting from use of this program must acknowledge the above.

The command `nmrshld` calls the GIAO-DF-HF program.

The GIAO-DF-HF code performs calculations of the chemical shieldling tensor at the level of density-fitted HF. For the calculation a preceding DF-HF calculation is required. Symmetry has to be set to `nosym`. The GIAO-DF-HF code can use the canonical orbitals from the DF-HF run or localized orbitals (recommended). Example:

```
***,Chemical shielding tensors for water molecule
symmetry,nosym
GEOMETRY={    !geometry input
h1;o,h1,r1;h2,o,r2,h1,theta}
r1=1 ang
r2=1 ang
theta=102
basis={       !specify basis
default,vdz
set,jkfit
default,vdz/jkfit
}
df-hf,df_basis=jkfit
locali,pipek !localize orbitals for GIAO-DF-HF
nmrshld      !invoke calculations of shieldings
```

The GIAO-DF-HF calculation returns the chemical shielding tensors σ for each nucleus in the order they were specified in the geometry input. For each tensor the diamagnetic and paramagnetic contributions (for further information see R. Ditchfield, Mol. Phys. 27(4), 789 (1974)) as well as the total absolute shift are printed in the output.

45 MINIMIZATION OF FUNCTIONS

The minimization of general functions of one or more variables can be carried out using the command:

`MINIMIZE`, *func*, x_1[, x_2, x_3, ...]

where *func* represents a function of up to 50 variables x_1, x_2, ... Two different optimization methods can be selected as described below which do or do not use numerical derivative information.

The optimization method, as well as finer control over *func*, can be chosen using the `METHOD` directive

`METHOD`, *key* [, *key1=value, key2=value,* ...]

where *key* defines the optimization method. Valid options for *key* are:

`BFGS`	Broyden-Fletcher-Goldfarb-Shanno conjugate gradient method, which uses numerical gradients (default)
`SIMPLEX`	Downhill simplex method, which uses only function evaluations

Options to these methods, *key1, key2,* ... , are:

`VARSCALE`=*vscale*	Optimization in space of scaled variables. *vscale=0* no scaling (not recommended) *vscale=1* optimization in the space of ln(x) *vscale=2* optimization in space of initial value scaling, e.g., x_1/x_{1i} (default)
`THRESH`=*thresh*	Required accuracy of either the optimized gradient (BFGS) or function value (SIMPLEX). The default is $1 \cdot 10^{-4}$.
`VSTEP`=*epsd*	Step size for numerical gradients (BFGS) or initial SIMPLEX vertices
`PROC`=*procname*	Specifies the procedure to be executed in each optimization step. This defines a complete function evaluation (if needed, numerical gradients will be evaluated using this procedure as well)
`STARTCMD`=*command*	Specifies a start command. In each optimization step all input beginning with *command* to the current `MINIMIZE` is processed.

Miscellaneous directives

`MAXIT`=*maxit*	maximum number of optimization cycles. The default is 30 for BFGS and 100 for SIMPLEX.

45.1 Examples

45.1.1 Geometry optimization

```
***, Simple geometry optimization

basis=vdz

geometry={
O
H 1 r
H 1 r 2 theta}

r=1.8
theta=104

hf
mp2

{minimize,energy,r,theta}
---
```

http://www.molpro.net/info/current/examples/min_optgeo.com

45.1.2 Basis function optimizations

```
***, Optimization of 2 d functions

geometry={Ne}

dexp=[2.0,1.0]

basis={
sp,Ne,vdz;c;
d,Ne,dexp(1),dexp(2)
}

hf
mp2
eval=energy

minimize,eval,dexp(1),dexp(2)
---
```

http://www.molpro.net/info/current/examples/basisopt_simple.com

```
***, Optimization of 2 d functions

geometry={Ne}

dexp=[2.0,1.0]

{minimize,eval,dexp(1),dexp(2)
method,bfgs,varscale=1,thresh=1e-5,proc=optd}


proc optd

basis={
sp,Ne,vdz;c;
d,Ne,dexp(1),dexp(2)
}

hf
mp2
eval=energy

endproc
```

http://www.molpro.net/info/current/examples/basisopt_proc.com

```
***, MP2 optimization of core-valence cc-pCVDZ functions

geometry={Ne}

sexp=20.
pexp=30.

{minimize,ecv,sexp,pexp
method,bfgs,varscale=1,thresh=1e-5,proc=myopt}


proc myopt

basis={
spd,Ne,vdz;c;
s,Ne,sexp
p,Ne,pexp
}

hf
{mp2;core,1}
eval=energy

{mp2;core,0}
eall=energy

ecv=eall-eval

endproc
```

http://www.molpro.net/info/current/examples/basisopt_cv.com

46 BASIS SET EXTRAPOLATION

Basis set extrapolation can be carried out for correlation consistent basis sets using

`EXTRAPOLATE,BASIS=`*`basislist`*`,`*`options`*

where basislist is a list of at least two basis sets separated by colons, e.g. AVTZ:AVQZ:AV5Z. Some extrapolation types need three or more basis sets, others only two. The default is to use n^{-3} extrapolation of the correlation energies, and in this case two subsequent basis sets and the corresponding energies are needed. The default is not to extrapolate the reference (HF) energies; the value obtained with the largest basis set is taken as reference energy for the CBS estimate. However, extrapolation of the reference is also possible by specifying the `METHOD_R` option.

The simplest way to perform extraplations for standard methods like MP2 or CCSD(T) is to use, e.g.

```
***,H2O
memory,32,m
gthresh,energy=1.d-8

r =   0.9572 ang, theta =  104.52
geometry={O;
          H1,O,r;
          H2,O,r,H1,theta}

basis=avtz
hf
ccsd(t)
extrapolate,basis=avqz:av5z

table,basissets,energr,energy-energr,energy
head,basis,ehf,ecorr,etot
```

This will perform the first calculation with AVTZ basis, and then compute the estimated basis set limit using the AVQZ and AV5Z basis sets. The correlation energy obtained in the calculation that is performed immediately before the extrapolate command will be extrapolated (in this case the CCSD(T) energy), and the necessary sequence of calculations [here HF;CCSD(T)] will be automatically carried out.

The resulting energies are returned in variables ENERGR (reference energies), ENERGY (total energies), and ENERGD (Davidson corrected energy if available); the corresponding basis sets are returned in variable BASISSETS. The results can be printed, e.g., in a table as shown above, or used otherwise. The above input produces the table

```
 BASIS        EHF            ECORR           ETOT
 AVQZ    -76.06600082   -0.29758099   -76.36358181
 AV5Z    -76.06732050   -0.30297495   -76.37029545
 CBS     -76.06732050   -0.30863418   -76.37595468
```

The extrapolated total energy is also returned in variable ECBS (ECBSD for Davidson corrected energy if available).

In order to extrapolate the HF energy as well (using exponential extrapolation), three energies are needed. One can modify the input as follows:

```
extrapolate,basis=avtz:avqz:av5z,method_r=ex1,npc=2
```

`method_r` determines the method for extrapolating the reference energy (in this case a single exponential); `npc=2` means that only the last two energies should be used to extrapolate the correlation energy (by default, a least square fit to all given energies is used). This yields

```
 BASIS          EREF            ECORR            ETOT
 AVTZ       -76.06061330     -0.28167606     -76.34228936
 AVQZ       -76.06600082     -0.29758099     -76.36358180
 AV5Z       -76.06732050     -0.30297495     -76.37029545
 CBS        -76.06774863     -0.30863419     -76.37638283
```

Rather than using the default procedure as above, one can also specify a procedure used to carry out the energy calculation, e.g.

```
extrapolate,basis=avtz:avqz:av5z,proc=runccsd, method_r=ex1,npc=2}

procedure runccsd
hf
ccsd(t)
endproc
```

Alternatively, the energies can be provided via variables `EREF`, `ECORR`, `ETOT` etc. These must be vectors, holding as many values as basis sets are given.

46.1 Options

The possible options and extrapolation methods are:

`BASIS`=*basissets*	Specify as set of correlation consistent basis sets, separated by colons.
`PROC`=*procname*	Specify a procedure to run the energy calculations
`STARTCMD`=*command*	Start command for the energy calculations: the sequence of commands from `STARTCMD` and the current `EXTRAPOLATE` is run. `STARTCMD` must come before the extrapolate command in the input.
`METHOD`=*key*	Specifies a keyword to define the extrapolation function, see section 46.2.
`METHOD_C`=*key*	Specifies a keyword to define the extrapolation function for the correlation energy, see section 46.2.
`METHOD_R`=*key*	Specifies a keyword to define the extrapolation function for the reference energy, see section 46.2.
`VARIABLE`=*name*	Specifies a variable name; this variable should contain the energies to be extrapolated.
`ETOT`=*variable*	Provide the total energies in *variable* (a vector with the same number of energies as basis sets are given) If only ETOT but not EREF is given, the total energy is extrapolated.
`EREF`=*variable*	Provide the reference energies to be extrapolated in *variable* (a vector with the same number of energies as basis sets are given)

ECORR=*variable* Provide the correlation energies to be extrapolated in *variable* (a vector with the same number of energies as basis sets are given)

ECORRD=*variable* Provide the Davidson corrected correlation energies to be extrapolated in *variable* (a vector with the same number of energies as basis sets are given). If both ECORR and ECORRD are given, both will be extrapolated.

MINB=*number* First basis set to be used for extrapolation (default 1)

MAXB=*number* Last basis set to be used for extrapolation (default number of basis sets)

NPR=*number* If given, the last NPR values are used for extrapolating the reference energy. NPR must be smaller or equal to the number of basis sets.

NPC=*number* If given, the last NPC values are used for extrapolating the reference energy. NPC must be smaller or equal to the number of basis sets.

XR=*array* Provide a vector of exponents to be used for defining the extrapolation functional for the reference energy when using the LX functional.

XC=*array* Provide a vector of exponents to be used for defining the extrapolation functional for the correlation energy when using the LX functional.

PR=*array* Provide the constant p to be used for defining the extrapolation functional for the reference energy.

PC=*array* Provide the constant p to be used for defining the extrapolation functional for the correlation energy.

46.2 Extrapolation functionals

The extrapolation functional is chosen by a keyword with the METHOD, METHOD_R, and/or METHOD_C options. The default functional is L3. In the following, n is the cardinal number of the basis set (e.g., 2 for VDZ, 3 for VTZ etc), and x is an arbitrary number. p is a constant given either by the PR or PC options (default $p = 0$). X is a number or a vector given either by the XR or XC options (only for LX; nx is the number of elements provided in X). A, B, A_i are the fitting coefficients that are optioized by least-squares fits.

L*x* $E_n = E_{\text{CBS}} + A \cdot (n+p)^{-x}$

LH*x* $E_n = E_{\text{CBS}} + A \cdot (n+\frac{1}{2})^{-x}$

LX $E_n = E_{\text{CBS}} + \sum_{i=1}^{nx} A_i \cdot (n+p)^{-x(i)}$

EX1 $E_n = E_{\text{CBS}} + A \cdot \exp(-C \cdot n)$

EX2 $E_n = E_{\text{CBS}} + A \cdot \exp(-(n-1)) + B \cdot \exp(-(n-1)^2)$

KM Two-point formula for extrapolating the HF reference energy, as proposed by A. Karton and J. M. L. Martin, Theor. Chem. Acc. **115**, 330 (2006): $E_{\text{HF,n}} = E_{\text{HF,CBS}} + A(n+1) \cdot \exp(-9\sqrt{n})$. Use METHOD_R=KM for this.

The following example shows various possibilities for extrapolation:

```
***,h2o
memory,32,m

gthresh,energy=1.d-9
basis=avtz

r =   0.9572 ang, theta =  104.52
geometry={!nosym
          O;
          H1,O,r;
          H2,O,r,H1,theta}

hf
{ccsd(t)}
text,compute energies, extrapolate reference energy using EX1 and correlation energy using L3
extrapolate,basis=avtz:avqz:av5z,method_c=l3,method_r=ex1,npc=2

ehf=energr(1:3)
etot=energy(1:3)

text,extrapolate total energy using EX2
extrapolate,basis=avtz:avqz:av5z,etot=etot,method=ex2

text,extrapolate reference energy by EX1 and correlation energy by EX2
extrapolate,basis=avtz:avqz:av5z,etot=etot,method_c=ex2,eref=ehf,method_r=ex1

text,extrapolate reference energy by EX1 and correlation energy by LH3
extrapolate,basis=avtz:avqz:av5z,etot=etot,method_c=LH3,eref=ehf,method_r=ex1,npc=2

text,extrapolate reference energy by EX1 and correlation energy by LX
extrapolate,basis=avtz:avqz:av5z,etot=etot,method_c=LX,eref=ehf,method_r=ex1,xc=[3,4],pc=0.5
```

http://www.molpro.net/info/current/examples/h2o_extrapolate_ccsd.com

The second example shows extrapolations of MRCI energies. In this case both the MRCI and the MRCI+Q energies are extrapolated.

```
***,h2o
memory,32,m

gthresh,energy=1.d-9
basis=avtz

r =   0.9572 ang, theta =  104.52
geometry={
          O;
          H1,O,r;
          H2,O,r,H1,theta}

hf
multi
mrci
text,Compute energies, extrapolate reference energy using EX1 and correlation energy using L3;
text,The Davidson corrected energy is also extraplated
extrapolate,basis=avtz:avqz:av5z,method_c=l3,method_r=ex1,npc=2

emc=energr
ecorr_mrci=energy-emc
ecorr_mrciq=energd-emc

text,Extrapolate reference energy by EX1 and correlation energy by LH3
text,The Davidson corrected energy is also extraplated
extrapolate,basis=avtz:avqz:av5z,ecorr=ecorr_mrci,ecorrd=ecorr_mrciq,method_c=LH3,eref=emc,met
```

http://www.molpro.net/info/current/examples/h2o_extrapolate_mrci.com

46.3 Geometry optimization using extrapolated energies

Geometry optimizations are possible by using numerical gradients obtained from extrapolated energies. Analytical energy gradients cannot be used.

The following possibilities exist:

1.) If OPTG directly follows the EXTRAPOLATE command, the extrapolated energy is optimized automatically (only variable settings may occur between EXTRAPOLATE and OPTG).

Examples:

Extrapolating the energy for the last command:

```
geometry={o;h1,o,r;h2,o,r,h1,theta}
theta=102
r=0.96 ang
basis=vtz

hf
ccsd(t)
extrapolate,basis=vtz:vqz

optg
```

http://www.molpro.net/info/current/examples/h2o_extrapol_opt1.com

Extrapolating the energy computed in a procedure:

```
geometry={o;h1,o,r;h2,o,r,h1,theta}
theta=102
r=0.96 ang

proc ccsdt
hf
ccsd(t)
endproc

extrapolate,basis=vtz:vqz,proc=ccsdt

optg
```

http://www.molpro.net/info/current/examples/h2o_extrapol_opt2.com

Note that this is not possible if EXTRAPOLATE gets the input energies from variables.

2.) Using a procedure for the extrapolation:

By default, variable ECBS is optimized, but other variables (e.g. ECBSD) can be specified using the VARIABLE option on the OPTG command.

```
geometry={o;h1,o,r;h2,o,r,h1,theta}
theta=102
r=0.96 ang
basis=vtz

proc cbs34
hf
ccsd(t)
extrapolate,basis=vtz:vqz
endproc

optg,variable=ecbs,proc=cbs34
```

http://www.molpro.net/info/current/examples/h2o_extrapol_opt3.com

```
geometry={o;h1,o,r;h2,o,r,h1,theta}
theta=102
r=0.96 ang

proc cbs34
basis=vtz
hf
ccsd(t)
eref(1)=energr
ecc(1)=energy

basis=vqz
hf
ccsd(t)
eref(2)=energr
ecc(2)=energy
extrapolate,basis=vtz:vqz,eref=eref,etot=ecc
endproc

optg,variable=ecbs,proc=cbs34
```

http://www.molpro.net/info/current/examples/h2o_extrapol_opt4.com

46.4 Harmonic vibrational frequencies using extrapolated energies

This is possible by defining the extrapolation in a procedure:

```
geometry={o;h1,o,r;h2,o,r,h1,theta}
theta=102
r=0.96 ang
basis=vtz

proc cbs34
hf
ccsd(t)
extrapolate,basis=vtz:vqz
endproc

optg,variable=ecbs,proc=cbs34
freq,variable=ecbs,proc=cbs34
```

http://www.molpro.net/info/current/examples/h2o_extrapol_freq.com

47 POTENTIAL ENERGY SURFACES (SURF)

SURF,*Start1D=label1*,*options*

The SURF program allows for the calculation of the potential energy surface around the equilibrium structure as required for the calculation of anharmonic frequencies (see the VSCF and VCI programs). Currently the program is limited to the case of one minimum. The potential is represented by energy grid points rather than a Taylor expansion. Within the SURF program the potential energy surface is expanded in terms of normal coordinates, linear combination of normal coordinates or localized normal coordinates. Consequently, a harmonic frequency calculation needs to be performed first. The potential will then be represented by a hierarchical scheme given by

$$V(q_1,\ldots,q_{3N-6}) = \sum_i V_i(q_i) + \sum_{i<j} V_{ij}(q_i,q_j) + \sum_{i<j<k} V_{ijk}(q_i,q_j,q_k) + \ldots \tag{64}$$

with

$$V_i(q_i) = V_i^0(q_i) - V(0) \tag{65}$$

$$V_{ij}(q_i,q_j) = V_{ij}^0(q_i,q_j) - \sum_{r\in\{i,j\}} V_r(q_r) - V(0) \tag{66}$$

$$V_{ijk}(q_i,q_j,q_k) = V_{ijk}^0(q_i,q_j,q_k) - \sum_{\substack{r,s\in\{i,j,k\}\\ r>s}} V_{rs}(q_r,q_s) - \sum_{r\in\{i,j,k\}} V_r(q_r) - V(0) \tag{67}$$

$$V_{ijkl}(q_i,q_j,q_k,q_l) = \ldots \tag{68}$$

where q_i denotes the coordinates. This expansion needs to be terminated after an n-body contribution as controlled by the keyword NDIM. The SURF program is fully parallelized in a sense that the calculation of different grid points is send to different processors (MPPX scheme). The START1D keyword is mandatory and defines the label where to jump in the input in order to do an electronic structure calculation which is terminated by the SURF command. This way the quality of the potential energy surface is defined.

```
label1
hf
ccsd
surf,start1D=label1
```

The SURF program is based on a iterative algorithm, i.e. grid points will be added automatically to the grid representation of the potential until a convergence threshold will be met. This guarantees a well-balanced description of the different terms in the expansion of the potential and simultaneously minimizes the number of *ab initio* calculations for a representation of the potential. For further details see:

G. Rauhut, *Efficient Calculation of Potential Energy Surfaces for the Generation of Vibrational Wave Functions*, J. Chem. Phys. **121**, 9313 (2004).
T. Hrenar, H.-J. Werner, G. Rauhut *Accurate Calculation of Anharmonic Vibrational Frequencies of Medium Sized Molecules Using Local Coupled Cluster Methods*, J. Chem. Phys. **126**, 134108 (2007).

47.1 Options

The following *options* are available:

NDIM=*value* — The keyword NDIM=*n* terminates the expansion of the PES after the *n*-body term. Currently, at most 4-body terms can be included, but the default is set to 3. Please note, when you use NDIM=4 as a keyword for the SURF program, you need to pass this information to the VSCF and VCI programs also. Otherwise these programs will neglect the 4-body terms.

NGRID=*value* — Based on a coarse grid of *ab initio* points a fine grid will be generated from automated interpolation techniques. The keyword NGRID=*n* determines the number of equidistant grid points in one dimension. NGRID=*n* has to be an even number. The default is currently set to 16. Note that the number of grid points also controls the extension of the *n*-dimensional potential energy surfaces (see keyword SCALE) and thus influences many internal thresholds which are optimized to the default value of NGRID. The number of grid points also determines the number of basis functions in the VSCF program. At present the maximum grid size is 20.

Grid points	14	16	18	20
Surface extension	4.30	4.69	5.05	5.39

VAR1D=*variable* — The SURF program reads the energy of electronic structure calculations from the internal MOLPRO variables, e.g. ENERGY, EMP2, ... The internal variable is specified by the keyword VAR1D. Within the example shown above, VAR1D=ENERGY would read the CCSD energy, while VAR1D=EMP2 would read the MP2 energy, which is a byproduct of the CCSD calculation. The default for the VAR1D keyword is the internal variable ENERGY.

DIPOLE=*value* — Dipole surfaces can be computed for all those methods for which analytical gradients are available in MOLPRO. For all methods except Hartree-Fock this requires the keyword CPHF,1 after the keyword for the electronic structure method. The calculation of dipole surfaces is limited to those multi-level schemes, for which all variables VAR1D, VAR2D and VAR3D are set to the default, i.e. ENERGY (see the section about the VMULT command). Symmetry is currently only implemented for the 1D, 2D and 3D dipole surfaces. For 4D terms symmetry will automatically switched off at the moment. The calculation of dipole surfaces effectively doubles the computation time for surface calculations.

SYM=*value* — Symmetry within electronic structure calculations can be exploited by the keyword SYM=Auto. Usually this leads to significant time savings. By default this symmetry recognition is switched off as certain calculations may cause some trouble (e.g. local correlation methods). Symmetry in electronic structure calculations may not be mistaken for symmetry of the contributions to the potential energy surface (see keyword MPG).

SCALE=*value* — The extension of the potential energy surfaces is determined from Gauss-Hermite quadrature points. Using a fine grid NGRID=16 the surface stretches out to the NGRID/2^{th} Gauss-Hermite point, i.e. 4.69, in each direction (see keyword NGRID). As these values are fairly large within the calculation of fundamental modes, a scaling factor, SCALE=f, has been introduced. A default scaling of 0.75 is used. Increasing the size of the surfaces usually requires the calculation of further *ab initio* points as the surface interpolation is more stable for small surfaces.

THRFIT=*value* The iterative algorithm for generating potential energy surfaces is based on a successive increase of interpolation points. The iterations are terminated once the interpolation of two subsequent iteration steps became stable. The convergence threshold can be changed by the keyword THRFIT=f. There is currently just one control variable for the different 1D, 2D, 3D, and 4D iterations. The 4 thresholds are different but depend on each other. Consequently, changing the default value (THRFIT=4.0d-2) will change all thresholds simultaneously which keeps the calculation balanced.

FIT1D=*value* The maximum order of the polynomials used for fitting within the iterative interpolation scheme can be controlled by the keywords FIT1D, FIT2D, FIT3D, FIT4D. The default is given by 8. However in certain cases higher values may be necessary, but require an appropriate number of coarse grid points, which can be controlled by MIN1D etc.

MIN1D=*value* The minimum number of coarse grid points can be controlled by the keywords MIN1D, MIN2D, MIN3D, MIN4D. These 4 keywords determine the minimum number of *ab initio* calculations in one dimension for each 1D, 2D, 3D and 4D surface. The defaults are currently MIN1D=4, MIN2D=4, MIN3D=2, MIN4D=2.

MAX1D=*value* The maximum number of coarse grid points can be controlled by the keywords MAX1D, MAX2D, MAX3D, MAX4D. These 4 keywords determine the maximum number of *ab initio* calculations in one dimension for each 1D, 2D, 3D and 4D surface. The defaults are currently MAX1D=NGRID, MAX2D=NGRID, MAX3D=10, MAX4D=4.

EXT12D=*value* Outer regions of the potential energy surfaces are determined by extrapolation rather than interpolation schemes. An extrapolation of 15% in case of the 1D and 2D contributions to the potential (Ext12D=0.85) and of 30% in case of the 3D and 4D terms (Ext34D=0.7) is currently used by default. For accurate calculations extrapolation can be switched off by setting both keywords to 1.0. Changing these keywords usually increases the number of *ab initio* calculations to be performed.

SKIP3D=*value* As the number of 3D and 4D surfaces can increase very rapidly, there exists the possibility to neglect unimportant 3D and 4D surfaces by the keywords SKIP3D and SKIP4D. The criterion for the prescreening of the 3D surfaces is based on the 2D terms and likewise for the 4D terms the 3D surfaces are used. The neglect of 3D surfaces automatically leads to the neglect of 4D surfaces, as the latter depend on the previous ones. By default prescreening is switched on, but can be switched off by SKIP3D=1.0 and SKIP4D=1.0.

MPG=*value* Symmetry of the normal modes should be recognized by the program automatically. Only Abelian point groups can be handled at the moment. Symmetry of the modes will be determined even if the NOSYM keyword is used in the electronic structure calculations. In certain cases numerical noise can be very high and thus prohibits a correct determination of the symmetry labels. This is denoted by the label Err for these modes. In such cases symmetry should be switched off in the calculation of the potential energy surfaces and the VCI calculations, because the results may be corrupted. Symmetry can be switched off by using MPG=1.

INFO=*value* — INFO=1 provides a list of the values of all relevant program parameters (options).

The following example shows the input of a calculation which computes energy and dipole surfaces at the MP2/cc-pVTZ level and subsequently determines the anharmonic frequencies at the VSCF and VCI levels. Hartree-Fock calculations will not be restarted and the .log-file is directed to the scratch directory as defined by the $TMPDIR variable.

```
memory,20,m
geomtyp=xyz
orient,mass
geometry={
   3
Water
O          0.0675762564        0.0000000000       -1.3259214590
H         -0.4362118830       -0.7612267436       -1.7014971211
H         -0.4362118830        0.7612267436       -1.7014971211
}

mass,iso
basis=vdz
logfile,scratch

hf
mp2
optg
{frequencies,symm=auto
 print,low=50}

label1
{hf
 start,atden}
{mp2
 cphf,1}

surf,start1D=label1,dipole=1,sym=auto
vscf,dipole=1,combi=1
vci,dipole=1,version=3,combi=1
```

47.2 Multi-level calculations

VMULT,*options*

The level of the electronic structure calculations can be changed for the different *i*-body terms in the expansion of the potential. As a consequence, the keywords START2D, START3D, VAR2D and VAR3D exist in full analogy to the keywords START1D and VAR1D in standard calculations (see above). The number always represents the level of the expansion term. Such calculations are termed multi-level calculations. There does *not* exist a corresponding set of keywords for the 4-body terms. 4-body terms will always use the variables specified for the 3-body terms.

`MULTI`=*value* The keywords `START1D`, `START2D`, `START3D` in combination with the commands `VAR1D`, `VAR2D` and `VAR3D` allow for the calculation of multi-level potential energy surfaces. This would imply in principle that the 1D term of the potential needs to be computed at all three levels and the 2D term at two computational levels. As certain low level results are a byproduct of more sophisticated methods (e.g. the HF energy is a byproduct of a MP2 calculation or the MP2 energy is a byproduct of a CCSD(T) calculation) the computational overhead can be avoided by the `MULTI` option.

`MULTI=1`: This is the default and most expensive choice. The 1D potential will be computed at all 3 levels of theory. Likewise, the 2D potential will be calculated at 2 levels explicitly. An example would be:

```
1D: CCSD(T)/cc-pVTZ
2D: MP4(SDQ)/cc-pVTZ
3D: MP2/cc-pVDZ

{SURF,Start1D=label1
 VMULT,Start2D=label2,Start3D=label3,Multi=1}
```

`MULTI=2`: All information is provided by the preceding calculations and thus no part of the potential has to be computed twice. Examples:

```
1D: CCSD(T)/cc-pVTZ
2D: CCSD(T)/cc-pVTZ
3D: MP2/cc-pVTZ

{SURF,Start1D=label1
 VMULT,Start2D=label1,Start3D=label2
 VMULT,Var3D=EMP2,Multi=2}
```

```
1D: CCSD(T)/cc-pVTZ
2D: MP2/cc-pVTZ
3D: MP2/cc-pVTZ

{SURF,Start1D=label1
 VMULT,Start2D=label2,Start3D=label2
 VMULT,Var2D=EMP2,Var3D=EMP2,Multi=2}
```

`MULTI=3`: The 2D potential provides all information for the 3D part while there is no connection between 1D and 2D. Consequently, The 1D contributions need to be computed twice (at the 1D and 2D levels) while all other terms will be computed just once. Examples:

```
1D: CCSD(T)/cc-pVTZ
2D: MP4(SDQ)/cc-pVTZ
3D: MP2/cc-pVTZ

{SURF,Start1D=label1
 VMULT,Start2D=label2,Start3D=label3
 VMULT,Var3D=EMP2,Multi=3}
```

```
1D: CCSD(T)/cc-pVTZ
2D: MP4(SDQ)/cc-pVTZ
3D: MP4(SDQ)/cc-pVTZ

{SURF,Start1D=label1
 VMULT,Start2D=label2,Start3D=label2,Multi=3}
```

MULTI=4: The 1D calculation provides all information for the 2D potential but does not so for the 3D part. Hence, the 1D contribution and the 2D contributions need to be computed twice. Examples:

```
1D: CCSD(T)/cc-pVTZ
2D: CCSD(T)/cc-pVTZ
3D: MP4(SDQ)/cc-pVTZ

{SURF,Start1D=label1
 VMULT,Start2D=label1,Start3D=label2,Multi=4}

1D: CCSD(T)/cc-pVTZ
2D: MP2/cc-pVTZ
3D: MP2/cc-pVDZ

{SURF,Start1D=label1
 VMULT,Start2D=label2,Start3D=label3
 VMULT,Var2D=EMP2,Multi=4}
```

In 2D and 4D calculations (i.e. NDIM=2,4) the VMULT command can be used as well. In 4D calculations the last level must always be identical to the 3D level. In 2D the meaning of MULTI=1 and MULTI=3 is the same. Likewise, MULTI=2 and MULTI=4 are the same in case of 2D calculations.

START2D=*label* — START2D and START3D define labels in the input stream in order to compute the 2D and 3D terms at different levels of electronic structure theory than the 1D terms. The use of the START2D and START3D commands usually requests the use of GOTO commands in the input.

VAR2D=*variable* — The keywords VAR2D and VAR3D are defined in full analogy to the VAR1D option. They specify the internal variable (e.g. ENERGY, EMP2, CCSD, ...) to be read out for a given grid point.

The following example shows a 1D:CCSD(T)/cc-pVTZ; 2D:MP4(SDQ)/cc-pVTZ and 3D:MP2/cc-pVTZ multi-level calculation. As the MP2 energy is a byproduct of the CCSD(T) and MP4(SDQ) calculations only the 1D grid points will be computed twice (at the CCSD(T) and MP4(SDQ) levels). The 1D and 2D energies will be obtained from the internal variable ENERGY while the 3D energies make use of the EMP2 variable.

```
memory,50,m
geomtyp=xyz
orient,mass
geometry={
6
Ethen
C          0.0000000000        0.0000000000       -0.6685890718
C          0.0000000000        0.0000000000        0.6685890718
H          0.0000000000       -0.9240027061       -1.2338497710
H          0.0000000000        0.9240027061       -1.2338497710
```

```
H          0.0000000000        0.9240027061        1.2338497710
H          0.0000000000       -0.9240027061        1.2338497710
}

mass,iso
basis=vtz
logfile,scratch

hf
ccsd(t)
optg
freq,symm=auto

label1
hf
ccsd(t)
goto,label4

label2
{hf
 start,atden}
{mp4
 notripl}
goto,label4

label3
{hf
 start,atden}
mp2

label4
{surf,start1D=label1,sym=auto
 vmult,start2D=label2,start3D=label3,Var3D=EMP2,Multi=3}
vscf
vci
```

47.3 Restart capabilities

DISK,*options*

As SURF calculations are very demanding it is highly recommended to dump the grid representation of the potential to disk. Two different options are available: (1) dumping to the binary .wfu-file (options SAVE and RESTART) and (2) dumping to an external ASCII-file (options WHERE, DUMP and DISK). Restarts from the binary file are only possible in case that the SURF calculation has been finished. For crashed SURF calculations restarts are feasible from the external ASCII-file.

SAVE=*value* Once a permanent file 2 has been declared in the Molpro input (file,2,filename.wfu), by default the potential generated by the SURF program will be dumped to disk. The default record is 5600.2, but this value can be changed by the SAVE command. This allows for running anharmonic frequency calculations in a separate job without an explicit recalculation of the potential.

RESTART=*value* Once an entire surface calculation has been completed, the calculation can be restarted from the binary file 2 specified in the MOLPRO input (see the SAVE command) using the RESTART option, e.g.

Restart=5600.2. Restarts using the RESTART option will fail for broken SURF calculations, only restarts for finished SURF calculations are possible. Note that, for restarting a calculation, you always need to specify a SURF card. You cannot restart a calculation simply beginning with a VSCF card, although the potential may be completely available on record 5600.2.

WHERE=*value* In combination with the keywords DUMP and EXTERN for an external restart file, the keyword WHERE specifies the path for the external ASCII file. Two options are available, WHERE=home and WHERE=scr. As the external files can be huge for SURF calculations, they will be stored on the scratch disk given by the MOLPRO variable $TMPDIR by default.

DUMP=*file name* The potential can be dumped into an external ASCII-file which can be used for restarting. Its name must be provided as the argument of the DUMP keyword, e.g. DUMP='formaldehyde.pot'. The ASCII-file provides the interface to other programs and offers the possibility for controlled storage and modification of the computed potentials. Dipole surfaces will also be dumped if available.

EXTERN=*file name* In principle, SURF calculations should be restartable at any point of a truncated calculation. However, since only fully converged difference potentials will be stored in the ASCII file (DUMP), this is not the case. As a consequence, SURF calculations can be restarted from dump-files once a batch of surfaces has been dumped. Since the generation of the 2D or 3D surfaces usually requires about 2 to 3 batches, there are about 6-10 restart points for surface calculation including 3D potentials. Restarting from the ASCII dump-file is possible for any type of VMULT calculation. As normal modes and harmonic frequencies will also be read in from the external file, harmonic frequency calculations need not to be repeated for such restarts.

47.4 Linear combinations of normal coordinates

LINCOMB,*options*

The LINCOMB directive allows for the calculation of linear combinations of normal coordinates for the expansion of the potential. This is realized by 2x2 Jacobi rotations. At most 3N-6/2 rotations can be provided in the input.

NM1=*n*, NM2=*m* Denote the two normal coordinates to be rotated.

ANGLE=*value* Rotation angle in degree.

47.5 Scaling of individual coordinates

SCALNM,*options*

The SCALE option of the SURF program enables a modifaction of the extension of all difference potentials by a common factor. In contrast to that the SCALNM directive allows for the scaling with respect to the individual normal coordinates. This is the recommended choice for potentials dominated by quartic rather than quadratic terms. At most 3N-6 individual scale factors can be provided.

MODE=*n*	Denotes the normal coordinate to be scaled.
SFAC=*value*	Scaling factor for mode MODE. The default is 1.0.

47.6 Modeling of high-order *n*-body terms

REPAR,*options*

Within the framework of multi-level calculations (see the directive VMULT), 3D and 4D terms can be modeled. The modeling scheme is based on a reparametrization of the semiempirical AM1 method. Consequently, in the input stream the energy variable to be read in must refer to a semiempirical calculation. After the 2D terms the program optimizes the semiempirical parameters in order to represent the 1D and 2D surfaces best.

TYPE=*n*	TYPE=1 specifies a standard reparametrization solely based on a local optimization starting from standard semiempirical parameters. TYPE=2 invokes a global optimization prior to the local one.
RMS1D=*n*	The keywords RMS1D and RMS2D specify the threshold for terminating the 1D and 2D iterations in the local optimization of the semiempirical parameters. The defaults are given by RMS1D=1.d-6 and RMS2D=1.d-6.
ITMAX1D=*n*	The maximum number of iterations in the local optimization of the semiempirical parameters can be controlled by ITMAX1D and ITMAX2D. The defaults are ITMAX1D=100 and ITMAX2D=150.

The following example shows the input for a surface calculation in which the 3D terms will be modeled without the use of global optimization schemes.

```
memory,20,m
geomtyp=xyz
orient,mass
geometry={
   3
Water
O          0.0675762564        0.0000000000       -1.3259214590
H         -0.4362118830       -0.7612267436       -1.7014971211
H         -0.4362118830        0.7612267436       -1.7014971211
}

hf
mp2
optg
freq

label1
abinitio
basis=vdz
int
rhf
mp2
goto,label4

label2
semi,am1
int
```

```
rhf

label4
{surf,start1D=label1
 vmult,start2D=label1,start3D=label2,multi=4
 repar,type=1}
vscf
vci
```

47.7 Quality Check

`CHECK`,*options*

The `CHECK` directive of the `SURF` program allows for a quality check of a completed surface. This routine simply computes the exact *ab initio* energies at randomly selected grid points and compares these values with the interpolated ones, which will be used subsequently for the determination of the wavefunction. This program is fully parallelized.

`LEVEL`=*n*	Denotes the level to be checked, i.e. 1 corresponds to 1D, etc.
`PONTS`=*value*	Determines the number of grid points in one dimension to be checked. The default is set to 4.

47.8 Recommendations

It is recommended to

- use the `MASS` keyword within the geometry specification in order to rotate the molecule into standard orientation. This is necessary for a full exploitation of symmetry within the generation of the potential energy surface.
- split up the calculations in three parts: (1) Do a very accurate geometry optimization and bring the optimized geometry into standard orientation. Start a second calculation based on the optimized geometry and do (2) a harmonic frequency calculation which should be stored in the restart-file (.wfu). Make a backup of the restart-file. (3) Start a `SURF`, `VSCF` and `VCI` calculation from the restart-file (.wfu) and dump the potential energy surfaces to an external file. Later on you can restart your `VSCF` and `VCI` calculations from this external file without recomputing the surface.
- use a 1D: CCSD(T)/vtz, 2D: MP4(SDQ)/vtz and 3D: MP2/vtz multi-level scheme in combination with symmetry and a parallelized MOLPRO version in order to speed up the calculations.
- reduce the quality of the normal coordinates. If you do not require extremely high accuracy, it is sufficient to compute the normal coordinates (i.e. the harmonic frequencies) at the MP2 level rather than the CCSD(T) level. This saves a lot of computer time and the deviations are usually not larger than 1 or 2 wavenumbers.

47.9 Standard Problems

- `Problem:` The SURF calculation crashes with an error message like

```
?ERROR IN VIRTORB: INCORRECT NUMBER OF ORB...

ERROR EXIT
CURRENT STACK:      MAIN
```

`Solution:` The program has problems in the symmetry conversion when restarting a Hartree-Fock calculation from the reference calculation at the equilibrium geometry. You need to start the Hartree-Fock calculations independently by using the keywords `start,atden`.

- `Problem:` In parallel calculations (MPPX) the CPU-time of a `VSCF` calculation differs considerably from the real-time (wallclock time).

  `Solution:` There may be two reasons for this: (1) Usually a `VSCF` calculation spends a significant amount of the total time in the Hartree-Fock program and the 2-electron integrals program. As the integrals are stored on disk, 2 processes on the same machine may write on disk at the same time and thus the calculation time depends to some extend on the disk controller. It is more efficient to stripe several disks and to use several controllers. This problem can be circumvented by distributing the job over several machines, but limiting the number of processors for each machine to 1. (2) The integrals program buffers the integrals. Parallel jobs may require too much memory (factor of 2 plus the shared memory) and thus the integrals buffering will be inefficient. Try to reduce the memory as much as you can. It might be advantageous to separate the memory demanding `VCI` calculation from the `SURF` calculation.

48 POLYNOMIAL REPRESENTATIONS (POLY)

POLY,*options*

The POLY program allows for the transformation of the potential energy surface and dipole surfaces from a grid representation to a polynomial representation. Once a polynomial representation has been chosen, the corresponding VSCF, VCI or VMP2 programs need to be selected (see below). The POLY program is fully parallelized in terms of the MPPX scheme.

48.1 Options

The following *options* are available:

NDIM=*value* — The keyword NDIM=*n* terminates the transformation after the *n*D terms within the *n*-mode expansion of the surfaces. The default is set to 3. The transformation of the 4D terms can be very time consuming.

NGRID=*value* — Once the value of the NGRID=*n* keyword for controlling the number of grid points has been changed in the SURF program, this information needs to be passed to the POLY program.

ORDPOL=*value* — This keyword controls the order of the polynomial to be determined in the fitting procedure. The default is set to 6.

ORDPROD=*value* — In multidimensional fitting the maximum order of the polynomial is not given by ORDPOL times the dimension, but rather by ORDPROD. The default is set to 12. This reduces the computational cost for fitting tremendously. However, for accurate calculations ORDPROD=8 and ORDPOL=18 are recommended.

DIPOLE=*value* — In case that dipole surfaces have been computed, they need to be transformed to a polynomial representation as well. This can be accomplished by DIPOLE=1.

PMP=*value* — The Watson correction term, i.e. PMP=1, usually is absorbed in the potential. Once the polynomials to be generated should include this correction, this needs to be specified. Any other values, i.e. 2 or 3, will be ignored.

SHOW=*value* — The coefficients of the polynomial representation can be printed. In order to identify quartic potentials, it is recommended to use SHOW=1. Higher values will lead to a very long output file.

49 THE VSCF PROGRAM (VSCF)

VSCF,*options*

The VSCF program is based on the Watson Hamiltonian

$$\hat{H} = \frac{1}{2}\sum_{\alpha\beta}(\hat{J}_\alpha - \hat{\pi}_\alpha)\mu_{\alpha\beta}(\hat{J}_\beta - \hat{\pi}_\beta) - \frac{1}{8}\sum_{\alpha}\mu_{\alpha\alpha} - \frac{1}{2}\sum_{i}\frac{\partial^2}{\partial q_i^2} + V(q_1,\ldots,q_{3N-6}) \qquad (69)$$

in which the potential energy surfaces, $V(q_1,\ldots,q_{3N-6})$, are provided by the SURF module. Vibrational angular momentum terms are switched off by default. Within the grid-based version of the program the one-dimensional Schrödinger equation is solved by the DVR procedure of Hamilton and Light. Note that, the number of basis functions (distributed Gaussians) is determined by the grid points of the potential and cannot be increased without changing the PES grid representation. In contrast to that the number of basis functions can be modified without restrictions in the polynomial based version. In all cases the basis is fixed to distributed Gaussians (DG). As VSCF calculations are extremely fast, these calculations cannot be restarted. For details see:

G. Rauhut, T. Hrenar, *A Combined Variational and Perturbational Study on the Vibrational Spectrum of* P_2F_4, Chem. Phys. **346**, 160 (2008).

49.1 Options

The following *options* are available:

TYPE=*value* — VSCF solutions can be obtained using a potential in grid representation, i.e. TYPE=GRID, or in a polynomial representation, TYPE=POLY. In the latter case the POLY program needs to be called prior to the VSCF program in order to transform the potential.

PMP=*value* — Vibrational angular momentum terms, i.e. $\frac{1}{2}\sum_{\alpha\beta}\hat{\pi}_\alpha\mu_{\alpha\beta}\hat{\pi}_\beta$, and the Watson correction term are by default switched off. PMP=1 adds the Watson correction term (see eq. 69) as a pseudo-potential like contribution to the fine grid of the potential. PMP=2 allows for the calculation of the integrals of the PMP operator using the approximation that the μ tensor is given as the inverse of the moment of inertia tensor at equilibrium geometry. When using PMP=4 the expansion of the effective moment of inertia tensor will be truncated after the 1D terms (rather than the 0D term in case of PMP=2. Note that the values higher than 2 are only active for non-linear molecules. PMP=5 truncates the series after the 2D term. In almost all cases PMP=2 is fully sufficient. Vibrational angular momentum terms are accounted for in a perturbational manner and do not affect the wavefunction.

COMBI=*value* — By default the VSCF program calculates the fundamental modes of the molecule only. However, choosing COMBI=1 allows for the calculation of the first vibrational overtones and $n\times(n-1)/2$ combination bands consisting of two modes in the first vibrational level.

SOLVER=*value* — For solving the one-dimensional Schrödinger equation within a grid representation two different algorithms can be used. The default, i.e.

	SOLVER=1, calls the discrete variable representation (DVR) as proposed by Hamilton and Light. Alternatively, the collocation algorithm of Young and Peet can be used (SOLVER=2).
THERMO=*value*	THERMO=1 allows for the improved calculation of thermodynamical quantities (compare the THERMO keyword in combination with a harmonic frequency calculation). However, the approach used here is an approximation: While the harmonic approximation ist still retained in the equation for the partition functions, the actual values of the frequencies entering into these functions are the anharmonic values derived from the VSCF calculation.
ROTJ=*value*	Rovibrational levels can be computed in an approximative fashion only (this does not work in combination with the COMBI option). Once the VSCF equations have been solved, the rotational constants will be computed from the vibrationally averaged structures for each vibrational level. This allows for a rough estimate and very fast calculation of the rovibrational levels. ROTJ=*n* determines the value of J. A negative number for J results in a calculation of all rovibrational levels from $J = 1$ up to the specified J value.
DIPOLE=*value*	DIPOLE=1 allows for the calculation of infrared intensities. Calculation of infrared intensities requires the calculation of dipole surfaces within the SURF program. By default intensities will not be computed.
NDIM=*value*	The expansion of the potential in the VSCF calculation can differ from the expansion in the SURF calculation. However, only values less or equal to the one used in the surface calculation can be used.
NBAS=*value*	The number of basis functions (distributed Gaussians) to be used for solving the VSCF equations can controlled by NBAS=*value*. The default is NBAS=20. This option is only active once a polynomial representation of the potential has been chosen, see the option TYPE=POLY and the POLY program.
ORDPOL=*val*	Once the default value has been changed in the POLY program, this value needs to be changed here as well.
ORDPROD=*val*	Once the default value has been changed in the POLY program, this value needs to be changed here as well.
SHOW=*value*	SHOW=1 prints the effective 1D polynomials in case that the potential is represented in terms of polynomials, see the option TYPE=POLY and the POLY program.
INFO=*value*	INFO=1 provides a list of the values of all relevant program parameters (options).

50 THE VCI PROGRAM (VCI)

VCI,*options*

VCI calculations account for vibration correlation effects and are based on potential energy surface as generated from the SURF program) and a basis of VSCF modals. All VCI calculations will be performed state-specific, i.e. for each vibrational mode an individual VCI calculation will be performed. As VCI calculations may require substantial computer resources, these calculations can be rather expensive. Currently, two different VCI programs (configuration selective and conventional) are available (see below). Moreover, VCI calculations can be performed using the grid-based version of the program or within a polynomial representation. The latter is significantly faster and is thus recommended. The different versions of the configuration selection VCI program and the underlying configuration selection scheme are described in detail in:

M. Neff, G. Rauhut, *Toward large scale vibrational configuration interaction calculations*, J. Chem. Phys. **131**, 124129 (2009).

50.1 Options

The following *options* are available:

TYPE=*value* — VCI solutions can be obtained using a potential in grid representation, i.e. TYPE=GRID, or in a polynomial representation, TYPE=POLY. In the latter case the POLY program needs to be called prior to the VSCF and VCI programs in order to transform the potential.

VERSION=*value* — Both, the grid-based and the polynomial-based versions of the VCI programs offer 4 different kinds of VCI implementations: VERSION=1 is the fastest program. It works state-specific and configuration selective and makes use of a simultaneous exclusion and internal contraction scheme (see the reference given above). VERSION=2 is identical with VERSION=1 but switches off the internal contractions. VERSION=3 is the most accurate configuration selective VCI program and does neither use the internal contraction scheme nor the simultaneous exclusion. VERSION=4 is a conventional VCI program without any of the aforementioned approximations. It is thus computationally extremely demanding.

CITYPE=*value* — CTYPE defines the maximum number of simultaneous excitations, i.e. Singles, Doubles, Triples, ... and thus determines the kind of calculations, i.e. VCISD, VCISDT, ... The default is CITYPE=4 (VCISDTQ), which appears to be a fair compromise between accuracy and computational speed. The maximum excitation level is currently limited to CITYPE=6.

LEVEX=*value* — LEVEX determines the level of excitation within one mode, i.e. $0 \rightarrow 1$, $0 \rightarrow 2$, $0 \rightarrow 3$, ... The default is LEVEX=4, which was found to be sufficient for many applications.

CIMAX=*value* — CIMAX is the maximum excitation level corresponding to CITYPE and LEVEX. In principle, a triple configuration $(4,4,4)$ would contribute to the VCI space. However, CIMAX=7 restricts this to $(4,3,0)$, $(3,3,1)$, $(3,2,2),\ldots$. The default is CIMAX=8, which needs to be extended for certain applications.

CITHR=*value* — CITHR controls the threshold within the configuration selection scheme (see VERSION=1-3). The default is 0.99995.

PMP=*value* — Vibrational angular momentum terms (Coriolis coupling), i.e. $\frac{1}{2}\sum_{\alpha\beta}\hat{\pi}_\alpha\mu_{\alpha\beta}\hat{\pi}_\beta$, and the Watson correction term are by default switched off. PMP=1 adds the Watson correction term (see eq. 69) as a pseudo-potential like contribution to the fine grid of the potential. PMP=2 allows for the calculation of the integrals of the PMP operator using the approximation that the μ tensor is given as the inverse of the moment of inertia tensor at equilibrium geometry. The PMP=2 option includes diagonal contributions in the VCI matrix only. This is significantly faster than calculating the contributions for all matrix elements (which corresponds to PMP=3 and usually introduces only small deviations. PMP=4 extends the constant μ-tensor (0D) by 1D terms. PMP=5 introduces 2D corrections to the μ-tensor. PMP=4 and PMP=5 make use of a prescreening technique in which the convergence of the PMP operator will be checked for each VCI matrix element. PMP=8 corresponds to PMP=5 without prescreening. Note that the 1D and 2D corrections increase the computational cost considerably and are only available for non-linear molecules.

COMBI=*value* — By default the VCI program calculates the fundamental modes of the molecule only. However, choosing COMBI=1 allows for the calculation of the first vibrational overtones and $n \times (n-1)/2$ combination bands consisting of two modes in the first vibrational level.

THERMO=*value* — THERMO=1 allows for the improved calculation of thermodynamical quantities (compare the THERMO keyword in combination with a harmonic frequency calculation). However, the approach used here is an approximation: While the harmonic approximation ist still retained in the equation for the partition functions, the actual values of the frequencies entering into these functions are the anharmonic values derived from the VCI calculation.

ROTJ=*value* — Rovibrational levels can be computed in an approximative fashion only (this does not work in combination with the COMBI option). Once the VCI wave function has been determined, the rotational constants will be computed from the vibrationally averaged structures for each vibrational level. This allows for a rough estimate and very fast calculation of the rovibrational levels. ROTJ=*n* determines the value of J. A negative number for J results in a calculation of all rovibrational levels from $J = 1$ up to the specified J value.

DIPOLE=*value* — DIPOLE=1 allows for the calculation of infrared intensities. Calculation of infrared intensities requires the calculation of dipole surfaces within the SURF program. By default intensities will not be computed.

NDIM=*value* — The expansion of the potential in the VCI calculation can differ from the expansion in the SURF calculation. However, only values less or equal to the one used in the surface calculation can be used.

MPG=*value* — Symmetry of the molecule will be recognized automatically within the VCI calculations. MPG=1 switches symmetry off.

NBAS=*value* — The number of basis functions (distributed Gaussians) to be used for obtaining the VCI solutions can be controlled controlled by NBAS=*value*. The number of basis functions must be identical to the number used in the VSCF program. The default is NBAS=20. This option is only active once a polynomial representation of the potential has been chosen, see the option TYPE=POLY and the POLY program.

DIAG=*value* In the polynomial configuration selective VCI program different diagonalization schemes can be used. DIAG=CON specifies a conventional non-iterative diagonalization as used in the grid-based versions. DIAG=JAC is the default and uses a Jacobi-Davidson scheme. DIAG=HJD denotes a disk-based Jacobi-Davidson algorithm.

INFO=*value* INFO=1 provides a list of the values of all relevant program parameters (options).

50.2 Recommendations

It is recommended to split up the calculations in three parts: (1) Do a very accurate geometry optimization and bring the optimized geometry into standard orientation (using the MASS keyword. Start a second calculation based on the optimized geometry and do (2) a harmonic frequency calculation which should be stored in the restart-file (.wfu). Make a backup of the restart-file. (3) Start a SURF, VSCF and VCI calculation from the restart-file (.wfu) and dump the potential energy surfaces to an external file (using the DUMP keyword). Later on you can restart your VSCF and VCI calculations from this external file without recomputing the surface.

50.3 Examples

The following input example (1) optimizes the geometry of water, (2) computes the harmonic frequencies, (3) generates a potential energy surface around the equilibrium structure, (4) computes the nuclear wave function and the infrared intensities at the VSCF level, and finally (5) performs three different VCI calculations using different configuration selection schemes. Vibrational angular momentum terms are included.

```
memory,20,m
basis=vdz
orient,mass
geomtyp=xyz
geometry={
   3
Water
O          0.0675762564        0.0000000000       -1.3259214590
H         -0.4362118830       -0.7612267436       -1.7014971211
H         -0.4362118830        0.7612267436       -1.7014971211
}

hf
mp2
optg
{frequencies,symm=auto
print,low=50}

label1
{hf
start,atden}
{mp2
cphf,1}
```

```
surf,start1D=label1,dipole=1,sym=auto
vscf,pmp=2,dipole=1
vci,pmp=3,dipole=1,version=1
vci,pmp=3,dipole=1,version=2
vci,pmp=3,dipole=1,version=3
```

51 VIBRATIONAL MP2 (VMP2)

VMP2,*options*

The VMP2 program allows to perform 2nd order vibrational Møller-Plesset calculations. The program has been implemented in a grid-based and a polynomial-based version. Most of the keywords as described for the VCI program are also valid for the VMP2 program, i.e. TYPE, CITYPE, LEVEX, CIMAX, NGRID, NDIM, NBAS, PMP, COMBI, THERMO, DIPOLE, ROTJ, MPG and INFO.

```
memory,20,m
basis=vdz
orient,mass
geomtyp=xyz
geometry={
   3
Water
O          0.0675762564        0.0000000000       -1.3259214590
H         -0.4362118830       -0.7612267436       -1.7014971211
H         -0.4362118830        0.7612267436       -1.7014971211
}

hf
mp2
optg
{frequencies,symm=auto
print,low=50}

label1
{hf
start,atden}
{mp2
cphf,1}

surf,start1D=label1,dipole=1,sym=auto
poly,pmp=1,dipole=1,show=1
vscf,type=poly,pmp=2,dipole=1
vmp2,type=poly,pmp=3,dipole=1
```

http://www.molpro.net/info/current/examples/vmp2.com

52 THE COSMO MODEL

The Conductor-like Screening Model (COSMO) (A. Klamt and G. Schüürmann, J. Chem. Soc. Perkin Trans. II 799-805 (1993)) is currently available for HF (`RHF,UHF`) and DFT (`RKS,UKS`) energy calculations and the corresponding gradients.

The COSMO model is invoked by the `COSMO` card:

`COSMO`[,*option*$_1$=*value*$_1$, *option*$_2$=*value*$_2$,...]

where option can be

`NPPA`	size of the underlying basis grid. The value must satisfy: $value = 10 \times 3^k \times 4^l + 2$ (default = 1082; type integer).
`NSPA`	number of segments for non hydrogen atoms. The value must satisfy: $values = 10 \times 3^k \times 4^l + 2$ (default = 92; type integer).
`CAVITY`	the intersection seams of the molecular surface are closed (1) or open (0) (default = 1; type integer).
`EPSILON`	dielectric permittivity (default = -1.d0, which means $\varepsilon = \infty$; type real)
`DISEX`	distance criteria for the **A**-matrix setup. Short range interactions (segment centre distances ¡ DISEX $\times$ mean atomic diameter) are calculated using the underlying basis grid. Long range interactions are calculated via the segment centres (default = 10.d0; type float).
`ROUTF`	factor used for outer cavity construction. The radii of the outer cavity are defined as: $r_i^{out} = r_i + ROUTF \times RSOLV$ (default = 0.85d0; type float)
`PHSRAN`	phase offset of coordinate randomization (default = 0.d0; type float)
`AMPRAN`	amplitude factor of coordinate randomization (default = 1.0d-5; type float)
`RSOLV`	additional radius for cavity construction (default = -1d0, the optimized H radius is used; type float).
`MAXNPS`	maximal number of surface segments (default = -1, will be estimated; type integer).

It is recommended to change the default values for problematic cases only.

By default the program uses optimized radii if existent and 1.17×vdW radius else. The optimized radii [Å] are: H=1.30, C=2.00, N=1.83, O=1.72, F=1.72, S=2.16, Cl=2.05, Br=2.16, I=2.32. Own proposals can be given directly subsequent to the cosmo card:

`RAD`,*symbol, radius*

where the *radius* has to be given in Å.

Example:
```
cosmo
rad,O,1.72
rad,H,1.3
```

Output file:

The COSMO output file will be written after every converged SCF calculation. The segment charges and potentials are corrected by the outlying charge correction. For the total charges and energies corrected and uncorrected values are given. The normal output file contains uncorrected values only. It is recommended to use the corrected values from the output file.

Optimizations:
It is recommended to use optimizer that operates with gradients exclusively. Line search techniques that use energies tends to fail, because of the energy discontinuities which may occur due to a reorganization of the segments after a geometry step. For the same reasons numerical gradients are not recommended.

52.1 BASIC THEORY

COSMO is a continuum solvation model, in which the solvent is represented as a dielectric continuum of permittivity ε. The solute molecule is placed in a cavity inside the continuum. The response of the continuum due to the charge distribution of the solute is described by the generation of a screening charge distribution on the cavity surface. This charge distribution can be calculated by solving the boundary equation of vanishing electrostatic potential on the surface of a conductor. After a discretization of the cavity surface into sufficiently small segments, the vector of the screening charges on the surface segments is

$$\mathbf{q}^* = -\mathbf{A}^{-1}\Phi$$

where Φ is the vector of the potential due to the solute charge distribution on the segments, and $\mathbf{A}$ is the interaction matrix of the screening charges on the segments. This solution is exact for an electric conductor. For finite dielectrics the true dielectric screening charges can be approximated very well by scaling the charge density of a conductor with $f(\varepsilon)$.

$$\mathbf{q} = f(\varepsilon)\mathbf{q}^*; \quad f(\varepsilon) = (\varepsilon - 1)/(\varepsilon + 0.5)$$

In every SCF step the screening charges $\mathbf{q}$ have to be generated from the potential Φ, and then added to the Hamiltonian as external point charges. The total energy of the system is

$$E_{tot} = E_0 + E_{diel}; \quad E_{diel} = \frac{1}{2}\Phi\mathbf{q}$$

where E_0 is the bare self-energy of the system and E_{diel} the dielectric energy.

Cavity construction:
First a surface of mutually excluding spheres of radius $R_i + rsolv$ is constructed, where the R_i are the radii of the atoms, defined as element specific radii and $rsolv$ is some radius representing a typical maximum curvature of a solvent molecular surface. $rsolv$ should not be misinterpreted as a mean solvent radius, nor modified for different solvents. Every atomic sphere is represented by an underlying basis grid of `nppa` points per full atom. Basis grid points which intersect a sphere of a different atom are neglected. In a second step the remainder of the basis grid points are projected to the surface defined by the radii R_i. As a third step of the cavity construction the remaining basis grid points are gathered to segments, which are the areas of constant screening charges in the numerical solution. Finally, the intersection seams between the atoms are filled with additional segments.
Now the A-matrix can be set up. The matrix elements will be calculated from the basis grid points of the segments for close and medium segment distances (governed by the `disex` value),

or using the segment centres for large segment distances.

Outlying charge correction:
The non vanishing electron density outside the cavity causes an error that can be corrected by the outlying charge correction. This correction uses the potential on the so called outer surface (defined by the radii $R_i + \text{rsolv} \times \text{routf}$) to estimate a correction term for the screening charges and the energies (A. Klamt and V. Jonas, J. Chem. Phys., 105, 9972-9981(1996)). The correction will be performed once at the end of a converged SCF calculation. All corrected values can be found in the COSMO output file.

53 QM/MM INTERFACES

The MOLPRO program package can be used in combination with other software to perform hybrid Quantum Mechanics/Molecular Mechanics (QM/MM) calculations. Through the use of point charges, electrostatic embedding can be used for both energy and gradient runs. In particular, lattices of point charges can be included in an external file, gradients with respect to charge positions can be computed, as described in section 10.5. Gradients with respect to QM nuclear positions can be computed (and include the effect of the MM charges) as usual using the FORCE command (section 41).

Although MOLPRO itself does not offer any interface to force field programs, the coupling is supplied by other commercial and non-commercial software. The following is a list of QM/MM software which allow the use of MOLPRO.

53.1 Chemshell

The Chemshell computational chemistry environment (http://www.chemshell.org) offers an interface to many well known force field software (CHARMM, GROMOS, GULP,...). The program sports several geometry optimization algorithms; a molecular dynamics driver for NVE, NVT and NPT ensembles; Monte Carlo; and many other utilities.

The Chemshell Manual can be found at the following website:

http://www.cse.scitech.ac.uk/ccg/software/chemshell/manual/

Instructions on the use of Molpro are available therein. Also concerning the Chemshell environment, a free Graphical User Interface (GUI) has been released. The CCP1GUI facilitates the input for hybrid calculations, allows visualisation of molecular structures and includes molecule editing tools:

http://www.cse.scitech.ac.uk/ccg/software/ccp1gui/

54 THE TDHF AND TDKS PROGRAMS

Real-time electronic dynamics using time-dependent Hartree-Fock and time-dependent Kohn-Sham theories can be performed using the commands `TDHF` and `TDKS` respectively, which have to be preceded by a `HF` and `KS` command. Unrestriced versions are available through `TDUHF` and `TDUKS` and should be preceded by `UHF` and `UKS` runs respectively. All methods require symmetry to be switched off. For details on the theory and methods see H. Eshuis, G. G. Balint-Kurti and F. R. Manby, *J. Chem. Phys.* **128**, 114113 (2008), and references therein. The commands take several options:

```
command,t=,dt=,ns=,ng=,grsize=,print=;
PULSE,options
```

The total propagation time (in au) is set by *t*; *dt* sets the timestep and *ns* the number of steps, where two of the three have to be provided. *ng* sets the number of grid points in one dimension (default = 0) and *grsize* the grid size in bohr (default = 10 bohr). Setting $ng > 2$ switches on the calculation of quantum currents (see below). The option *print* determines the level of output (0=normal output, 1=object linear in matrices, 2=matrices as well, > 2 debug). The subcommand PULSE determines the envelope used, and takes several options depending on the envelope selected. Possibilities are:

NONE: no pulse, no options

STEP, e_x, e_y, e_z, length
a DC-field of strength e_q is applied in the qth direction for a total time *length* (in au)

DC, e_x, e_y, e_z, α
a DC-field of strength e_q is applied in the qth direction. The field is switched on exponentially with a rate determined by α.

TRAP, $e_x, e_y, e_z, \omega, \alpha$
an oscillating field of strength e_q is applied in the qth direction. The field oscillates with angular frequency ω. The envelope reaches e_q in one period of the field, stays constant for α periods and then decays to zero in one period.

CW, e_x, e_y, e_z, ω
an oscillating field of strength e_q is applied in the qth direction. The field oscillates with angular frequency ω. No envelope present.

CWSIN, $e_x, e_y, e_z, \omega, \alpha$
an oscillating field of strength e_q is applied in the qth direction. The field oscillates with angular frequency ω and is switched on using a $\sin^2$ envelope, where α determines how fast the field is switched on.

CWGAUSS, $e_x, e_y, e_z, \omega, \alpha$
like CWSIN, but with a Gaussian envelope

The finite pulses TRAP and STEP produce an absorption spectrum obtained from the time-dependent dipole moment sampled after the field is switched off. It is located in *($input).spec*, or in *molpro.spec* when running interactively, and contains of 4 columns, which contain energy (eV) and the absorption in the x, y, z direction respectively.

All runs produce a file *($input).dat* (or *molpro.dat*) which contains time-dependent properties, like the components of the field, components of the dipole moment, total energy, orbital occupation numbers, orbital energies and total number of electrons. The Molpro output file specifies the order of the data in the file. For very long runs the size of the file is restricted by printing only every twentieth set of data.

In case of an unrestricted run three files are produced: *($input).dat, ($input)a.dat, ($input)b.dat*. The first file contains the generic data about the field, total energy, dipole moments and the expectation value of the total spin operator. The other two files contain the orbital energies and occupation numbers for the α and β electrons respectively.

TDKS and TDUKS use the options provided by the KS and UKS comands. At the moment TDKS/TDUKS suffers from numerical instabilities when using strong fields. Divergence of the energy is observed, possibly due to the use of quadrature for the evaluation of the potential.

Quantum currents will be calculated when choosing $ng > 2$. A cubic grid will be computed of size *grsize* with a total of ng^3 gridpoints. The imaginary part of the density will be summed at every timestep and after the dynamics the total current will be evaluated at every gridpoint. The user can extract the required data from this array by printing out parts of it, or by integrating over point, but this requires actual coding, as this has not been implemented sufficiently neat. It is also straightforward to evaluate the currents at every timestep or at selected timesteps. This is not done automatically, because it slows down the dynamics considerably.

55 ORBITAL MERGING

Orbitals can be manipulated using the `MERGE` facility. For instance, this allows the construction of molecular orbitals from atomic orbitals, to merge and orthogonalize different orbital sets, or to perform 2×2 rotations between individual orbitals. Other orbital manipulations can be performed using the `LOCALI` program (see section 18) or the `MATROP` program (section 56).

The merge program is called using

`MERGE` [,*namout.file*]

All subcommands described in the following sections may be abbreviated by three characters. *namout.file* specifies the output data set (see also `SAVE` command). If *namout.file* is omitted and no `SAVE` card is present, the new orbitals are not saved. All output orbitals must be supplied via `ORBITAL` and `ADD`, `MOVE`, `EXTRA`, or `PROJECT` directives before they can be saved.

55.1 Defining the input orbitals (`ORBITAL`)

`ORBITAL`,*namin.file*,*specifications*

Reads an input orbital set from a dump record. *specifications* can be used to select specific orbital sets, as described in section 4.11. Subsets of these orbitals can be added to the output set by the `ADD`, `MOVE`, or `EXTRA` commands.

55.2 Moving orbitals to the output set (`MOVE`)

`MOVE`,*orb1.sym1,orb2.sym2,orb3.sym3,ioff,fac,istart,iend*

Moves orbitals *orb1.sym1* to *orb2.sym2* from the input set to the first vector of symmetry *sym3* in the output set which is undefined so far. The first *orb3-1* vectors in the output set are skipped regardless of whether they have been defined before or not. If *sym2* > *sym1*, *sym3* will run from *sym1* to *sym2* and the input for *sym3* has no effect. If *orb1.sym1* is negative, abs(*orb1*) is the maximum number of orbitals to be moved, starting with orbital *1.sym1*, up to *orb2.sym2*. If *orb2.sym2* is negative, abs(*orb2*) is the maximum number of vectors to be moved, starting at *orb1.isym1* up to the last orbital in symmetry *sym2*.

Orbitals from the input set which have already been moved or added to the output set are generally skipped. If *orb1* and *orb2* are zero, the whole input set is moved to the output set. In this case the input and output dimensions must be identical. If *orb1* is nonzero but *orb2* is zero, *orb2* is set to the last orbital in symmetry *sym2*. If *sym2=0*, *sym2* is set to *sym1*. *ioff* is an offset in the output vector, relative to the global offset set by `OFFSET` directive. *fac* has no effect for move. The elements *istart* to *iend* of the input vector are moved. If *istart=0* and *iend=0*, the whole input vector is moved.

The usage of the `MOVE` directive is most easily understood by looking at the examples given below. See also `ADD` and `EXTRA` commands.

55.3 Adding orbitals to the output set (`ADD`)

`ADD`,*orb1.sym1,orb2.sym2,orb3.sym3,ioff,fac,istart,iend*

This adds orbitals *orb1.sym1* to *orb2.sym2* to the output vectors, starting at *orb3.sym3*. The input vectors are scaled by the factor *fac*. If *fac=0*, *fac* is set to 1.0. For other details see

MOVE command. Note, however, that the output vectors which have already been defined are not skipped as for MOVE.

See also MOVE and EXTRA commands.

55.4 Defining extra symmetries (EXTRA)

EXTRA,*exsym,orb1.sym1,orb2.sym2,orb3.sym3,ioff,fac,istart,iend*

Works exactly as MOVE, but only input vectors with extra symmetry *exsym* are considered. If *orb1.sym1* and *orb2.sym2* are zero, all input vectors are moved to the output set ordered according to increasing extra symmetries.

Examples:

`EXTRA,1,-4.1`	will move the next 4 orbitals in symmetry 1 which have extra symmetry 1. Orbitals which have been moved before are skipped.
`EXTRA,2,1.1`	will move all orbitals of symmetry 1 which have extra symmetry 2. Orbitals which have been moved before are skipped.
`EXTRA`	will move all orbitals (all symmetries) and order them according to extra symmetries.
`EXTRA,3,1.1,0.8`	Will move all orbitals which have extra symmetry 3 in all symmetries. Orbitals which have been moved before are skipped.

See also ADD and MOVE commands.

55.5 Defining offsets in the output set (OFFSET)

OFFSET,iof_1,iof_2,...,iof_8;

Sets offsets in the output vector for symmetries 1 to 8. In subsequent MOVE or ADD commands, the input vectors are moved to the locations iof_i+1 in the output vectors. The offset for individual ADD or MOVE commands can be modified by the parameter *ioff* on these cards. This card should immediately follow the orbital directive to which it applies. Generally, this card is only needed if the dimensions of input and output vectors are not identical.

If the dimensions of the input orbital sets are smaller than the current basis dimension, the offsets are determined automatically in the following way: each time an orbital set is read in, the previous input orbital dimensions are added to the offsets. Hence, this works correctly if the orbital sets are given in the correct order and if the individual dimensions add up to the current total dimension. If this is not the case, the offsets should be specified on an OFFSET card which must follow the orbital directive.

55.6 Projecting orbitals (PROJECT)

PROJECT,*namin.file*

This command will read vectors from record *namin.file*. These vectors must have the same dimension as those of the current calculation. All orbitals defined so far by the ORBITAL, MOVE, and ADD directives are projected out of the input set. The projected orbitals are then orthonormalized and moved to the undefined output vectors. This should always yield a complete set of vectors.

55.7 Symmetric orthonormalization (ORTH)

ORTH,$n_1, n_2, \ldots, n_8$

Symmetrically orthonormalizes the first n_i vectors in each symmetry i. These vectors must be supplied before by ORBITAL and MOVE or ADD directives.

55.8 Schmidt orthonormalization (SCHMIDT)

SCHMIDT,$n_1, n_2, \ldots, n_8$

Schmidt orthonormalizes the first n_i vectors in each symmetry i. These vectors must be supplied before by ORBITAL and MOVE or ADD directives.

55.9 Rotating orbitals (ROTATE)

ROTATE,*iorb1.sym,iorb2,angle*

Will perform 2×2 rotation of orbitals *iorb1* and *iorb2* in symmetry *sym* by the specified *angle* (in degree). *angle=0* means to swap the orbitals (equivalent to *angle=90*) These vectors must be supplied before by ORBITAL and MOVE or ADD directives.

55.10 Initialization of a new output set (INIT)

INIT,*namout.file*

Will initialize a new output set. All previous vectors in the output set are lost unless they have been saved by a SAVE directive!

55.11 Saving the merged orbitals

SAVE,*namout.file*

Saves the current output set to record *namout.file*. The current output set must be complete and will be Schmidt orthonormalized before it is saved. If the SAVE directive is not supplied, the output vectors will be saved after all valid commands have been processed to the record specified on the MERGE card.

55.12 Printing options (PRINT)

PRINT,*iprint,ideb*

Specifies print options.

$iprint = 0$	no print
$iprint \geq 1$:	orthonormalized orbitals specified on ORTH card are printed.
$iprint \geq 2$:	orbitals are also printed before this orthonormalization.
$iprint \geq 3$:	all final vectors are printed.
$ideb \neq 0$:	the overlap matrices are printed at various stages.

55.13 Examples

55.13.1 H_2F

This example merges the orbitals of H_2 and F

```
***,example for merge
print,orbitals,basis
rh2=1.4
rhf=300.
basis=vdz
symmetry,x,y                             !use C2v symmetry
geometry={F}

text,F
{rhf;wf,9,1,1;occ,3,1,1;orbital,2130.2} !rhf for f-atom

text,H2
symmetry,x,y                             !use C2v symmetry
geometry={
         H1,
         H2,H1,rh2}

{hf;orbital,2100.2}                      !scf for h2
{multi;occ,2;orbital,2101.2}             !mcscf for h2

text,FH2
geometry={F;                             !linear geometry for F+H2
         H1,F,rhf
         H2,H1,rh2,F,180}

{merge
orbital,2130.2                           !rhf orbitals for F-atom
move,1.1,2.1,1.1                         !move orbitals 1.1, 2.1
move,3.1,0.4,4.1;                        !move all remaining, starting at 4.1
orbital,2100.2                           !hf orbitals for H2
move,1.1,0.4                             !move these to free positions
save,2131.2}                             !save merged orbitals

{rhf;occ,4,1,1;start,2131.2              !rhf for F+H2
orbital,2132.2}

{merge
orbital,2130.2                           !rhf orbitals for F-atom
move,1.1,2.1,1.1                         !move orbitals 1.1, 2.1
move,3.1,3.1,4.1;                        !move orbital 3.1 to 4.1
move,4.1,0.4,6.1                         !move all remaining, starting at 6.1
orbital,2101.2                           !mcscf orbitals for H2
move,1.1,0.4                             !move these to free positions
save,2141.2}                             !save merged orbitals

{multi;occ,5,1,1;start,2141.2}           !casscf for F+H2 using valence space
```

http://www.molpro.net/info/current/examples/h2f_merge.com

55.13.2 NO

This example merges the SCF orbitals of N and O to get a full valence space for NO. In the simplest case the atomic calculations are performed in the individual separate basis sets, but using the same symmetry (C_{2v}) as the molecular calculation.

```
***,NO merge
r=2.1

symmetry,x,y
geometry={n}          !N-atom, c2v symmetry

{rhf;occ,3,1,1;       !rhf nitrogen
wf,7,4,3;             !4S state
orbital,2110.2}       !save orbitals to record 2110 on file 2

symmetry,x,y
geometry={o}

{rhf;occ,3,1,1;       !rhf for oxygen
wf,8,4,2              !3P state
orbital,2120.2}       !save orbitals to record 2120 on file 2

geometry={n;o,n,r}    ! NO molecule, c2v symmetry

{MERGE
ORBITAL,2110.2        ! read orbitals of N atom
MOVE,1.1,1.1          ! move 1s orbital to output vector 1.1
MOVE,2.1,2.1,3.1      ! move 2s orbital to output vector 3.1
MOVE,3.1,3.1,5.1      ! move 2pz orbital to output vector 5.1
MOVE,1.2,1.2          ! move 2px orbital to output vector 1.2
MOVE,1.3,1.3          ! move 2py orbital to output vector 1.3
MOVE,4.1,,7.1         ! move virtual orbitals of symmetry 1
MOVE,2.2,,3.2         ! move virtual orbitals of symmetry 2
MOVE,2.3,,3.3         ! move virtual orbitals of symmetry 2
MOVE,1.4              ! move virtual orbitals of symmetry 2
ORBITAL,2120.2        ! read orbitals of O atom
MOVE,1.1,0.4          ! move all oxygen orbitals into place
ROT,3.1,4.1,45;       ! rotate 2s orbitals to make bonding and antibonding
                      ! linear combinations
ROT,5.1,6.1,-45;      ! rotate 2pz orbitals to make bonding and antibonding
                      ! linear combinations
PRINT,1               ! set print option
ORTH,6,2,2            ! symmetrically orthonormalize the valence orbitals
                      ! the resulting orbitals are printed
save,2150.2}          ! save merged orbitals to record 2150.2

{multi;occ,6,2,2      ! perform full valence casscf for NO
wf,15,2,1             ! 2Pix state
wf,15,3,1             ! 2Piy state
start,2150.2}         ! start with merged orbitals
```

http://www.molpro.net/info/current/examples/no_merge1.com

One can also do the atomic calculations in the total basis set, using dummy cards. In this case the procedure is more complicated, since the union of the two orbital spaces is over-complete. The calculation can be done as follows:

a) SCF for the total molecule, orbitals saved to 2100.2

b) SCF for the N atom with dummy basis on the O atom, orbitals saved on 2110.2

c) SCF for the O atom with dummy basis on the N atom, orbitals saved on 2120.2

d) Merge the atomic SCF orbitals. Finally, obtain the virtual orbitals by projecting the merge orbitals out of the SCF orbitals for NO.

```
***,NO merge
geometry={n;o,n,r}
r=2.1

{rhf;occ,5,2,1      !rhf for NO
wf,15,2,1           !2Pi state
orbital,2100.2}     !save orbitals to record 2100 on file 2

dummy,o             !oxygen is dummy
{rhf;occ,3,1,1;      !rhf nitrogen
wf,7,4,3;           !4S state
orbital,2110.2}     !save orbitals to record 2110 on file 2

dummy,n             !nitrogen is dummy
{rhf;occ,3,1,1;     !rhf for oxygen
wf,8,4,2            !3P state
orbital,2120.2}     !save orbitals to record 2120 on file 2

dummy               ! remove dummies
{MERGE               !call merge program

ORBITAL,2110.2      ! read orbitals of N atom
MOVE,1.1,1.1        ! move input vector 1.1 to output vector 1.1
MOVE,2.1,3.1,3.1    ! move input vectors 2.1,3.1 to output vectors
                    ! 3.1 and 4.1
MOVE,1.2,1.2        ! move input vector 1.2  to output vector 1.2
MOVE,1.3,1.3        ! move input vector 1.3  to output vector 1.3
ORBITAL,2120.2      ! read orbitals of O atom
MOVE,1.1,3.1        ! move input vectors 1.1 to 3.1 to output vectors
                    ! 2.1, 5.1, 6.1
MOVE,1.2,1.2        ! move input vector 1.2 to output vector 2.2
MOVE,1.3,1.3        ! move input vector 1.3 to output vector 2.3
ROT,3.1,5.1,45;     ! rotate 2s orbitals to make bonding and antibonding
                    ! linear combinations
ROT,4.1,6.1,-45;    ! rotate 2pz orbitals to make bonding and antibonding
                    ! linear combinations
PRINT,1             ! set print option
ORTH,6,2,2          ! symmetrically orthonormalize the valence orbitals
                    ! the resulting orbitals are printed
PROJ,2100.2         ! Project valence orbitals out of scf orbitals of the
                    ! molecule and add virtual orbital set.
SAVE,2150.2         ! save merged orbitals to record 2150 on file 2
}

{multi;occ,6,2,2    ! perform full valence casscf for NO
wf,15,2,1           ! 2Pi state
wf,15,3,1           ! 2Pi state
start,2150.2}       ! start with merged orbitals
```

http://www.molpro.net/info/current/examples/no_merge2.com

56 MATRIX OPERATIONS

MATROP;

MATROP performs simple matrix manipulations for matrices whose dimensions are those of the one particle basis set. To do so, first required matrices are loaded into memory using the LOAD command. To each matrix an internal *name* (an arbitrary user defined string) is assigned, by which it is referenced in further commands. After performing operations, the resulting matrices can be saved to a dump record using the SAVE directive. Numbers, e.g. traces or individual matrix elements, can be saved in variables.

code may be one of the following:

`LOAD`	Loads a matrix from a file
`SAVE`	Saves a matrix to a file
`ADD`	Adds matrices
`TRACE`	Forms the trace of a matrix or of the product of two matrices
`MULT`	Multiplies two matrices
`TRAN`	Transforms a matrix
`DMO`	Transforms density into MO basis
`NATORB`	Computes natural orbitals
`DIAG`	Diagonalizes a matrix
`OPRD`	Forms an outer product of two vectors
`DENS`	Forms a closed-shell density matrix
`FOCK`	Computes a closed-shell fock matrix
`COUL`	Computes a coulomb operator
`EXCH`	Computes an exchange operator
`PRINT`	Prints a matrix
`PRID`	Prints diagonal elements of a matrix
`PRIO`	Prints orbitals
`ELEM`	Assigns a matrix element to a variable
`READ`	Reads a square matrix from input
`WRITE`	Writes a square matrix to a file
`SET`	Assigns a value to a variable
`ADDVEC`	Adds a multiple of a column of one matrix to a column of a second matrix

Note that the file name appearing in above commands is converted to lower case on unix machines.

See the following subsections for explanations.

56.1 Calling the matrix facility (`MATROP`)

The program is called by the input card `MATROP` without further specifications.

```
MATROP
```

It can be followed by the following commands in any order, with the restriction that a maximum of 50 matrices can be handled. The first entry in each command line is a command keyword, followed by the name of the result matrix. If the specified result matrix *result* already exists, it is overwritten, otherwise a new matrix is created. All matrices needed in the operations must must have been loaded or defined before, unless otherwise stated.

If a backquote (‘) is appended to a name, the matrix is transposed.

56.2 Loading matrices (LOAD)

All matrices which are needed in any of the subsequent commands must first be loaded into memory using the LOAD command. Depending on the matrix type, the LOAD command has slightly different options. In all forms of LOAD *name* is an arbitrary string (up to 16 characters long) by which the loaded matrix is denoted in subsequent commands.

56.2.1 Loading orbitals

LOAD,*name*,ORB [,*record*] [,*specifications*]

loads an orbital coefficient matrix from the given dump record. If the record is not specified, the last dump record is used. Specific orbitals sets can be selected using the optional *specifications*, as explained in section 4.11. The keyword ORB needs not to be given if *name*=ORB.

56.2.2 Loading density matrices

LOAD,*name*,DEN [,*record*] [,*specifications*]

loads a density matrix from the given dump record. If the record is not given, the last dump record is used. Specific orbitals sets can be selected using the optional *specifications*, as explained in section 4.11. The keyword DEN needs not to be given if *name*=DEN.

56.2.3 Loading the AO overlap matrix S

LOAD,*name*,S

loads the overlap matrix in the AO basis. The keyword S needs not to be given if *name*=S.

56.2.4 Loading $\mathbf{S}^{-1/2}$

LOAD,*name*,SMH

loads $\mathbf{S}^{-1/2}$, where **S** is the overlap matrix in the AO basis. The keyword SMH needs not to be given if *name*=SMH.

56.2.5 Loading the one-electron hamiltonian

LOAD,*name*,H0

LOAD,*name*,H01

loads the one-electron hamiltonian in the AO basis. H01 differs from H0 by the addition of perturbations, if present (see sections 34.5.1, 34.5.2). The keyword H0 (H01) needs not to be given if *name*=H0 (H01). The nuclear energy associated to H0 or H01 is internally stored.

56.2.6 Loading the kinetic or potential energy operators

LOAD,*name*,EKIN

LOAD,*name*,EPOT

loads the individual parts of the one-electron hamiltonian in the AO basis. EPOT is summed for all atoms. The nuclear energy is associated to EPOT and internally stored. The keyword EKIN (EPOT) needs not to be given if *name*=EKIN (EPOT).

56.2.7 Loading one-electron property operators

LOAD,*name*,OPER,*opname*,[*isym*],*x,y,z*

loads one-electron operator *opname*, where *opname* is a keyword specifying the operator (a component must be given). See section 6.13 for valid keys. *isym* is the total symmetry of the operator (default 1), and *x,y,z* is the origin of the operator. If the operator is not available yet in the operator record, it is automatically computed. The nuclear value is associated internally to *name* and also stored in variable OPNUC (this variable is overwritten for each operator which is loaded, but can be copied to another variable using the SET command. Note that the electronic part of dipole and quadrupole operators are multiplied by -1.

56.2.8 Loading matrices from plain records

LOAD,*name*,TRIANG,*record*,[*isym*]
LOAD,*name*,SQUARE,*record*,[*isym*]

Loads a triangular or square matrix from a plain record (not a dump record or operator record). If *isym* is not given, 1 is assumed.

56.3 Saving matrices (SAVE)

SAVE,*name*,*record* [,*type*]

At present, *type* can be DENSITY, ORBITALS, FOCK, H0, ORBEN, OPER, TRIANG, SQUARE, or VECTOR. If type is not given but known from LOAD or another command, this is assumed. Orbitals, density matrices, fock matrices, and orbital energies are saved to a dump record (the same one should normally be used for all these quantities). If *type* is H0, the one-electron hamiltonian is overwritten by the current matrix and the nuclear energy is modified according to the value associated to *name*. The nuclear energy is also stored in the variable ENUC. All other matrices can be saved in triangular or square form to plain records using the TRIANG and SQUARE options, respectively (for triangular storage, the matrix is symmetrized before being stored). Eigenvectors can be saved in plain records using the VECTOR option. Only one matrix or vector can be stored in each plain record.

One-electron operators can be stored in the operator record using

SAVE,*name*,OPER, [PARITY=*np*], [NUC=*opnuc*], CENTRE=*icen*],[COORD=[*x,y,z*]]

The user-defined operator *name* can can then be used on subsequent EXPEC or GEXPEC cards. $np = 1, 0, -1$ for symmetric, square, antisymmetric operators, respectively (default 1). If CENTRE is specified, the operator is assumed to have its origin at the given centre, where *icen* refers to the row number of the z-matrix input. The coordinates can also be specified explicitly using COORD. By default, the coordinates of the last read operator are assumed, or otherwise zero.

If `NATURAL` orbitals are generated and saved in a dump record, the occupation numbers are automatically stored as well. This is convenient for later use, e.g., in MOLDEN.

56.4 Adding matrices (`ADD`)

`ADD`,*result* [,*fac1*],*mat1* [,*fac2*],mat2,...

calculates *result* = *fac1* · *mat1* + *fac2* · *mat2* + ...

The strings *result*, *mat1*, *mat2* are internal names specifying the matrices. *mat1*, *mat2* must exist, otherwise an error occurs. If *result* does not exist, it is created.

The factors *fac1*, *fac2* are optional (may be variables). If not given, one is assumed.

The nuclear values associated to the individual matrices are added accordingly and the result is associated to *result*.

56.5 Trace of a matrix or the product of two matrices (`TRACE`)

`TRACE`,*variable, mat1*,,[*factor*]

Computes *variable* = *factor**tr(*mat1*).

`TRACE`,*variable, mat1, mat2*,[*factor*]

Computes *variable* = *factor**trace(*mat1* · *mat2*).

The result of the trace operation is stored in the MOLPRO variable *variable*, which can be used in subsequent operations.

If *factor* is not given, one is assumed.

56.6 Setting variables (`SET`)

`SET`,*variable*,*value*

Assigns *value* to MOLPRO variable *variable*, where *value* can be an expression involving any number of variables or numbers. Indexing of *variable* is not possible, however.

56.7 Multiplying matrices (`MULT`)

`MULT`,*result, mat1, mat2*,[*fac1*],[*fac2*]

calculates *result* = *fac2* * *result* + *fac1* * *mat1* · *mat2*

The strings *result* , *mat1* , *mat2* are the internal names of the matrices. If *fac1* is not given, *fac1=1* is assumed. If *fac2* is not given, *fac2=0* is assumed. If a backquote (‘) is appended to *mat1* or *mat2* the corresponding matrix is transposed before the operation. If a backquote is appended to *result*, the resulting matrix is transposed.

56.8 Transforming operators (TRAN)

TRAN,*result*, *Op*, *C*

calculates *result* = *C(T)*Op*C*. The strings *result*, *C*, and *Op* are the internal names of the matrices. If a backquote (‘) is appended to *C* or *Op* the corresponding matrix is transposed before the operation. Thus,

TRAN,*result*, *Op*, *C‘*

computes *result* = *C*Op*C(T)*.

56.9 Transforming density matrices into the MO basis (DMO)

DMO,*result*, *D*, *C*

calculates *result* = *C(T)*S*D*S*C*. The strings *result*, *C*, and *D* are internal names.

56.10 Diagonalizing a matrix DIAG

DIAG,*eigvec*,*eigval*,*matrix* [,*iprint*]

Diagonalizes *matrix*. The eigenvectors and eigenvalues are stored internally with associated names *eigvec* and *eigval*, respectively (arbitrary strings of up to 16 characters). The if *iprint*.gt.0, the eigenvalues are printed. If *iprint.gt.1*, also the eigenvectors are printed.

56.11 Generating natural orbitals (NATORB)

NATORB,*name*,*dens*,*thresh*

computes natural orbitals for density matrix *dens*. Orbitals with occupation numbers greater or equal to *thresh* (default 1.d-4) are printed.

56.12 Forming an outer product of two vectors (OPRD)

OPRD,*result*,*matrix*,*orb1*,*orb2*,*factor*

Takes the column vectors *v1* and *v2* from *matrix* and adds their outer product to *result*. *v1* and *v2* must be given in the form *icol.isym*, e.g., 3.2 means the third vector in symmetry 2. The result is

$$result(a,b) = result(a,b) + factor * v1(a) * v2(b)$$

If *result* has not been used before, it is zeroed before performing the operation.

56.13 Combining matrix columns (ADDVEC)

ADDVEC,*result*,*orbr*,*source*,*orbs*,*factor*

Takes the column vector *orbs* from *source* and adds it to columnn *orbr* of *result*. *v1* and *v2* must be given in the form *icol.isym*, e.g., 3.2 means the third vector in symmetry 2.

56.14 Forming a closed-shell density matrix (DENS)

DENS,*density*,*orbitals*,$iocc_1$, $iocc_2$ ...

Forms a closed-shell density matrix *density* from the given *orbitals*. The number of occupied orbitals in each symmetry *i* must be provided in $iocc_i$.

56.15 Computing a fock matrix (FOCK)

FOCK,*f*,*d*

computes a closed shell fock matrix using density *d*. The result is stored in *f*.

56.16 Computing a coulomb operator (COUL)

COUL,*J*,*d*

computes a coulomb operator J(d) using density *d*.

56.17 Computing an exchange operator (EXCH)

EXCH,*K*,*d*

computes an exchange operator K(d) using density *d*.

56.18 Printing matrices (PRINT)

PRINT,*name*,[*ncol(1), ncol(2),*...]

prints matrix *name*. *ncol(isym)* is the number of columns to be printed for row symmetry *isym* (if not given, all columns are printed). For printing orbitals one can also use ORB.

56.19 Printing diagonal elements of a matrix (PRID)

PRID,*name* prints the diagonal elements of matrix *name*.

56.20 Printing orbitals (PRIO)

PRIO,*name*,$n_1, n_2, n_3, \ldots, n_8$

prints orbitals *name*. The first n_i orbitals are printed in symmetry *i*. If $n_i = 0$, all orbitals of that symmetry are printed.

56.21 Assigning matrix elements to a variable (ELEM)

ELEM,*name*,*matrix*, *col*,*row*

assigns elements (col,row) of *matrix* to variable *name*. *col* and *row* must be given in the form *number.isym*, where *number* is the row or column number in symmetry *isym*. The product of the row and column symmetries must agree with the matrix symmetry.

56.22 Reading a matrix from the input file (`READ`)

`READ`,*name*,[[TYPE=]*type*],[[SUBTYPE=]*subtype*],[[SYM=]*symmetry*], [FILE=*file*]
{ *values* }

Reads a square matrix (symmetry 1) from input or an ASCII file. The *values* can be in free format, but their total number must be correct. Comment lines starting with '#', '*', or '!' are skipped. If the data are given in input, the data block must be enclosed either by curley brackets or the first linbe must be `BEGIN_DATA` and the last line `END_DATA`. If a *filename* is specified as option, the data are read from this file. In this case, the `BEGIN_DATA`, `END_DATA` lines in the file are optional, and no data block must follow.

For compatibility with older versions, the data can also be included in the input using the `INCLUDE` command (see section 3.1). In this case, the include file must contain the `BEGIN_DATA` and `END_DATA` lines (this is autopmatically the case if the file has been written using the `MATROP,WRITE` directive).

type is a string which can be used to assign a matrix type. If appropriate, this should be any of the ones used in the `LOAD` command. In addition, `SUBTYPE` can be specified if necessary. This describes, e.g., the type of orbitals or density matrices (e.g., for natural orbitals *TYPE*=`ORB` and *SUBTYPE*=`NATURAL`). The matrix symmetry needs to be given only if it is not equal to 1.

56.23 Writing a matrix to an ASCII file (`WRITE`)

`WRITE`,*name*,[*filename* [*status*]]

Writes a matrix to an ASCII file. If *filename* is not given the matrix is written to the output file, otherwise to the specified file (*filename* is converted to lower case). If *filename*=`PUNCH` it is written to the current punch file.

If *status*=`NEW`, `ERASE` or `em REWIND`, a new file is written, otherwise as existing file is appended.

56.24 Examples

The following example shows various uses of the `MATROP` commands.

```
***,h2o matrop examples
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                                !bond length
theta=104                              !bond angle
hf                                     !do scf calculation
{multi
natorb
canonical}
{matrop
load,D_ao,DEN,2140.2                   !load mcscf density matrix
load,Cnat,ORB,2140.2,natural           !load mcscf natural orbitals
load,Ccan,ORB,2140.2,canonical         !load mcscf canonical orbitals
load,Dscf,DEN,2100.2                   !load scf density matrix
load,S                                 !load overlap matrix

prio,Cnat,4,1,2                        !prints occupied casscf orbitals

elem,d11,Dscf,1.1,1.1                  !print element D(1,1)
elem,d21,Dscf,2.1,1.1                  !print element D(2,1)
elem,d12,Dscf,1.1,2.1                  !print element D(1,2)

tran,S_mo,s,Cnat                       !transform s into MO basis (same as above)
print,S_mo                             !print result - should be unit matrix

trace,Nao,S_mo                         !trace of S_MO = number of basis functions
trace,Nel,D_ao,S                       !form trace(DS) = number of electrons

mult,SC,S,Cnat                         !form SC=S*Cnat
tran,D_nat,D_ao,SC                     !transform density to natural MO (could also be done usi

prid,D_nat                             !print diagonal elements (occupation numbers)

dmo,D_can,D_ao,Ccan                    !transform D_ao to canonical MO basis. Same as above sim
add,D_neg,-1,D_can                     !multiply d_can by -1
diag,U,EIG,D_neg                       !diagonalizes density D_can
mult,Cnat1,Ccan,U                      !transforms canonical orbitals to natural orbitals
prio,Cnat1,4,1,2                       !prints new natural orbitals

natorb,Cnat2,D_ao                      !make natural orbitals using MCSCF density D_ao directly
prio,Cnat2,4,1,2                       !prints new natural orbitals (should be the same as abov

add,diffden,D_ao,-1,Dscf               !form mcscf-scf difference density
natorb,C_diff,diffden                  !make natural orbitals for difference density

write,diffden,denfile                  !write difference density to ASCII file denfile
save,C_diff,2500.2                     !store natural orbitals for difference density in dump r
}
```

http://www.molpro.net/info/current/examples/matrop.com

This second example adds a quadrupole field to `H0`. The result is exactly the same as using the `QUAD` command. `H0` is overwritten by the modified one-electron matrix, and the nuclear energy is automatically changed appropriately. The subsequent SCF calculations use the modified one-electron operator.

Note that it is usually recommended to add fields with the DIP, `QUAD`, or `FIELD` commands.

```
memory,2,m
R    =    0.96488518 ANG
THETA=  101.90140469
geometry={H1
          O,H1,R;
          H2,O,R,H1,THETA}
{hf;wf,10,1}

field=0.05             !define field strength

{matrop
load,h0,h0             !load one-electron hamiltonian
load,xx,oper,xx        !load second moments
load,yy,oper,yy
load,zz,oper,zz
add,h01,h0,field,zz,-0.5*field,xx,-0.5*field,yy   !add second moments to h0 and store in h01
save,h01,1210.1,h0}    !save h0
hf                     !do scf with modified h0

{matrop
load,h0,h0             !load h0
load,qmzz,oper,qmzz    !load quadrupole moment qmzz
add,h01,h0,field,qmzz  !add quadrupole moment to h0  (same result as above with second moments
save,h01,1210.1,h0}    !save h0
hf                     !do scf with modified h0

quad,,,field           !add quadrupole field to h0
hf                     !do scf with modified h0 (same result as above with matrop)

field,zz,field,xx,-0.5*field,yy,-0.5*field   ! (add general field; same result as above)
hf                     !do scf with modified h0 (same result as above with matrop)

field,zz,field       !same as before with separate field commands
field+,xx,-0.5*field
field+,yy,-0.5*field
hf                     !do scf with modified h0 (same result as above with matrop)
```

http://www.molpro.net/info/current/examples/matropfield.com

56.25 Exercise: SCF program

Write a closed-shell SCF program for H_2O using MATROP!

Hints:

First generate a starting orbital guess by finding the eigenvectors of *h0*. Store the orbitals in a record. Basis and geometry are defined in the usual way before the first call to MATROP.

Then use a MOLPRO DO loop and call MATROP for each iteration. Save the current energy in a variable (note that the nuclear energy is stored in variable ENUC). Also, compute the dipole moment in each iteration. At the end of the iteration perform a convergence test on the energy change using the IF command. This must be done outside MATROP just before the ENDDO. At this stage, you can also store the iteration numbers, energies, and dipole moments in arrays, and print these after reaching convergence using TABLE. For the following geometry and basis set

```
geometry={o;h1,o,r;h2,o,r,h1,theta}   !Z-matrix geometry input
r=1 ang                               !bond length
theta=104                             !bond angle
basis=vdz                             !basis set
thresh=1.d-8                          !convergence threshold
```

the result could look as follows:

```
 SCF has converged in 24 iterations

    ITER        E              DIP
     1.0   -68.92227207    2.17407361
     2.0   -71.31376891   -5.06209922
     3.0   -73.73536433    2.10199751
     4.0   -74.64753557   -1.79658706
     5.0   -75.41652680    1.43669203
     6.0   -75.77903293    0.17616098
     7.0   -75.93094231    1.05644998
     8.0   -75.98812258    0.63401784
     9.0   -76.00939154    0.91637513
    10.0   -76.01708679    0.76319435
    11.0   -76.01988143    0.86107911
    12.0   -76.02088864    0.80513445
    13.0   -76.02125263    0.83990621
    14.0   -76.02138387    0.81956198
    15.0   -76.02143124    0.83202128
    16.0   -76.02144833    0.82464809
    17.0   -76.02145450    0.82912805
    18.0   -76.02145672    0.82646089
    19.0   -76.02145752    0.82807428
    20.0   -76.02145781    0.82711046
    21.0   -76.02145792    0.82769196
    22.0   -76.02145796    0.82734386
    23.0   -76.02145797    0.82755355
    24.0   -76.02145797    0.82742787
```

It does not converge terribly fast, but it works!

A Installation Guide

MOLPRO is distributed to licensees on a self-service basis using the world-wide web. Those entitled to the code should obtain it from https://www.molpro.net/download supplying the username and password given to them. The web pages contain both source code and binaries, although not everyone is entitled to source code, and binaries are not available for every platform.

Execution of MOLPRO, whether a supplied binary or built from source, requires a valid licence key. Note that the key consists of two components, namely a list of comma-separated key=value pairs, and a password string, and these are separated by '&'. In most cases the licence key will be automatically downloaded from the website when building or installing the software.

A.1 Installation of pre-built binaries

Binaries are given as self-extracting tar archives which are installed by running them on the command line. There are binaries tuned for several architectures. These also support parallel execution. The parallel binaries are built using GA with TCGMSG, and the default connection across nodes is rsh. One can use ssh instead of rsh by changing environment variable TCGRSH, e.g.

```
export TCGRSH=/usr/bin/ssh
```

for bash shell. There is a generic serial binary which should run on all IA32 architectures.

The tar archives are fully relocatable, the location can be changed when running the script interactively, the default is `/usr/local`.

If the script finds a licence key which has been cached in `$HOME/.molpro/token` from a previous install then that key will be installed with the software. If the script cannot find a key or automatically download it from the molpro website then the script will prompt that this part of the install has failed. All files of Molpro are installed, but the user must then manually install the key with the library files in a file named `.token`, e.g.: `/usr/local/lib/molpro-`*`mpptype-arch`*`/.token`

Other configuration options as described in section A.2.5 may also be specified in the script file: `/usr/local/bin/molpro`

A.2 Installation from source files

A.2.1 Overview

There are usually four distinct stages in installing MOLPRO from source files:

Configuration A shell script that allows specification of configuration options is run, and creates a configuration file that drives subsequent installation steps.

Compilation The program is compiled and linked, and other miscellaneous utilities and files, including the default options file, are built. The essential resulting components are

1. The `molpro` shell script which launches thge main executable. In serial case one can directly run the main executable.
2. The `molpro.exe` executable, which is the main program. For parallel computation, multiple copies of `molpro.exe` are started by a single instance of `molpro` shell script using the appropriate system utility, e.g. `mpirun`, `parallel`, etc.
3. Machine-ready basis-set, and other utility, libraries.

Validation A suite of self-checking test jobs is run to provide assurance that the code as built will run correctly.

Installation The program can be run directly from the source tree in which it is built, but it is usually recommended to run the procedure that installs the essential components in standard system directories.

A.2.2 Prerequisites

The following are required or strongly recommended for installation from source code.

1. A Fortran 90 compiler. Fortran77-only compilers will not suffice. On HPC systems the latest vendor-supplied compiler should be used. The full list of supported compilers can be found at http://www.molpro.net/supported.

2. GNU *make*, freely available from http://www.fsf.org and mirrors. GNU *make* must be used; most system-standard makes do not work. In order to avoid the use of a wrong *make*, it may be useful to set an alias, e.g., `alias make='gmake -s'`. A recent version of GNU *make* is required, 3.80 or above.

3. About 10GB disk space (strongly system-dependent; more with large-blocksize file systems, and where binary files are large) during compilation. Typically 100Mb is needed for the finally installed program. Large calculations will require larger amounts of disk space.

4. One or more large scratch file systems, each containing a directory that users may write on. There are parts of the program in which demanding I/O is performed simultaneously on two different files, and it is therefore helpful to provide at least two filesystems on different physical disks if other solutions, such as striping, are not available. The directory names should be stored in the environment variables `$TMPDIR`, `$TMPDIR2`, `$TMPDIR3`,.... These variables should be set before the program is installed (preferably in `.profile` or `.cshrc`), since at some stages the installation procedures will check for them (cf. section A.2.5).

5. If the program is to be built for parallel execution then the Global Arrays toolkit or the MPI-2 library is needed. For building MOLPRO with the Global Arrays toolkit, we recommend the latest stable version (although earlier versions may also work). This is available from http://www.emsl.pnl.gov/docs/global and should be installed prior to compiling MOLPRO. As mentioned in Global Arrays documentation, there are three possible ways for building GA: (1) GA with MPI; (2) GA with TCGMSG-MPI; and (3) GA with TCGMSG. Molpro can work with either of these interfaces. For building MOLPRO with the MPI-2 library, we recommend to use the built-in MPI-2 library, which may have advantages of optimization on some platforms. If there is no built-in one on the platform, a fresh MPI-2 library (e.g.: MPICH2, see http://www.mcs.anl.gov/research/projects/mpich2) should be installed prior to compiling MOLPRO. Many MPI-2 libraries, including Intel MPI, Bull MPI, MPICH2, and Open MPI, have been tested, and others untested could also work.

6. The source distribution of MOLPRO, which consists of a compressed tar archive with a file name of the form `molpro.2010.1.tar.gz`. The archive can be unpacked using `gunzip` and `tar`.

A.2.3 Configuration

Once the distribution has been unpacked, change to the `Molpro` directory that has been created. Having changed to the `Molpro` directory, you should check that the directory containing the Fortran compiler you want to use is in your PATH. Then run the command

```
./configure -batch
```

which creates the file `CONFIG`. This file contains machine-dependent parameters, such as compiler options. Normally `CONFIG` will not need changing, but you should at the least examine it, and change any configuration parameters which you deem necessary. Any changes made to `CONFIG` will be lost next time `./configure` is invoked, so it is best to supply as many of these as possible via the command line.

The `configure` procedure may be given command line options, and, if run without `-batch` , additionally prompts for a number of parameters:

1. On certain machines it is possible to compile the program to use either 32 or 64 bit integers, and in this case `configure` may be given a command-line option `-i4` or `-i8` respectively to override the default behaviour. Generally, the 64-bit choice allows larger calculations (files larger than 2Gb, more than 16 active orbitals), but can be slower if the underlying hardware does not support 64-bit integers. Note that if `-i4` is used then large files (greater than 2Gb) are supported on most systems, but even then the sizes of MOLPRO records are restricted to 16 Gb since the internal addressing in MOLPRO uses 32-bit integers. If `-i8` is used, the record and file sizes are effectively unlimited. Normally we recommend using the default determined by configure.

2. In the case of building for parallel execution, the option `-mpp` or `-mppx` must be given on the command line. For the distinction between these two parallelism modes, please refer to the user manual, section 2. The option `-mppbase` must also be given followed by the location of the Global Arrays build directory or the MPI-2 library include directory.

   For the case of using the Global Arrays toolkit, one example can be

   ```
   ./configure -mpp -mppbase /usr/local/ga-[version]
   ```

   If using a Global Arrays build with an MPI library the appropriate MPI executable should appear first in PATH when more than one is available.

   Queries regarding Global Arrays installations should be sent directly to the Global Arrays team, any Molpro related queries will assume a fully functional Global Arrays suite with all internal tests run successfully.

   For the case of using the MPI-2 library, one example can be

   ```
   ./configure -mpp -mppbase /usr/local/mpich2-install/include
   ```

   and the `-mppbase` directory should contain file mpi.h. Please ensure the built-in or freshly built MPI-2 library fully supports MPI-2 standard and works properly.

3. If any system libraries are in unusual places, it may be necessary to specify them explicitly as the arguments to a `-L` command-line option.

4. `configure` asks whether you wish to use system BLAS subroutine libraries. MOLPRO has its own optimised Fortran version of these libraries, and this can safely be used. On most machines, however, it will be advantageous to use a system-tuned version instead. On the command line one can specify the level of BLAS to be used from the system, e.g. `-blas2`. For example if you specify 2, the system libraries will be used for level 2 and level 1 BLAS, but MOLPRO's internal routines will be used for level 3 (i.e., matrix-matrix multiplication). Normally, however, one would choose either 0 or 3, which are the defaults depending upon whether a BLAS library is found.

   A special situation arises if 64-bit integers are in use (-i8), since on many platforms the system BLAS libraries only supports 32-bit integer arguments. In such cases (e.g., IBM, SGI, SUN) either 0 or 4 can be given for the BLAS level. BLAS=0 should always work and means that the MOLPRO Fortran BLAS routines are used. On some platforms (IBM, SGI, SUN) BLAS=4 will give better performance; in this case some 32-bit BLAS routines are used from the system library (these are then called from wrapper routines, which convert 64 to 32-bit integer arguments. Note that this might cause problems if more than 2 GB of memory is used).

   For good performance it is important to use appropriate BLAS libraries; in particular, a fast implementation of the matrix multiplication `dgemm` is very important for MOLPRO. Therefore you should use a system tuned BLAS library whenever available.

MOLPRO will automatically detect the most appropriate BLAS library in many cases. In certain cases, in particular when the BLAS library is installed in a non-default location, configure should be directed to the appropriate directory with:

```
./configure -blaspath /path/to/lib/dir
```

Specification of BLAS libraries can be simplified by placing any relevant downloaded libraries in the directory `blaslibs`; `configure` searches this directory (and then, with lower priority, some potential system directories) for libraries relevant to the hardware.

For Intel and AMD Linux systems we recommend the following BLAS libraries:

MKL — The Intel Math Kernel Library (MKL)

ATLAS — The Automatically Tuned Linear Algebra Software (ATLAS) library. You must use the atlas library specific to your processor:

Pentium III	`Linux_PIIISSE1`
Pentium 4,Xeon	`Linux_P4SSE2`
AMD Athlon	`Linux_ATHLON`
AMD Opteron	`Linux_HAMMER64SSE2_2` (64 bit)

When using atlas MOLPRO will automatically compile in the extra lapack subroutines which do not come by default with the package and so the liblapack.a which comes with Atlas is sufficient.

ACML — For Opteron systems then AMD Core Math Library (ACML) is the preferred blas library.

SGI Altix can use the `scsl` library is preferred. HP platforms can use the `mlib` math library. IBM Power platforms can use the `essl` package.

5. `configure` prompts for the optional bin directory (`INSTBIN`) for linking MOLPRO. This directory should be one normally in the PATH of all users who will access MOLPRO, and its specification will depend on whether the installation is private or public.

6. `configure` prompts for the Molpro installation directory (`PREFIX`).

7. `configure` prompts for the destination directory for documentation. This should normally be a directory that is mounted on a worldwide web server. This is only relevant if the documentation is also going to be installed from this directory (see below).

The full list of command-line options recognized by `configure` are:

`-af90`	use Absoft Pro Fortran compiler
`-aims`	compile aims code
`-auto-ga-hpmpi`	auto-build GA with MPI and HP MPI (experimental)
`-auto-ga-mpich`	auto-build GA with MPI and MPICH1 (experimental)
`-auto-ga-mpich2`	auto-build GA with MPI and MPICH2 (experimental)
`-auto-ga-mvapich2`	auto-build GA with MPI and MVAPICH2 (experimental)
`-auto-ga-mvapich2ib`	auto-build GA with MPI and MVAPICH2 over Infiniband (experimental)
`-auto-ga-openmpi`	auto-build GA with MPI and Open MPI (experimental)

`-auto-ga-tcgmsg`	auto-build GA with TCGMSG (experimental)
`-auto-ga-tcgmsg-hpmpi`	auto-build GA with TCGMSG-MPI and HP MPI (experimental)
`-auto-ga-tcgmsg-mpich`	auto-build GA with TCGMSG-MPI and MPICH1 (experimental)
`-auto-ga-tcgmsg-mpich2`	auto-build GA with TCGMSG-MPI and MPICH2 (experimental)
`-auto-ga-tcgmsg-mvapich2`	auto-build GA with TCGMSG-MPI and MVAPICH2 (experimental)
`-auto-ga-tcgmsg-mvapich2ib`	auto-build GA with TCGMSG-MPI and MVAPICH2 over Infiniband (experimental)
`-auto-ga-tcgmsg-openmpi`	auto-build GA with TCGMSG-MPI and Open MPI (experimental)
`-auto-mpich2`	auto-build MPICH2 (experimental)
`-auto-mvapich2`	auto-build MVAPICH2 (experimental)
`-auto-mvapich2ib`	auto-build MVAPICH2 over Infiniband (experimental)
`-auto-openmpi`	auto-build Open MPI (experimental)
`-batch`	run script non-interactively
`-blas`	use external BLAS library
`-blaspath`	specify blas library path
`-cc`	use C compiler named cc
`-clearspeed`	compile clearspeed code
`-clearspeedbase`	specify clearspeed base path for includes and libraries
`-cuda`	try to get settings for compiling CUDA code
`-f`	specify Molpro fortran pre-processor flags
`-f90`	use f90 Fortran compiler
`-fcc`	use Fujitsu C compiler
`-force-link`	Force linking of main executable
`-fort`	use fort Fortran compiler
`-frt`	use frt Fortran compiler
`-g95`	use G95 Fortran compiler
`-gcc`	use GNU Compiler Collection C compiler
`-gforker`	Use settings for mpich2 configured with gforker option
`-gfortran`	use gfortran Fortran compiler
`-hdf5`	use external HDF5 library
`-hdf5path`	specify HDF5 library path
`-i386`	use settings for i386 machine
`-i4`	Makes default integer variables 4 bytes long
`-i686`	use settings for i686 machine
`-i8`	Makes default integer variables 8 bytes long
`-icc`	use Intel C compiler

`-ifort`	use Intel Fortran compiler
`-inst-pl`	append PL to PREFIX when running make install
`-intel-mpi-lsf`	Use settings for Intel MPI with LSF
`-lapack`	use external LAPACK library
`-lapackpath`	specify LAPACK library path
`-letter`	specify letter latex paper size
`-mpp`	produce parallel Molpro
`-mppbase`	specify mpp base path for includes and libraries
`-mppx`	produce trivial parallel Molpro
`-nagfor`	use NAG Fortran compiler
`-natom`	max number of atoms
`-nbasis`	max number of basis functions
`-noblas`	Don't use external BLAS library
`-noclearspeed`	do not compile clearspeed code
`-nocuda`	don't compile CUDA code
`-nocxx`	do not compile C++ code
`-nohdf5`	don't use external HDF5 library
`-nolapack`	don't use external LAPACK library
`-nolargefiles`	Do not use largefiles
`-noopenmp`	compile without openmp
`-nprim`	max number of primitives
`-nrec`	max number of records
`-nstate`	max number of states per symmetry
`-nsymm`	max number of state symmetries
`-nvalence`	max number of valence orbitals
`-nvcc`	use NVIDIA CUDA C compiler
`-opencc`	use Open64 C compiler
`-openf90`	use Open64 Fortran compiler
`-openmp`	compile with openmp
`-openmpi`	Use settings for standard openmpi
`-openmpi-sge`	Use settings for openmpi compiled with SGE
`-pathcc`	use Pathscale C compiler
`-pathf90`	use Pathscale Fortran compiler
`-pgcc`	use Portland C compiler
`-pgf90`	use Portland Fortran compiler
`-prefix`	Specify top-level installation directory
`-rpm`	Build with RPM settings
`-slater`	compile slater code
`-sm_13`	Use settings for sm_13 architecture for CUDA compilation

`-sm_20`	Use settings for sm_20 architecture for CUDA compilation
`-suncc`	use Sun C compiler
`-sunf90`	use Sun Fortran compiler
`-trial`	use trial parse object
`-x86_64`	use settings for 64-bit x86 machine
`-xlf`	use IBM Fortran compiler

A.2.4 Compilation and linking

After configuration, the remainder of the installation is accomplished using the GNU *make* command. Remember that the default *make* on many systems will not work, and that it is essential to use GNU *make* (cf. section A.2.2). Everything needed to make a functioning program together with all ancillary files is carried out by default simply by issuing the command

```
make
```

in the MOLPRO base directory. Most of the standard options for GNU *make* can be used safely; in particular, `-j` can be used to speed up compilation on a parallel machine. The program can then be accessed by making sure the `bin/` directory is included in the PATH and issuing the command `molpro`. If MPI library is used for building Global Arrays or building MOLPRO directly, please be aware that some MPI libraries use mpd daemons to launch parallel jobs. In this case, mpd daemons must already be running before `make`.

A.2.5 Adjusting the default environment for MOLPRO

The default running options for MOLPRO are stored in the script `bin/molpro`. After program installation, either using binary or from source files, this file should be reviewed and adjusted, if necessary, to make system wide changes.

A.2.6 Tuning

MOLPRO can be tuned for a particular system by running in the root directory the command

```
make tuning
```

This job automatically determines a number of tuning parameters and appends these to the file `bin/molpro`. Using these parameters, MOLPRO will select the best BLAS routines depending on the problem size. This job should run on an empty system. It may typically take 10 minutes, depending on the processor speed, and you should wait for completion of this run before doing the next steps.

A.2.7 Testing

At this stage, it is essential to check that the program has compiled correctly. The makefile target *test* (i.e., command `make test`) will do this using the full suite of test jobs, and although this takes a significantly long time, it should always be done when porting for the first time. A much faster test, which checks the main routes through the program, can be done using `make quicktest`. For parallel installation, it is highly desirable to perform this validation with more than one running process. This can be done conveniently through the `make` command line as, for example,

```
make MOLPRO_OPTIONS=-n2 test
```

If any test jobs fail, the cause must be investigated. It may be helpful in such circumstances to compare the target platform with the lists of platforms on which MOLPRO is thought to function at http://www.molpro.net/supported. If, after due efforts to fix problems of a local origin, the problem cannot be resolved, the developers of MOLPRO would appreciate receiving a report. There is a web-based mechanism at https://www.molpro.net/bugzilla at which as many details as possible should be filled in. It may also be helpful to attach a copy of the CONFIG file along with the failing output. Please note that the purpose of such bug reports is to help the developers improve the code, and not for providing advice on installation or running.

A.2.8 Installing the program for production

Although the program can be used in situ, it is usually convenient to copy only those files needed at run time into appropriate installation directories as specified at configuration time (see section A.2.3) and stored in the file `CONFIG`. To install the program in this way, do

```
make install
```

The complete source tree can then be archived and deleted. The overall effect of this is to create a shell script in the `INSTBIN` directory. The name should relate to the architecture, type of build, integer etc. Symbolic links relating to the type of build are then made, and finally providing that `INSTBIN/molpro` is not a file, a symbolic link is created to the new script. In some cases it is preferable to create a localized script in `INSTBIN/molpro` which will not be over written. The overall effect of this cascade of links is to provide, in the normal case, the commands `molpro` and one or both of `molpros` (serial) and `molprop` (parallel) for normal use, with the long names remaining available for explicit selection of particular variants.

For normal single-variant installations, none of the above has to be worried about, and the `molpro` command will be available from directory `INSTBIN`.

During the install process the key from `$HOME/.molpro/token` is copied to `PREFIX/.token` so that the key will work for all users of the installed version.

A.2.9 Installation of documentation

The documentation is available on the web at http://www.molpro.net/info/users. It is also included with the source code. The PDF user's manual is found in the directory `Molpro/doc/manual.pdf`, with the HTML version in the directory `Molpro/doc/manual/index.html`. After `make install` the documentation is installed in the `doc` subdirectory of `PREFIX` specified in `CONFIG` file generated by the `configure` command. Numerous example input files are included in the manual, and can alternatively be seen in the directory `Molpro/examples`.

A.2.10 Fedora 13 (32-bit)

Installing MOLPRO from source on Fedora 13 Linux can be done very easily using only free software. These notes assume a standard installation of Fedora 13. For all builds one should install the following additional packages via yum:

gcc-c++	provides GNU C and C++ compiler,
gcc-gfortran	provides GNU Fortran compiler,
zlib-devel	provides a library for zipped files.

Optionally one can choose to install:

blas-devel	provides a BLAS library,
lapack-devel	provides a LAPACK library,

which will be used instead of compiling the equivalent MOLPRO routines.

Serial MOLPRO After unpacking MOLPRO one can simply type

```
./configure -batch -gcc -gfortran
make
```

to build MOLPRO.

Parallel MOLPRO **using the Global Arrays with MPI2** The following additional packages should be installed via apt-get:

mpich2	provides MPI-2 library files
mpich2-devel	provides MPI-2 include files

Set up mpd by running:

```
mpd
```

and following the on screen instructions.

The latest stable version of the Global Arrays toolkit should be downloaded. Build and test the Global Arrays toolkit with:

```
make TARGET=LINUX FC=gfortran CC=gcc USE_MPI=1 \
     MPI_LIB=/usr/lib/mpich2/lib \
     MPI_INCLUDE=/usr/include/mpich2-i386 LIBMPI=-lmpich
mpirun ./global/testing/test.x
```

Then configure and build MOLPRO with:

```
./configure -batch -gcc -gfortran \
            -mpp -mppbase /path/to/directory/ga-4-3-3
make
```

A.2.11 openSUSE 11.3 (32 & 64-bit)

Installing MOLPRO from source on openSUSE 11.3 Linux can be done very easily using only free software. These notes assume a standard installation of openSUSE 11.3. For all builds one should install the following additional packages via YaST:

gcc-c++	provides GNU C and C++ compiler,
gcc-fortran	provides GNU Fortran compiler,

make	provides GNU make
zlib-devel	provides a library for zipped files.

Optionally one can choose to install:

blas	provides a BLAS library,
lapack	provides a LAPACK library,

which will be used instead of compiling the equivalent MOLPRO routines.

Serial MOLPRO After unpacking MOLPRO one can simply type

```
./configure -batch -gcc -gfortran
make
```

to build MOLPRO.

Parallel MOLPRO **using the Global Arrays with MPI** The following additional packages should be installed via YaST:

mpich	provides MPI-1 library
mpich-devel	provides MPI-1 library

The latest stable version of the Global Arrays toolkit should be downloaded. Build and test the Global Arrays toolkit for 64-bit machines with:

```
make TARGET=LINUX64 FC=gfortran CC=gcc USE_MPI=1 \
     MPI_LIB=/opt/mpich/ch-p4/lib64 \
     MPI_INCLUDE=/opt/mpich/ch-p4/include LIBMPI=-lmpich
/opt/mpich/ch-p4/bin/mpirun ./global/testing/test.x
```

or on 32-bit machines with:

```
make TARGET=LINUX FC=gfortran CC=gcc USE_MPI=1 \
     MPI_LIB=/opt/mpich/ch-p4/lib \
     MPI_INCLUDE=/opt/mpich/ch-p4/include LIBMPI=-lmpich
/opt/mpich/ch-p4/bin/mpirun ./global/testing/test.x
```

Then configure and build MOLPRO with:

```
./configure -batch -gcc -gfortran \
            -mpp -mppbase /path/to/directory/ga-4-3-3
make
```

A.2.12 Ubuntu 10.04 (32-bit)

Installing MOLPRO from source on Ubuntu 10.04 Linux can be done very easily using only free software. These notes assume a standard installation of Ubuntu 10.04. For all builds one should install the following additional packages via apt-get:

build-essential	provides GNU C++ compiler,
gfortran	provides GNU Fortran compiler,
curl	provides curl for downloading patches
zlib1g-dev	provides a library for zipped files.

Optionally one can choose to install:

libblas-dev	provides a BLAS library,
liblapack-dev	provides a LAPACK library,

which will be used instead of compiling the equivalent MOLPRO routines.

Serial MOLPRO After unpacking MOLPRO one can simply type

```
./configure -batch -gcc -gfortran
make
```

to build MOLPRO.

Parallel MOLPRO **using the Global Arrays with MPI** The following additional packages should be installed via apt-get:

mpich-bin	provides MPI-1 library
libmpich1.0-dev	provides MPI-1 library
openssh-server	provides ssh access to localhost

Set up password-less ssh by running the following commands and not entering a password when prompted::

```
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

This must be done for each user account which will be running MOLPRO.

The latest stable version of the Global Arrays toolkit should be downloaded. Build and test the Global Arrays toolkit with:

```
make TARGET=LINUX FC=gfortran CC=gcc USE_MPI=1 \
     MPI_LIB=/usr/lib/mpich/lib \
     MPI_INCLUDE=/usr/lib/mpich/include LIBMPI=-lmpich
mpirun ./global/testing/test.x
```

Then configure and build MOLPRO with:

```
./configure -batch -gcc -gfortran \
            -mpp -mppbase /path/to/directory/ga-4-3-3
make
```

A.2.13 Installation on a Cygwin system

On a Windows machine Cygwin should be installed. In addition to the default package list one should also install the packages listed in table 15. If undertaking development work table 16

Package	Package Group	Reason
gcc4	Devel	compiling C files
gcc4-fortran	Devel	compiling Fortran files
gcc4-g++	Devel	compiling C++ files
make	Devel	need GNU make
curl	Web	token download

Table 15: Cygwin requirments for user install

contains a list of potentially useful packages.

Package	Package Group	Reason
bison	Devel	bison
gdb	Devel	gdb
git	Devel	git
libxslt	Gnome	xsltproc
openssh	Net	ssh
vim	Editors	vi

Table 16: Cygwin packages for developers

With the above steps, configure can be run and the Molpro built in the normal way.

B Recent Changes

B.1 New features of MOLPRO2010.1

The functionality is essentially the same as in 2009.1, but many bug fixes and small improvements have been added. Please note the following major changes, in particular of the default RI basis sets in explicitly correlated methods as described below.

B.1.1 AIC density fitting integral program

A faster integral program for density fitting, written by Gerald Knizia, has been added. In particular this speeds up the integral evaluation in F12 calculations by up to a factor of about 10 (depending on the basis set). This program is now used by default, but can be disabled by setting

```
dfit,aic=0
```

in the beginning of the input.

B.1.2 Pair specific geminal exponents in explicitly correlated methods

Different exponents for the Slater-type geminals can be used for valence-valence, core-valence, and core-core pairs. See manual for details.

B.1.3 Change of defaults in explicitly correlated methods

For explicitly correlated F12 calculations that use the VnZ-F12 orbital basis sets (OBS), it is now the default to use the corresponding VnZ-F12/OPTRI basis sets to construct the complementary auxiliary orbital basis (CABS). In case that CABS is not used (e.g., in LMP2-F12), the OBS and OPTRI sets are merged automatically. This yields exactly the same results as would be obtained with the CABS approach. In order to use the default RI sets of 2009.1, please specify option RI_BASIS=JKFIT on the command line, or

```
explicit,ri_basis=jkfit
```

For compatibility reasons, it is still the default to use the JKFIT sets as RI basis for the AVnZ orbital basis sets. In order to use the corresponding OPTRI sets (where available) please specify option RI_BASIS=OPTRI.

B.1.4 New basis sets in the Molpro library

A number of new basis sets have been added to the Molpro library since version 2009.1. The references for these sets can be found in the headers of the respective libmol files.

Li, Be, Na, Mg:

a) New official versions of the correlation consistent basis sets for these elements have been added, both non-relativistic and those contracted for Douglas-Kroll relativistic calculations. Specifically these are:

cc-pVnZ (n=D-5) cc-pwCVnZ (n=D-5) aug-cc-pVnZ (n=D-5) aug-cc-pwCVnZ (n=D-5)

and the above with a -DK extension. The older cc-pVnZ basis sets for these elements can still be accessed via the keywords vdz-old, etc.

b) New basis sets, including RI and MP2 auxiliary sets, have been added for F12 explicit correlation calculations:

cc-pVnZ-F12 (n=D-Q) cc-pCVnZ-F12 (n=D-Q)

The optimized CABS auxiliary sets have the same name but with a /OptRI context

For MP2 and CCSD auxiliary sets, the cc-pVnZ/MP2FIT sets of Hättig can be used, but a new cc-pV5Z/MP2FIT set has been added that is optimal for the new cc-pV5Z basis set; new aug-cc-pVnZ/MP2FIT (n=D-5) sets have been added as well. The original cc-pV5Z/MP2FIT sets of Hättig have been renamed v5z-old/mp2fit.

Cu-Zn, Y-Cd, Hf-Hg:

a) While the aug-cc-pVnZ-PP (n=D-5) and cc-pwCVnZ-PP (n=D-5) sets were already available, the combination aug-cc-pwCVnZ-PP was not yet defined. These have now been added for these elements.

b) Triple-zeta DK sets have been included now for all of these elements. Unless otherwise noted, these were optimized for 2nd-order DKH. In the cases of Hf-Hg, sets contracted for 3rd-order DKH are also now included:

cc-pVTZ-DK cc-pwCVTZ-DK aug-cc-pVTZ-DK aug-cc-pwCVTZ-DK

and the above with -DK replaced by -DK3 for DKH3 calculations in the case of Hf-Hg.

H-He, B-Ne, Al-Ar, Ga-Kr: a) A variety of DK contracted basis sets have been added for these elements:

aug-cc-pVnZ-DK (n=D-5) cc-pCVnZ-DK (n=D-5) cc-pwCVnZ-DK (n=D-5) aug-cc-pCVnZ-DK (n=D-5) aug-cc-pwCVnZ-DK (n=D-5)

b) Official cc-pCV6Z and aug-cc-pCV6Z are now also available for Al-Ar

c) For explicitly correlated calculations, the core-valence sets have been added:

cc-pCVnZ-F12 (n=D-Q) for B-Ne, Al-Ar cc-pCVnZ-F12/OptRI (n=D-Q) for B-Ne, Al-Ar

d) cc-pVnZ-F12/OptRI (n=D-Q) as also been added for He

Turbomole def2 basis sets: The complete Turbomole def2 basis set family has been added to the Molpro basis library (for all elements H to Rn, except Lanthanides). The def2-orbital basis sets can now be accessed as SV(P), SVP, TZVP, TZVPP, QZVP and QZVPP. In this nomenclature SVP, TZVPP, and QZVPP correspond to valence double-zeta (VDZ), valence triple-zeta (VTZ) and valence quadruple-zeta (VQZ) basis sets, respectively.

Auxiliary density fitting basis sets for all elements are available as well (e.g., TZVPP/JFIT, TZVPP/JKFIT, TZVPP/MP2FIT) and are chosen automatically in density-fitted calculations. Supposedly, the JKFIT sets are universal and also applicable in combination with the AVnZ basis sets. Initial results indicate that they also work well with the cc-pVnZ-PP and aug-cc-pVnZ-PP series of basis sets.

The orbital basis sets can also be accessed in singly and doubly augmented versions (carrying A or DA prefixes, respectively, e.g., ASVP, DASVP), and the auxiliary fitting sets in singly augmented versions (e.g., ATZVPP/MP2FIT).

The old Turbomole basis sets have been renamed; if required, they can be accessed with a def1-prefix (e..g, def1-SVP, def1-TZVPP, etc.).

B.1.5 Improved support for MPI implementation of parallelism

The *ppidd* harness that manages interprocess communication has been improved. The performance of the implementation based on pure MPI, as an alternative to use of the Global Arrays toolkit, is considerably improved, through the use of dedicated helper processes that service one-sided remote memory accesses.

B.1.6 Change of the order of states and the defaults for computing the Davidson correction in multi-state MRCI

The previous way to compute the Davidson correction in multi-state MRCI could lead to non-continuous cluster corrected energies. This is now avoided by ordering the MRCI eigenstates according to increasing energy (previously they were ordered according to maximum overlap with the reference wavefunctions). Furthermore, additional options for computing the Davidson correction in multi-state calculations are added (for details see manual). The old behavior can be recovered using options SWAP,ROTREF=-1.

B.1.7 IPEA shift for CASPT2

A variant of the IPEA shift of G. Ghigo, B. O. Roos, and P.A. Malmqvist, Chem. Phys. Lett. **396**, 142 (2004) has been added. The implementation is not exactly identical to the one in MOLCAS, since in our program the singly external configurations are not (RS2) or only partially (RS2C) contracted. The shift is invoked by giving option `IPEA=`*shift* on the `RS2` or `RS2C` commands; the recommended value for *shift* is 0.25. For details of the implementation see manual.

B.1.8 Karton-Martin extrapolation of HF energies

The two-point formula for extrapolating the HF reference energy, as proposed by A. Karton and J. M. L. Martin, Theor. Chem. Acc. **115**, 330 (2006) has been added: $E_{\mathrm{HF,n}} = E_{\mathrm{HF,CBS}} + A(n+1)\cdot\exp(-9\sqrt{n})$. Use `METHOD_R=KM` for this.

B.2 New features of MOLPRO2009.1

B.2.1 Basis set updates

Correlation consistent basis sets for Li, Be, Na, and Mg have been updated to their official versions as reported in Prascher et al., Theor. Chem. Acc. (2010). These now also include core-valence, diffuse augmented, and Douglas-Kroll relativistically contracted versions. The previous sets are still available but have been renamed vdz-old, vtz-old, etc.

B.2.2 Explicitly correlated calculations

Due to new findings, the default behavior of the F12 programs was changed in the following points:

1. For open-shell systems the default wave function ansatz for was modified. This affects RMP2-F12 and open-shell CCSD-F12 calculations. The new default generally improves open-shell treatments and leads to more consistent behavior. The previous behavior can be restored by

   ```
   explicit,extgen=0
   ```

   (for more details see manual).

2. The procedure for the construction of complementary auxiliary basis sets (CABS) and the thresholds were changed. This affects all non-local F12 calculations. The previous behavior can be restored by

   ```
   explicit,ortho_cabs=0,thrcabs=1-7,thrcabs_rel=1e-8
   ```

3. In numeric frequency calculations, the freezing of auxiliary basis sets was improved. This can affect calculations where many redundant functions are deleted.

4. Pair energies for the explicitly correlated methods can be printed using

   ```
   print,pairs
   ```

   If inner-shell orbitals are correlated, the cc, cv, and vv contributions to the correlation energies are also printed.

B.2.3 Improvements to the Hartree-Fock program

The atomic density guess in Hartree-Fock has been improved and extended. Guess basis sets are now available for most atoms and for all pseudopotentials. Most pseudopotentials have been linked to the appropriate basis sets, so that it is sufficient to specify, e.g.

```
basis=vtz-pp
```

which will select the correlation consistent triple zeta basis sets and the associated (small core) pseudopotential. Similarly, it is mostly sufficient to specify the basis set for other pseudopotential/basis set combinations.

If the wavefunction symmetry is not given in the Hartree-Fock input and not known from a previous calculation, the HF program attempts to determine it automatically from the aufbau pricniple (previously, symmetry 1 was assumed in all cases). For example,

```
geometry={n};
{hf;wf,spin=3}
```

automatically finds that the wavefunction symmetry is 8.

B.2.4 Changes to geometry input

1. Rationalisation of options for molecular geometry. It is now illegal to specify symmetry and orientation options (eg x;noorient;angstrom) inside a geometry block, which now contains just the geometry specification (Z-matrix or XYZ). Options have to be specified using the new ORIENT and SYMMETRY commands, and/or existing commands such ANGSTROM. This change will, unfortunately, render many inputs incompatible with 2008.1 and earlier versions of Molpro, but has been introduced to allow correct and clean parsing of geometries containing, for example, yttrium atoms, which previously conflicted with the Y symmetry option.

2. Simplification of geometry input. The program now detects automatically whether the geometry is specified as a Z-matrix, or using cartesian coordinates, and so there is no need any more to set the geomtyp variable. The standard XYZ format is still accepted for cartesian coordinates, but the first two lines (number of atoms, and a comment) can be omitted if desired.

B.2.5 MPI-2 parallel implementation

The program now can be built from the source files with the Global Arrays toolkit or the MPI-2 library for parallel execution.

B.3 New features of MOLPRO2008.1

The new features of MOLPRO version 2008.1 include the following.

1. Efficient closed-shell and open-shell MP2-F12 and CCSD(T)-F12 methods which dramatically improve the basis set convergence, as described in J. Chem. Phys. 126, 164102 (2007); ibid. 127, 221106 (2007); ibid. 128, 154103 (2008).

2. Natural bond order (NBO) and natural population analysis (NPA) as described in Mol. Phys. 105, 2753 (2007) and references therein.

3. Correlation regions within a localized molecular orbital approach as described in J. Chem. Phys. 128, 144106 (2008).

4. Automated calculation of anharmonic vibrational frequencies and zero-point energies using VCI methods as described in J. Chem. Phys. 126, 134108 (2007) and references therein.

5. Coupling of DFT and coupled cluster methods as described in Phys. Chem. Chem. Phys. 10, 3353 (2008) and references therein.

6. Enhanced connections to other programs, including graphical display of output and 3-dimensional structures.

7. Support for latest operating systems and compilers, including Mac OS X.

B.4 New features of MOLPRO2006.1

Features and enhancements in MOLPRO version 2006.1 most notably included efficient density fitting methods, explicitly correlated methods, local coupled cluster methods, and several new gradient programs: following:

1. More consistent input language and input pre-checking.

2. More flexible basis input, allowing to handle multiple basis sets

3. New more efficient density functional implementation, additional density functionals.

4. Low-order scaling local coupled cluster methods with perturbative treatment of triples excitations (LCCSD(T) and variants like LQCISD(T))

5. Efficient density fitting (DF) programs for Hartree-Fock (DF-HF), Density functional Kohn-Sham theory (DF-KS), Second-order Møller-Plesset perturbation theory (DF-MP2), as well as for all local methods (DF-LMP2, DF-LMP4, DF-LQCISD(T), DF-LCCSD(T))

6. Analytical QCISD(T) gradients

7. Analytical MRPT2 (CASPT2) and MS-CASPT2 gradients, using state averaged MCSCF reference functions

8. Analytical DF-HF, DF-KS, DF-LMP2, and DF-SCS-LMP2 gradients

9. Explicitly correlated methods with density fitting: DF-MP2-R12/2A', DF-MP2-F12/2A' as well as the local variants DF-LMP2-R12/2*A(loc) and DF-LMP2-F12/2*A(loc).

10. Coupling of multi-reference perturbation theory and configuration interaction (CIPT2)

11. DFT-SAPT

12. Transition moments and transition Hamiltonian between CASSCF and MRCI wavefunctions with different orbitals.

13. A new spin-orbit integral program for generally contracted basis sets.

14. Douglas-Kroll-Hess Hamiltonian up to arbitrary order.

15. Improved procedures for geometry optimization and numerical Hessian calculations, including constrained optimization.
16. Improved facilities to treat large lattices of point charges for QM/MM calculations, including lattice gradients (see section 53).
17. An interface to the MRCC program of M. Kallay, allowing coupled-cluster calculations with arbitrary excitation level.
18. Automatic *embarrassingly parallel* computation of numerical gradients and Hessians (`mppx Version`).
19. Additional parallel codes, e.g. DF-HF, DF-KS, DF-LCCSD(T) (partly, including triples).
20. Additional output formats for tables (XHTML, LaTeX, Maple, Mathematica, Matlab and comma-separated variables), orbitals and basis sets (XML), and an optional well-formed XML output stream with important results marked up.

B.5 New features of MOLPRO2002.6

Relative to version 2002.1, there are the following changes and additions:

1. Support for IA-64 Linux systems (HP and NEC) and HP-UX 11.22 for IA-64 (Itanium2).
2. Support for NEC-SX systems.
3. Support for IBM-power4 systems.
4. Modified handling of Molpro system variables. The `SET` command has changed (see sections 8 and 8.4).
5. The total charge of the molecule can be specified in a variable `CHARGE` or on the `WF` card, see section 4.9.
6. Improved numerical geometry optimization using symmetrical displacement coordinates (see sections 41.2 and 42).
7. Improved numerical frequency calculations using the symmetry (`AUTO` option, see section 43).

B.6 New features of MOLPRO2002

Relative to version 2000.1, there are the following principal changes and additions:

1. Modules direct and local are now included in the base version. This means that integral-direct procedures as described in

   M. Schütz, R. Lindh, and H.-J. Werner, Mol. Phys. 96, 719 (1999),

   linear-scaling local MP2, as described in

   G. Hetzer, P. Pulay, and H.-J. Werner, Chem. Phys. Lett. 290, 143 (1998),
   M. Schütz, G. Hetzer, and H.-J. Werner, J. Chem. Phys. **111**, 5691 (1999),
   G. Hetzer, M. Schütz, H. Stoll, and H.-J. Werner, J. Chem. Phys. **113**, 9443 (2000),

   as well as LMP2 gradients as described in

A. El Azhary, G. Rauhut, P. Pulay, and H.-J. Werner, J. Chem. Phys. **108**, 5185 (1998)

are now available without special license. The linear scaling LCCSD(T) methods as described in

M. Schütz and H.-J. Werner, J. Chem. Phys. **114**, 661 (2001),
M. Schütz and H.-J. Werner, Chem. Phys. Lett. **318**, 370 (2000),
M. Schütz, J. Chem. Phys. **113**, 9986 (2000)

will be made available at a later stage.

2. QCISD gradients as described in Phys. Chem. Chem. Phys. **3**, 4853 (2001) are now available.

3. Additional and more flexible options for computing numerical gradients and performing geometry optimizations.

4. A large number of additional density functionals have been added, together with support for the automated functional implementer described in Comp. Phys. Commun. **136** 310–318 (2001).

5. Multipole moments of arbitrary order can be computed.

6. Further modules have been parallelized, in particular the CCSD(T) and direct LMP2 codes. The parallel running procedures have been improved. The parallel version is available as an optional module.

7. The basis set library has been extended.

8. Some subtle changes in the basis set input: it is not possible any more that several one-line basis input cards with definitions for individual atoms follow each other. Each new basis card supercedes previous ones. Either all specifications must be given on *one* `BASIS` card, or a basis input block must be used. `BASIS,NAME` is now entirely equivalent to `BASIS=NAME`, i.e. a global default basis set is defined and the variable `BASIS` is set in both cases.

9. Pseudopotential energy calculations can now be performed with up to *i*-functions, gradients with up to *h*-functions.

10. Many internal changes have been made to make MOLPRO more modular and stable. Support has been added for recent operating systems on Compaq, HP, SGI, SUN, and Linux. The patching system has been improved.

B.7 Features that were new in MOLPRO2000

Relative to version 98.1, there are the following principal changes and additions:

1. There was a fundamental error in the derivation of the spin-restricted open-shell coupled-cluster equations in J. Chem. Phys. 99, 5129 (1993) that is also reflected in the RCCSD code in MOLPRO version 98.1 and earlier. This error has now been corrected, and an erratum has been published in J. Chem. Phys. **112**, 3106 (2000). Fortunately, the numerical implications of the error were small, and it is not anticipated that any computed properties will have been significantly in error.

2. There was a programming error in the transformation of gradients from Cartesian to internal coordinates, which in some cases resulted in slow convergence of geometry optimizations. The error is now fixed.

3. Vibrational frequencies formerly by default used average atomic masses, rather than those of the most common isotopes, which is now the default behaviour.

4. MCSCF second derivatives (author Riccardo Tarroni) added (preliminary version, only without symmetry). Frequency and geometry optimization programs are modified so that they can use the analytic Hessian.

5. New internally contracted multi-reference second-order perturbation theory code (author Paolo Celani) through command `RS2C`, as described in P. Celani and H.-J. Werner, J. Chem. Phys. **112**, 5546 (2000).

6. EOM-CCSD for excited states (author Tatiana Korona).

7. QCISD dipole moments as true analytical energy derivatives (author Guntram Rauhut).

8. Linear scaling (CPU and memory) LMP2 as described by G. Hetzer, P. Pulay, and H.-J. Werner, Chem. Phys. Lett. 290, 143 (1998).
   M. Schütz, G. Hetzer, and H.-J. Werner, J. Chem. Phys. **111**, 5691 (1999).

9. Improved handling of basis and geometry records. 98.1 and 99.1 dump files can be restarted, but in case of problems with restarting old files, add `RESTART,NOGEOM` immediately after the `file` card. Also, if there are unjustified messages coming up in very large cases about "ORBITALS CORRESPOND TO DIFFERENT GEOMETRY" try `ORBITAL`,*record*,`NOCHECK`. (This can happen for cases with more than 100 atoms, since the old version was limited to 100).

10. Reorganization and generalization of basis input. Increased basis library.

11. Counterpoise geometry optimizations.

12. Improved running procedures for MPP machines. Parallel direct scf and scf gradients are working. These features are only available with the MPP module, which is not yet being distributed.

13. Important bugfixes for DFT grids, CCSD with paging, finite field calculations without core orbitals, spin-orbit coupling.

14. Many other internal changes.

As an additional service to the MOLPRO community, an electronic mailing list has been set up to provide a forum for open discussion on all aspects of installing and using MOLPRO. The mailing list is intended as the primary means of disseminating hints and tips on how to use Molpro effectively. It is not a means of raising queries directly with the authors of the program. For clearly demonstrable program errors, reports should continue to be sent to molpro@molpro.net; however, 'how-to' questions sent there will merely be redirected to this mailing list.

In order to subscribe to the list, send mail to molpro-user-request@molpro.net containing the text `subscribe`; for help, send mail containing the text `help`.

Messages can be sent to the list (molpro-user@molpro.net), but this can be done only by subscribers. Previous postings can be viewed in the archive at http://www.molpro.net/molpro-user/archive irrespective of whether or not you subscribe to the list. Experienced Molpro users are encouraged to post responses to queries raised. Please do contribute to make this resource mutually useful.

B.8 Facilities that were new in MOLPRO98

MOLPRO98 has the full functionality of MOLPRO96, but in order to make the code more modular and easier to use and maintain, a number of structural changes have been made. In particular, the number of different records has been significantly reduced. The information for a given wavefunction type, like orbitals, density matrices, fock matrices, occupation numbers and other information, is now stored in a single dump record. Even different orbital types, e.g., canonical, natural, or localized orbitals, are stored in the same record, and the user can subsequently access individual sets by keywords on the `ORBITAL` directive. New facilities allow the use of starting orbitals computed with different basis sets and/or different symmetries for SCF or MCSCF calculations. The default starting guess for SCF calculations has been much improved, which is most useful in calculations for large molecules. The use of special procedures for computing non-adiabatic couplings or diabatization of orbitals has been significantly simplified. We hope that these changes make the program easier to use and reduce the probability of input errors. However, in order to use the new facilities efficiently, even experienced MOLPRO users should read the sections *RECORDS* and *SELECTING ORBITALS AND DENSITY MATRICES* in the manual. It is likely that standard MOLPRO96 inputs still work, but changes may be required in more special cases involving particular records for orbitals, density matrices, or operators.

All one-electron operators needed to compute expectation values and transition quantities are now stored in a single record. Operators for which expectation values are requested can be selected globally for all programs of a given run using the global `GEXPEC` directive, or for a specific program using the `EXPEC` directive. All operators are computed automatically when needed, and the user does not have to give input for this any more. See section *ONE-ELECTRON OPERATORES AND EXPECTATION VALUES* of the manual for details.

Due to the changed structure of dump and operator records, the utility program `MATROP` has a new input syntax. MOLPRO96 inputs for `MATROP` do not work any more.

In addition to these organizational changes, a number of new programs have been added. Analytic energy gradients can now be evaluated for MP2 and DFT wavefunctions, and harmonic vibrational frequencies, intensities, and thermodynamic quantities can be computed automatically using finite differences of analytical gradients. Geometry optimization has been further improved, and new facilities for reaction path following have been added.

An interface to the graphics program MOLDEN has been added, which allows to visualize molecular structures, orbitals, electron densities, or vibrations.

Integral-direct calculations, in which the two-electron integrals in the AO basis are never stored on disk but always recomputed when needed, are now available for all kinds of wavefunctions, with the exception of perturbative triple excitations in MP4 and CCSD(T) calculations. This allows the use of significantly larger basis sets than was possible before. The direct option can be selected globally using the `GDIRECT` command, or for a specific program using the `DIRECT` directive. See section *INTEGRAL DIRECT METHODS* in the manual for details. Note that the `DIRECT` module is optional and not part of the basic MOLPRO distribution.

Local electron correlation methods have been further improved. In combination with the integral-direct modules, which implement efficient prescreening techniques, the scaling of the computational cost with molecular size is dramatically reduced, approaching now quadratic or even linear scaling for MP2 and higher correlation methods. This makes possible to perform correlated calculations for much larger molecules than were previously feasible. However, since these methods are subject of active current research and still under intense development, we decided not to include them in the current MOLPRO release. They will be optionally available in one of the next releases.

C Density functional descriptions

C.1 `B86`: X$\alpha\beta\gamma$

Divergence free semiempirical gradient-corrected exchange energy functional. $\lambda = \gamma$ in ref.

$$g = -\frac{c\,(\rho\,(s))^{4/3}\left(1+\beta\,(\chi\,(s))^2\right)}{1+\lambda\,(\chi\,(s))^2}, \tag{70}$$

$$G = -\frac{c\,(\rho\,(s))^{4/3}\left(1+\beta\,(\chi\,(s))^2\right)}{1+\lambda\,(\chi\,(s))^2}, \tag{71}$$

$$c = 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \tag{72}$$

$$\beta = 0.0076, \tag{73}$$

$$\lambda = 0.004. \tag{74}$$

C.2 `B86MGC`: X$\alpha\beta\gamma$ with Modified Gradient Correction

B86 with modified gradient correction for large density gradients.

$$g = -c\,(\rho\,(s))^{4/3} - \frac{\beta\,(\chi\,(s))^2\,(\rho\,(s))^{4/3}}{(1+\lambda\,(\chi\,(s))^2)^{4/5}}, \tag{75}$$

$$G = -c\,(\rho\,(s))^{4/3} - \frac{\beta\,(\chi\,(s))^2\,(\rho\,(s))^{4/3}}{(1+\lambda\,(\chi\,(s))^2)^{4/5}}, \tag{76}$$

$$c = 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \tag{77}$$

$$\beta = 0.00375, \tag{78}$$

$$\lambda = 0.007. \tag{79}$$

C.3 `B86R`: X$\alpha\beta\gamma$ Re-optimised

Re-optimised β of B86 used in part 3 of Becke's 1997 paper.

$$g = -\frac{c\,(\rho\,(s))^{4/3}\left(1+\beta\,(\chi\,(s))^2\right)}{1+\lambda\,(\chi\,(s))^2}, \tag{80}$$

$$G = -\frac{c\,(\rho\,(s))^{4/3}\left(1+\beta\,(\chi\,(s))^2\right)}{1+\lambda\,(\chi\,(s))^2}, \tag{81}$$

$$c = 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \tag{82}$$

$$\beta = 0.00787, \tag{83}$$

$$\lambda = 0.004. \tag{84}$$

C.4 B88: Becke 1988 Exchange Functional

$$G = -(\rho(s))^{4/3}\left(c + \frac{\beta(\chi(s))^2}{1+6\beta\chi(s)\,arcsinh(\chi(s))}\right), \tag{85}$$

$$g = -(\rho(s))^{4/3}\left(c + \frac{\beta(\chi(s))^2}{1+6\beta\chi(s)\,arcsinh(\chi(s))}\right), \tag{86}$$

$$c = 3/8\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \tag{87}$$

$$\beta = 0.0042. \tag{88}$$

C.5 B88C: Becke 1988 Correlation Functional

Correlation functional depending on B86MGC exchange functional with empirical atomic parameters, t and u. The exchange functional that is used in conjunction with B88C should replace B88MGC here.

$$f = -0.8\rho(a)\rho(b)q^2\left(1 - \frac{\ln(1+q)}{q}\right), \tag{89}$$

$$q = t(x + y), \tag{90}$$

$$x = 0.5\left(c\sqrt[3]{\rho(a)} + \frac{\beta(\chi(a))^2\sqrt[3]{\rho(a)}}{(1+\lambda(\chi(a))^2)^{4/5}}\right)^{-1}, \tag{91}$$

$$y = 0.5\left(c\sqrt[3]{\rho(b)} + \frac{\beta(\chi(b))^2\sqrt[3]{\rho(b)}}{(1+\lambda(\chi(b))^2)^{4/5}}\right)^{-1}, \tag{92}$$

$$t = 0.63, \tag{93}$$

$$g = -0.01\rho(s)dz^4\left(1 - 2\frac{\ln(1+1/2z)}{z}\right), \tag{94}$$

$$z = 2ur, \tag{95}$$

$$r = 0.5\rho(s)\left(c(\rho(s))^{4/3} + \frac{\beta(\chi(s))^2(\rho(s))^{4/3}}{(1+\lambda(\chi(s))^2)^{4/5}}\right)^{-1}, \tag{96}$$

$$u = 0.96, \tag{97}$$

$$d = \tau(s) - 1/4\frac{\sigma(ss)}{\rho(s)}, \tag{98}$$

$$G = -0.01\rho(s)dz^4\left(1 - 2\frac{\ln(1+1/2z)}{z}\right), \tag{99}$$

$$c = 3/8\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \tag{100}$$

$$\beta = 0.00375, \tag{101}$$

$$\lambda = 0.007. \tag{102}$$

C.6 `B95`: Becke 1995 Correlation Functional

tau dependent Dynamical correlation functional.

$$T = [0.031091, 0.015545, 0.016887], \tag{103}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{104}$$

$$V = [7.5957, 14.1189, 10.357], \tag{105}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{106}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{107}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{108}$$

$$P = [1, 1, 1], \tag{109}$$

$$f = \frac{E}{1 + l((\chi(a))^2 + (\chi(b))^2)}, \tag{110}$$

$$g = \frac{F\varepsilon(\rho(s), 0)}{H(1+\nu(\chi(s))^2)^2}, \tag{111}$$

$$G = \frac{F\varepsilon\left(\rho\left(s\right),0\right)}{H\left(1+\nu\left(\chi\left(s\right)\right)^{2}\right)^{2}}, \tag{112}$$

$$\begin{aligned} E = {} & \varepsilon\left(\rho\left(a\right),\rho\left(b\right)\right) \\ & -\varepsilon\left(\rho\left(a\right),0\right) \\ & -\varepsilon\left(\rho\left(b\right),0\right), \end{aligned} \tag{113}$$

$$l = 0.0031, \tag{114}$$

$$\begin{aligned} F = {} & \tau\left(s\right) \\ & -1/4\,\frac{\sigma\left(ss\right)}{\rho\left(s\right)}, \end{aligned} \tag{115}$$

$$H = 3/5\,6^{2/3}\left(\pi^{2}\right)^{2/3}\left(\rho\left(s\right)\right)^{5/3}, \tag{116}$$

$$\nu = 0.038, \tag{117}$$

$$\begin{aligned} & \varepsilon\left(\alpha,\beta\right) \\ & = (\alpha + \beta)\Bigg(e\left(r\left(\alpha,\beta\right),T_1,U_1,V_1,W_1,X_1,Y_1,P_1\right) \\ & \quad - \frac{e\left(r\left(\alpha,\beta\right),T_3,U_3,V_3,W_3,X_3,Y_3,P_3\right)\omega\left(\zeta\left(\alpha,\beta\right)\right)\left(1-\left(\zeta\left(\alpha,\beta\right)\right)^{4}\right)}{c} \\ & \quad + \left(e\left(r\left(\alpha,\beta\right),T_2,U_2,V_2,W_2,X_2,Y_2,P_2\right)\right. \\ & \quad \left. - e\left(r\left(\alpha,\beta\right),T_1,U_1,V_1,W_1,X_1,Y_1,P_1\right)\right)\omega\left(\zeta\left(\alpha,\beta\right)\right)\left(\zeta\left(\alpha,\beta\right)\right)^{4}\Bigg), \end{aligned} \tag{118}$$

$$\begin{aligned} & r\left(\alpha,\beta\right) \\ & = 1/4\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\frac{1}{\pi\left(\alpha+\beta\right)}}, \end{aligned} \tag{119}$$

$$\begin{aligned} & \zeta\left(\alpha,\beta\right) \\ & = \frac{\alpha-\beta}{\alpha+\beta}, \end{aligned} \tag{120}$$

$$\begin{aligned} & \omega\left(z\right) \\ & = \frac{\left(1+z\right)^{4/3}+\left(1-z\right)^{4/3}-2}{2\sqrt[3]{2}-2}, \end{aligned} \tag{121}$$

$$\begin{aligned} & e\left(r,t,u,v,w,x,y,p\right) \\ & = \\ & -2t\left(1 + ur\right)\ln\left(1 + 1/2\,\frac{1}{t\left(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1}\right)}\right), \end{aligned} \tag{122}$$

$$c = 1.709921. \tag{123}$$

C.7 `B97DF`: Density functional part of B97

This functional needs to be mixed with 0.1943*exact exchange.

$$T = [0.031091, 0.015545, 0.016887], \tag{124}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{125}$$

$$V = [7.5957, 14.1189, 10.357], \tag{126}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{127}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{128}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{129}$$

$$P = [1, 1, 1], \tag{130}$$

$$A = [0.9454, 0.7471, -4.5961], \tag{131}$$

$$B = [0.1737, 2.3487, -2.4868], \tag{132}$$

$$C = [0.8094, 0.5073, 0.7481], \tag{133}$$

$$\lambda = [0.006, 0.2, 0.004], \tag{134}$$

$$d = 1/2\,(\chi(a))^2 + 1/2\,(\chi(b))^2, \tag{135}$$

$$f = (\varepsilon(\rho(a), \rho(b)) - \varepsilon(\rho(a), 0) - \varepsilon(\rho(b), 0))\left(A_0 + A_1 \eta(d, \lambda_1) + A_2 (\eta(d, \lambda_1))^2\right), \tag{136}$$

$$\eta(\theta, \mu) = \frac{\mu\,\theta}{1 + \mu\,\theta}, \tag{137}$$

$$g = \varepsilon(\rho(s), 0)\left(B_0 + B_1 \eta\left((\chi(s))^2, \lambda_2\right) + B_2\left(\eta\left((\chi(s))^2, \lambda_2\right)\right)^2\right) - 3/8\,\sqrt[3]{3}\,4^{2/3}\sqrt[3]{\pi^{-1}}\,(\rho(s))^{4/3}\left(C_0 + C_1 \eta\left((\chi(s))^2, \lambda_3\right) + C_2\left(\eta\left((\chi(s))^2, \lambda_3\right)\right)^2\right), \tag{138}$$

$$G = \varepsilon\left(\rho\left(s\right),0\right)\left(B_0 + B_1\eta\left(\left(\chi\left(s\right)\right)^2,\lambda_2\right) + B_2\left(\eta\left(\left(\chi\left(s\right)\right)^2,\lambda_2\right)\right)^2\right) - 3/8\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}\left(\rho\left(s\right)\right)^{4/3}\left(C_0 + C_1\eta\left(\left(\chi\left(s\right)\right)^2,\lambda_3\right) + C_2\left(\eta\left(\left(\chi\left(s\right)\right)^2,\lambda_3\right)\right)^2\right), \tag{139}$$

$$\varepsilon\left(\alpha,\beta\right) = \left(\alpha + \beta\right)\left(e\left(r\left(\alpha,\beta\right),T_1,U_1,V_1,W_1,X_1,Y_1,P_1\right) - \frac{e\left(r\left(\alpha,\beta\right),T_3,U_3,V_3,W_3,X_3,Y_3,P_3\right)\omega\left(\zeta\left(\alpha,\beta\right)\right)\left(1-\left(\zeta\left(\alpha,\beta\right)\right)^4\right)}{c} + \left(e\left(r\left(\alpha,\beta\right),T_2,U_2,V_2,W_2,X_2,Y_2,P_2\right) - e\left(r\left(\alpha,\beta\right),T_1,U_1,V_1,W_1,X_1,Y_1,P_1\right)\right)\omega\left(\zeta\left(\alpha,\beta\right)\right)\left(\zeta\left(\alpha,\beta\right)\right)^4\right), \tag{140}$$

$$r\left(\alpha,\beta\right) = 1/4\sqrt[3]{3}4^{2/3}\sqrt[3]{\frac{1}{\pi\left(\alpha+\beta\right)}}, \tag{141}$$

$$\zeta\left(\alpha,\beta\right) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{142}$$

$$\omega\left(z\right) = \frac{\left(1+z\right)^{4/3}+\left(1-z\right)^{4/3}-2}{2\sqrt[3]{2}-2}, \tag{143}$$

$$e\left(r,t,u,v,w,x,y,p\right) = -2t\left(1 + ur\right)\ln\left(1 + 1/2\,\frac{1}{t\left(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1}\right)}\right), \tag{144}$$

$$c = 1.709921. \tag{145}$$

C.8 `B97RDF`: Density functional part of B97 Re-parameterized by Hamprecht et al

Re-parameterization of the B97 functional in a self-consistent procedure by Hamprecht et al. This functional needs to be mixed with 0.21*exact exchange.

$$T = [0.031091, 0.015545, 0.016887], \tag{146}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{147}$$

$$V = [7.5957, 14.1189, 10.357], \tag{148}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{149}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{150}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{151}$$

$$P = [1, 1, 1], \tag{152}$$

$$A = [0.955689, 0.788552, -5.47869], \tag{153}$$

$$B = [0.0820011, 2.71681, -2.87103], \tag{154}$$

$$C = [0.789518, 0.573805, 0.660975], \tag{155}$$

$$\lambda = [0.006, 0.2, 0.004], \tag{156}$$

$$d = 1/2\,(\chi(a))^2 + 1/2\,(\chi(b))^2, \tag{157}$$

$$f = (\varepsilon(\rho(a), \rho(b)) - \varepsilon(\rho(a), 0) - \varepsilon(\rho(b), 0))\left(A_0 + A_1\eta(d, \lambda_1) + A_2(\eta(d, \lambda_1))^2\right), \tag{158}$$

$$\eta(\theta, \mu) = \frac{\mu\,\theta}{1 + \mu\,\theta}, \tag{159}$$

$$g = \varepsilon(\rho(s), 0)\left(B_0 + B_1\eta\left((\chi(s))^2, \lambda_2\right) + B_2\left(\eta\left((\chi(s))^2, \lambda_2\right)\right)^2\right) - 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}\,(\rho(s))^{4/3}\left(C_0 + C_1\eta\left((\chi(s))^2, \lambda_3\right) + C_2\left(\eta\left((\chi(s))^2, \lambda_3\right)\right)^2\right), \tag{160}$$

$$G = \varepsilon(\rho(s), 0)\left(B_0 + B_1\eta\left((\chi(s))^2, \lambda_2\right) + B_2\left(\eta\left((\chi(s))^2, \lambda_2\right)\right)^2\right) - 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}\,(\rho(s))^{4/3}\left(C_0 + C_1\eta\left((\chi(s))^2, \lambda_3\right) + C_2\left(\eta\left((\chi(s))^2, \lambda_3\right)\right)^2\right), \tag{161}$$

$$\varepsilon(\alpha,\beta) = (\alpha + \beta)\Bigg(e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1) - \frac{e(r(\alpha,\beta),T_3,U_3,V_3,W_3,X_3,Y_3,P_3)\,\omega(\zeta(\alpha,\beta))\left(1-(\zeta(\alpha,\beta))^4\right)}{c} + (e(r(\alpha,\beta),T_2,U_2,V_2,W_2,X_2,Y_2,P_2) - e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1))\,\omega(\zeta(\alpha,\beta))(\zeta(\alpha,\beta))^4\Bigg), \tag{162}$$

$$r(\alpha,\beta) = 1/4\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{163}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{164}$$

$$\omega(z) = \frac{(1+z)^{4/3}+(1-z)^{4/3}-2}{2\sqrt[3]{2}-2}, \tag{165}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1+ur)\ln\left(1+1/2\,\frac{1}{t(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1})}\right), \tag{166}$$

$$c = 1.709921. \tag{167}$$

C.9 `BR`: Becke-Roussel Exchange Functional

A. D. Becke and M. R. Roussel,Phys. Rev. A 39, 3761 (1989)

$$K = \frac{1}{2}\sum_s \rho_s U_s, \tag{168}$$

where

$$U_s = -(1-e^{-x}-xe^{-x}/2)/b, \tag{169}$$

$$b = \frac{x^3e^{-x}}{8\pi\rho_s} \tag{170}$$

and x is defined by the nonlinear equation

$$\frac{xe^{-2x/3}}{x-2} = \frac{2\pi^{2/3}\rho_s^{5/3}}{3Q_s}, \tag{171}$$

where

$$Q_s = (\upsilon_s - 2\gamma D_s)/6, \tag{172}$$

$$D_s = \tau_s - \frac{\sigma_{ss}}{4\rho_s} \tag{173}$$

and

$$\gamma = 1. \tag{174}$$

C.10 `BRUEG`: Becke-Roussel Exchange Functional — Uniform Electron Gas Limit

A. D. Becke and M. R. Roussel,Phys. Rev. A 39, 3761 (1989)

As for `BR` but with $\gamma = 0.8$.

C.11 `BW`: Becke-Wigner Exchange-Correlation Functional

Hybrid exchange-correlation functional comprimising Becke's 1998 exchange and Wigner's spin-polarised correlation functionals.

$$\alpha = -3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \tag{175}$$

$$g = \alpha\,(\rho\,(s))^{4/3} - \frac{\beta\,(\rho\,(s))^{4/3}\,(\chi\,(s))^2}{1+6\beta\,\chi\,(s)\,arcsinh\,(\chi\,(s))}, \tag{176}$$

$$G = \alpha\,(\rho\,(s))^{4/3} - \frac{\beta\,(\rho\,(s))^{4/3}\,(\chi\,(s))^2}{1+6\beta\,\chi\,(s)\,arcsinh\,(\chi\,(s))}, \tag{177}$$

$$f = -4\,c\rho\,(a)\,\rho\,(b)\,\rho^{-1}\left(1 + \frac{d}{\sqrt[3]{\rho}}\right)^{-1}, \tag{178}$$

$$\beta = 0.0042, \tag{179}$$

$$c = 0.04918, \tag{180}$$

$$d = 0.349. \tag{181}$$

C.12 CS1: Colle-Salvetti correlation functional

R. Colle and O. Salvetti, Theor. Chim. Acta 37, 329 (1974); C. Lee, W. Yang and R. G. Parr, Phys. Rev. B 37, 785(1988)

CS1 is formally identical to CS2, except for a reformulation in which the terms involving υ are eliminated by integration by parts. This makes the functional more economical to evaluate. In the limit of exact quadrature, CS1 and CS2 are identical, but small numerical differences appear with finite integration grids.

C.13 CS2: Colle-Salvetti correlation functional

R. Colle and O. Salvetti, Theor. Chim. Acta 37, 329 (1974); C. Lee, W. Yang and R. G. Parr, Phys. Rev. B 37, 785(1988)

CS2 is defined through

$$K = -a\left(\frac{\rho + 2b\rho^{-5/3}\left[\rho_\alpha t_\alpha + \rho_\beta t_\beta - \rho t_W\right]e^{-c\rho^{-1/3}}}{1+d\rho^{-1/3}}\right) \tag{182}$$

where

$$t_\alpha = \frac{\tau_\alpha}{2} - \frac{\upsilon_\alpha}{8} \tag{183}$$

$$t_\beta = \frac{\tau_\beta}{2} - \frac{\upsilon_\beta}{8} \tag{184}$$

$$t_W = \frac{1}{8}\frac{\sigma}{\rho} - \frac{1}{2}\upsilon \tag{185}$$

and the constants are $a = 0.04918, b = 0.132, c = 0.2533, d = 0.349$.

C.14 DIRAC: Slater-Dirac Exchange Energy

Automatically generated Slater-Dirac exchange.

$$g = -c\left(\rho\left(s\right)\right)^{4/3}, \tag{186}$$

$$c = 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}. \tag{187}$$

C.15 ECERF: Short-range LDA correlation functional

Local-density approximation of correlation energy
for short-range interelectronic interaction $\mathrm{erf}(\mu r_{21})/r_{12}$,
S. Paziani, S. Moroni, P. Gori-Giorgi, and G. B. Bachelet, Phys. Rev. B 73, 155111 (2006).

$$\varepsilon_c^{\mathrm{SR}}(r_s,\zeta,\mu) = \varepsilon_c^{\mathrm{PW92}}(r_s,\zeta) - \frac{[\phi_2(\zeta)]^3 Q\left(\frac{\mu\sqrt{r_s}}{\phi_2(\zeta)}\right) + a_1\mu^3 + a_2\mu^4 + a_3\mu^5 + a_4\mu^6 + a_5\mu^8}{(1+b_0^2\mu^2)^4}, \tag{188}$$

where

$$Q(x) = \frac{2\ln(2)-2}{\pi^2}\ln\left(\frac{1+ax+bx^2+cx^3}{1+ax+dx^2}\right), \tag{189}$$

with $a = 5.84605$, $c = 3.91744$, $d = 3.44851$, and $b = d - 3\pi\alpha/(4\ln(2) - 4)$. The parameters $a_i(r_s, \zeta)$ are given by

$$\begin{aligned}
a_1 &= 4b_0^6 C_3 + b_0^8 C_5, \\
a_2 &= 4b_0^6 C_2 + b_0^8 C_4 + 6b_0^4 \varepsilon_c^{\mathrm{PW92}}, \\
a_3 &= b_0^8 C_3, \\
a_4 &= b_0^8 C_2 + 4b_0^6 \varepsilon_c^{\mathrm{PW92}}, \\
a_5 &= b_0^8 \varepsilon_c^{\mathrm{PW92}},
\end{aligned}$$

with

$$\begin{aligned}
C_2 &= -\frac{3(1-\zeta^2)\, g_c(0, r_s, \zeta = 0)}{8 r_s^3} \\
C_3 &= -(1-\zeta^2)\frac{g(0, r_s, \zeta = 0)}{\sqrt{2\pi}\, r_s^3} \\
C_4 &= -\frac{9\, c_4(r_s, \zeta)}{64 r_s^3} \\
C_5 &= -\frac{9\, c_5(r_s, \zeta)}{40\sqrt{2\pi} r_s^3} \\
c_4(r_s, \zeta) &= \left(\frac{1+\zeta}{2}\right)^2 g''\left(0, r_s\left(\frac{2}{1+\zeta}\right)^{1/3}, \zeta = 1\right) + \left(\frac{1-\zeta}{2}\right)^2 \times \\
& \quad g''\left(0, r_s\left(\frac{2}{1-\zeta}\right)^{1/3}, \zeta = 1\right) + (1-\zeta^2) D_2(r_s) - \frac{\phi_8(\zeta)}{5\,\alpha^2 r_s^2} \\
c_5(r_s, \zeta) &= \left(\frac{1+\zeta}{2}\right)^2 g''\left(0, r_s\left(\frac{2}{1+\zeta}\right)^{1/3}, \zeta = 1\right) + \left(\frac{1-\zeta}{2}\right)^2 \times \\
& \quad g''\left(0, r_s\left(\frac{2}{1-\zeta}\right)^{1/3}, \zeta = 1\right) + (1-\zeta^2) D_3(r_s),
\end{aligned} \tag{190}$$

and

$$b_0(r_s) = 0.784949\, r_s \tag{191}$$

$$g''(0, r_s, \zeta = 1) = \frac{2^{5/3}}{5\,\alpha^2 r_s^2} \frac{1 - 0.02267 r_s}{(1 + 0.4319 r_s + 0.04 r_s^2)} \tag{192}$$

$$D_2(r_s) = \frac{e^{-0.547 r_s}}{r_s^2}\left(-0.388 r_s + 0.676 r_s^2\right) \tag{193}$$

$$D_3(r_s) = \frac{e^{-0.31 r_s}}{r_s^3}\left(-4.95 r_s + r_s^2\right). \tag{194}$$

Finally, $\varepsilon_c^{\mathrm{PW92}}(r_s, \zeta)$ is the Perdew-Wang parametrization of the correlation energy of the standard uniform electron gas [J.P. Perdew and Y. Wang, Phys. Rev. B 45, 13244 (1992)], and

$$g(0, r_s, \zeta = 0) = \frac{1}{2}(1 - B r_s + C r_s^2 + D r_s^3 + E r_s^4)\, \mathrm{e}^{-d r_s}, \tag{195}$$

is the on-top pair-distribution function of the standard jellium model [P. Gori-Giorgi and J.P. Perdew, Phys. Rev. B 64, 155102 (2001)], where $B = -0.0207$, $C = 0.08193$, $D = -0.01277$, $E = 0.001859$, $d = 0.7524$. The correlation part of the on-top pair-distribution function is $g_c(0, r_s, \zeta = 0) = g(0, r_s, \zeta = 0) - \frac{1}{2}$.

C.16 EXERF: Short-range LDA correlation functional

Local-density approximation of exchange energy
for short-range interelectronic interaction $\mathrm{erf}(\mu r_{12})/r_{12}$,
A. Savin, in Recent Developments and Applications of Modern Density Functional Theory, edited by J.M. Seminario (Elsevier, Amsterdam, 1996).

$$\varepsilon_x^{\mathrm{SR}}(r_s,\zeta,\mu) = \frac{3}{4\pi}\frac{\phi_4(\zeta)}{\alpha\, r_s} - \frac{1}{2}(1+\zeta)^{4/3} f_x\left(r_s,\mu(1+\zeta)^{-1/3}\right) + \frac{1}{2}(1-\zeta)^{4/3} f_x\left(r_s,\mu(1-\zeta)^{-1/3}\right) \quad (196)$$

with

$$\phi_n(\zeta) = \frac{1}{2}\left[(1+\zeta)^{n/3} + (1-\zeta)^{n/3}\right], \quad (197)$$

$$f_x(r_s,\mu) = -\frac{\mu}{\pi}\left[(2y-4y^3)\,e^{-1/4y^2} - 3y + 4y^3 + \sqrt{\pi}\,\mathrm{erf}\left(\frac{1}{2y}\right)\right], \qquad y = \frac{\mu\,\alpha\, r_s}{2}, \quad (198)$$

and $\alpha = (4/9\pi)^{1/3}$.

C.17 G96: Gill's 1996 Gradient Corrected Exchange Functional

$$\alpha = -3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}, \quad (199)$$

$$g = (\rho(s))^{4/3}\left(\alpha - \frac{1}{137}(\chi(s))^{3/2}\right), \quad (200)$$

$$G = (\rho(s))^{4/3}\left(\alpha - \frac{1}{137}(\chi(s))^{3/2}\right). \quad (201)$$

C.18 HCTH120: Handy least squares fitted functional

$$T = [0.031091, 0.015545, 0.016887], \quad (202)$$

$$U = [0.21370, 0.20548, 0.11125], \quad (203)$$

$$V = [7.5957, 14.1189, 10.357], \quad (204)$$

$$W = [3.5876, 6.1977, 3.6231], \quad (205)$$

$$X = [1.6382, 3.3662, 0.88026], \quad (206)$$

$$Y = [0.49294, 0.62517, 0.49671], \quad (207)$$

$$P = [1, 1, 1], \tag{208}$$

$$A = [0.51473, 6.9298, -24.707, 23.110, -11.323], \tag{209}$$

$$B = [0.48951, -0.2607, 0.4329, -1.9925, 2.4853], \tag{210}$$

$$C = [1.09163, -0.7472, 5.0783, -4.1075, 1.1717], \tag{211}$$

$$\lambda = [0.006, 0.2, 0.004], \tag{212}$$

$$d = 1/2\,(\chi(a))^2 + 1/2\,(\chi(b))^2, \tag{213}$$

$$f = (\varepsilon(\rho(a), \rho(b)) - \varepsilon(\rho(a), 0) - \varepsilon(\rho(b), 0))\big(A_0 + A_1\eta(d, \lambda_1) + A_2(\eta(d, \lambda_1))^2 + A_3(\eta(d, \lambda_1))^3 + A_4(\eta(d, \lambda_1))^4\big), \tag{214}$$

$$\eta(\theta, \mu) = \frac{\mu\,\theta}{1 + \mu\,\theta}, \tag{215}$$

$$g = \varepsilon(\rho(s), 0)\big(B_0 + B_1\eta\big((\chi(s))^2, \lambda_2\big) + B_2\big(\eta\big((\chi(s))^2, \lambda_2\big)\big)^2 + B_3\big(\eta\big((\chi(s))^2, \lambda_2\big)\big)^3 + B_4\big(\eta\big((\chi(s))^2, \lambda_2\big)\big)^4\big) - 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}(\rho(s))^{4/3}\big(C_0 + C_1\eta\big((\chi(s))^2, \lambda_3\big) + C_2\big(\eta\big((\chi(s))^2, \lambda_3\big)\big)^2 + C_3\big(\eta\big((\chi(s))^2, \lambda_3\big)\big)^3 + C_4\big(\eta\big((\chi(s))^2, \lambda_3\big)\big)^4\big), \tag{216}$$

$$\varepsilon(\alpha,\beta) = (\alpha+\beta)\Bigg(e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1) - \frac{e(r(\alpha,\beta),T_3,U_3,V_3,W_3,X_3,Y_3,P_3)\,\omega(\zeta(\alpha,\beta))\left(1-(\zeta(\alpha,\beta))^4\right)}{c} + (e(r(\alpha,\beta),T_2,U_2,V_2,W_2,X_2,Y_2,P_2) - e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1))\,\omega(\zeta(\alpha,\beta))\,(\zeta(\alpha,\beta))^4\Bigg), \tag{217}$$

$$r(\alpha,\beta) = 1/4\sqrt[3]{3}4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{218}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{219}$$

$$\omega(z) = \frac{(1+z)^{4/3}+(1-z)^{4/3}-2}{2\sqrt[3]{2}-2}, \tag{220}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1+ur)\ln\left(1+1/2\frac{1}{t\left(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1}\right)}\right), \tag{221}$$

$$c = 1.709921. \tag{222}$$

C.19 HCTH147: Handy least squares fitted functional

$$T = [0.031091, 0.015545, 0.016887], \tag{223}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{224}$$

$$V = [7.5957, 14.1189, 10.357], \tag{225}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{226}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{227}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{228}$$

$$P = [1, 1, 1], \tag{229}$$

$$A = [0.542352, 7.01464, -28.3822, 35.0329, -20.4284], \tag{230}$$

$$B = [0.562576, -0.0171436, -1.30636, 1.05747, 0.885429], \tag{231}$$

$$C = [1.09025, -0.799194, 5.57212, -5.86760, 3.04544], \tag{232}$$

$$\lambda = [0.006, 0.2, 0.004], \tag{233}$$

$$d = 1/2\,(\chi(a))^2 + 1/2\,(\chi(b))^2, \tag{234}$$

$$f = (\varepsilon(\rho(a),\rho(b)) - \varepsilon(\rho(a),0) - \varepsilon(\rho(b),0))\left(A_0 + A_1\eta(d,\lambda_1) + A_2(\eta(d,\lambda_1))^2 + A_3(\eta(d,\lambda_1))^3 + A_4(\eta(d,\lambda_1))^4\right), \tag{235}$$

$$\eta(\theta,\mu) = \frac{\mu\,\theta}{1+\mu\,\theta}, \tag{236}$$

$$g = \varepsilon(\rho(s),0)\left(B_0 + B_1\eta\left((\chi(s))^2,\lambda_2\right) + B_2\left(\eta\left((\chi(s))^2,\lambda_2\right)\right)^2 + B_3\left(\eta\left((\chi(s))^2,\lambda_2\right)\right)^3 + B_4\left(\eta\left((\chi(s))^2,\lambda_2\right)\right)^4\right) - 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}\,(\rho(s))^{4/3}\left(C_0 + C_1\eta\left((\chi(s))^2,\lambda_3\right) + C_2\left(\eta\left((\chi(s))^2,\lambda_3\right)\right)^2 + C_3\left(\eta\left((\chi(s))^2,\lambda_3\right)\right)^3 + C_4\left(\eta\left((\chi(s))^2,\lambda_3\right)\right)^4\right), \tag{237}$$

$$\varepsilon(\alpha,\beta) = (\alpha + \beta)\Bigg(e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1) + \frac{e(r(\alpha,\beta),T_3,U_3,V_3,W_3,X_3,Y_3,P_3)\,\omega(\zeta(\alpha,\beta))\left(1-(\zeta(\alpha,\beta))^4\right)}{c} + (e(r(\alpha,\beta),T_2,U_2,V_2,W_2,X_2,Y_2,P_2) - e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1))\,\omega(\zeta(\alpha,\beta))\,(\zeta(\alpha,\beta))^4\Bigg), \tag{238}$$

$$r(\alpha,\beta) = 1/4\,\sqrt[3]{3}\,4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{239}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{240}$$

$$\omega(z) = \frac{(1+z)^{4/3}+(1-z)^{4/3}-2}{2\sqrt[3]{2}-2}, \tag{241}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1+ur)\ln\left(1+1/2\,\frac{1}{t(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1})}\right), \tag{242}$$

$$c = 1.709921. \tag{243}$$

C.20 `HCTH93`: Handy least squares fitted functional

$$T = [0.031091, 0.015545, 0.016887], \tag{244}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{245}$$

$$V = [7.5957, 14.1189, 10.357], \tag{246}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{247}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{248}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{249}$$

$$P = [1, 1, 1], \tag{250}$$

$$A = [0.72997, 3.35287, -11.543, 8.08564, -4.47857], \tag{251}$$

$$B = [0.222601, -0.0338622, -0.012517, -0.802496, 1.55396], \tag{252}$$

$$C = [1.0932, -0.744056, 5.5992, -6.78549, 4.49357], \tag{253}$$

$$\lambda = [0.006, 0.2, 0.004], \tag{254}$$

$$d = 1/2\,(\chi(a))^2 + 1/2\,(\chi(b))^2, \tag{255}$$

$$f = (\varepsilon(\rho(a), \rho(b)) - \varepsilon(\rho(a), 0) - \varepsilon(\rho(b), 0))\left(A_0 + A_1 \eta(d, \lambda_1) + A_2 (\eta(d, \lambda_1))^2 + A_3 (\eta(d, \lambda_1))^3 + A_4 (\eta(d, \lambda_1))^4\right), \tag{256}$$

$$\eta(\theta, \mu) = \frac{\mu\,\theta}{1 + \mu\,\theta}, \tag{257}$$

$$g = \varepsilon(\rho(s), 0)\left(B_0 + B_1 \eta\left((\chi(s))^2, \lambda_2\right) + B_2 \left(\eta\left((\chi(s))^2, \lambda_2\right)\right)^2 + B_3 \left(\eta\left((\chi(s))^2, \lambda_2\right)\right)^3 + B_4 \left(\eta\left((\chi(s))^2, \lambda_2\right)\right)^4\right) - 3/8\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\pi^{-1}}\,(\rho(s))^{4/3}\left(C_0 + C_1 \eta\left((\chi(s))^2, \lambda_3\right) + C_2 \left(\eta\left((\chi(s))^2, \lambda_3\right)\right)^2 + C_3 \left(\eta\left((\chi(s))^2, \lambda_3\right)\right)^3 + C_4 \left(\eta\left((\chi(s))^2, \lambda_3\right)\right)^4\right), \tag{258}$$

$$\varepsilon(\alpha, \beta) = (\alpha + \beta)\left(e(r(\alpha, \beta), T_1, U_1, V_1, W_1, X_1, Y_1, P_1) - \frac{e(r(\alpha, \beta), T_3, U_3, V_3, W_3, X_3, Y_3, P_3)\,\omega(\zeta(\alpha, \beta))\left(1 - (\zeta(\alpha, \beta))^4\right)}{c} + (e(r(\alpha, \beta), T_2, U_2, V_2, W_2, X_2, Y_2, P_2) - e(r(\alpha, \beta), T_1, U_1, V_1, W_1, X_1, Y_1, P_1))\,\omega(\zeta(\alpha, \beta))\,(\zeta(\alpha, \beta))^4\right), \tag{259}$$

$$r(\alpha, \beta) = 1/4\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha + \beta)}}, \tag{260}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{261}$$

$$\omega(z) = \frac{(1+z)^{4/3}+(1-z)^{4/3}-2}{2\sqrt[3]{2}-2}, \tag{262}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1+ur)\ln\left(1+1/2\frac{1}{t(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1})}\right), \tag{263}$$

$$c = 1.709921. \tag{264}$$

C.21 `LTA`: Local τ Approximation

LSDA exchange functional with density represented as a function of τ.

$$g = 1/2E(2\tau(s)), \tag{265}$$

$$E(\alpha) = 1/9c5^{4/5}\sqrt[5]{9}\left(\frac{\alpha\sqrt[3]{3}}{(\pi^2)^{2/3}}\right)^{4/5}, \tag{266}$$

$$c = -3/4\sqrt[3]{3}\sqrt[3]{\pi^{-1}}, \tag{267}$$

$$G = 1/2E(2\tau(s)). \tag{268}$$

C.22 `LYP`: Lee, Yang and Parr Correlation Functional

C. Lee, W. Yang and R. G. Parr, Phys. Rev. B 37, 785(1988); B. Miehlich, A. Savin, H. Stoll and H. Preuss, Chem. Phys. Letters 157, 200 (1989)

$$\begin{aligned} K &= 4\frac{A\rho_\alpha\rho_\beta Z}{\rho} + AB\omega\sigma\left(\rho_\alpha\rho_\beta(47-7\delta)/18-2\rho^2/3\right) \\ &+ \sum_s AB\omega\left(\rho_s\rho_{\bar{s}}\left(8\,2^{2/3}e\rho_s^{8/3}-(5/2-\delta/18)\sigma_{ss}-\frac{(\delta-11)\rho_s\sigma_{ss}}{9\rho}\right).\right. \\ &\left.+\left(2\rho^2/3-\rho_s^2\right)\sigma_{\bar{s}\bar{s}}\right), \end{aligned} \tag{269}$$

where

$$\omega = e^{-\frac{c}{\rho^{1/3}}}Z\rho^{-11/3}, \tag{270}$$

$$\delta = \frac{c + dZ}{\rho^{1/3}}, \tag{271}$$

$$B = 0.04918, \tag{272}$$

$$A = 0.132, \tag{273}$$

$$c = 0.2533, \tag{274}$$

$$d = 0.349, \tag{275}$$

$$e = \frac{3}{10}\left(3\pi^2\right)^{2/3} \tag{276}$$

and

$$Z = \left(1 + \frac{d}{\rho^{1/3}}\right)^{-1}. \tag{277}$$

C.23 `MK00`: Exchange Functional for Accurate Virtual Orbital Energies

$$g = -3\,\frac{\pi\,(\rho\,(s))^3}{\tau\,(s) - 1/4\,\upsilon\,(s)}. \tag{278}$$

C.24 `MK00B`: Exchange Functional for Accurate Virtual Orbital Energies

MK00 with gradient correction of the form of B88X but with different empirical parameter.

$$g = -3\,\frac{\pi\,(\rho\,(s))^3}{\tau\,(s) - 1/4\,\upsilon\,(s)} - \frac{\beta\,(\rho\,(s))^{4/3}\,(\chi\,(s))^2}{1 + 6\beta\,\chi\,(s)\,arcsinh\,(\chi\,(s))}, \tag{279}$$

$$\beta = 0.0016, \tag{280}$$

$$G = -3\,\frac{\pi\,(\rho\,(s))^3}{\tau\,(s) - 1/4\,\upsilon\,(s)} - \frac{\beta\,(\rho\,(s))^{4/3}\,(\chi\,(s))^2}{1 + 6\beta\,\chi\,(s)\,arcsinh\,(\chi\,(s))}. \tag{281}$$

C.25 `P86`: .

Gradient correction to VWN.

$$f = \rho\,e + \frac{e^{-\Phi}C\,(r)\,\sigma}{d\rho^{4/3}}, \tag{282}$$

$$r = 1/4\,\sqrt[3]{3}4^{2/3}\sqrt[3]{\frac{1}{\pi\,\rho}}, \tag{283}$$

$$x = \sqrt{r}, \tag{284}$$

$$\zeta = \frac{\rho(a) - \rho(b)}{\rho}, \tag{285}$$

$$k = [0.0310907, 0.01554535, -1/6\,\pi^{-2}], \tag{286}$$

$$l = [-0.10498, -0.325, -0.0047584], \tag{287}$$

$$m = [3.72744, 7.06042, 1.13107], \tag{288}$$

$$n = [12.9352, 18.0578, 13.0045], \tag{289}$$

$$e = \Lambda + \omega y \left(1 + h\zeta^4\right), \tag{290}$$

$$y = \frac{9}{8}(1 + \zeta)^{4/3} + \frac{9}{8}(1 - \zeta)^{4/3} - 9/4, \tag{291}$$

$$h = 4/9\,\frac{\lambda - \Lambda}{\left(\sqrt[3]{2} - 1\right)\omega} - 1, \tag{292}$$

$$\Lambda = q(k_1, l_1, m_1, n_1), \tag{293}$$

$$\lambda = q(k_2, l_2, m_2, n_2), \tag{294}$$

$$\omega = q(k_3, l_3, m_3, n_3), \tag{295}$$

$$q(A, p, c, d) = A\left(\ln\left(\frac{x^2}{X(x, c, d)}\right) + 2c\arctan\left(\frac{Q(c, d)}{2x + c}\right)(Q(c, d))^{-1} - cp\left(\ln\left(\frac{(x - p)^2}{X(x, c, d)}\right) + 2(c + 2p)\arctan\left(\frac{Q(c, d)}{2x + c}\right)(Q(c, d))^{-1}\right)(X(p, c, d))^{-1}\right), \tag{296}$$

$$Q(c, d) = \sqrt{4d - c^2}, \tag{297}$$

$$X(i,c,d) = i^2 + ci + d, \tag{298}$$

$$\Phi = 0.007390075\,\frac{z\sqrt{\sigma}}{C(r)\,\rho^{7/6}}, \tag{299}$$

$$d = \sqrt[3]{2}\sqrt{(1/2+1/2\,\zeta)^{5/3} + (1/2-1/2\,\zeta)^{5/3}}, \tag{300}$$

$$C(r) = 0.001667 + \frac{0.002568+\alpha r+\beta r^2}{1+\xi r+\delta r^2+10000\beta r^3}, \tag{301}$$

$$z = 0.11, \tag{302}$$

$$\alpha = 0.023266, \tag{303}$$

$$\beta = 0.000007389, \tag{304}$$

$$\xi = 8.723, \tag{305}$$

$$\delta = 0.472. \tag{306}$$

C.26 `PBEC`: PBE Correlation Functional

$$f = \rho\,(\varepsilon(\rho(a),\rho(b)) + H(d,\rho(a),\rho(b))), \tag{307}$$

$$G = \rho\,(\varepsilon(\rho(s),0) + C(Q,\rho(s),0)), \tag{308}$$

$$d = 1/12\,\frac{\sqrt{\sigma}3^{5/6}}{u(\rho(a),\rho(b))\sqrt[6]{\pi^{-1}}\rho^{7/6}}, \tag{309}$$

$$u(\alpha,\beta) = 1/2\,(1 + \zeta(\alpha,\beta))^{2/3} + 1/2\,(1 - \zeta(\alpha,\beta))^{2/3}, \tag{310}$$

$$H(d,\alpha,\beta) = 1/2\,(u(\rho(a),\rho(b)))^3\lambda^2\ln\left(1 + 2\,\frac{\iota\,(d^2+A(\alpha,\beta)\,d^4)}{\lambda\,(1+A(\alpha,\beta)\,d^2+(A(\alpha,\beta))^2d^4)}\right)\iota^{-1}, \tag{311}$$

$$A(\alpha,\beta) = 2\iota\lambda^{-1}\left(e^{-2\frac{\iota\varepsilon(\alpha,\beta)}{(u(\rho(a),\rho(b)))^3\lambda^2}} - 1\right)^{-1}, \quad (312)$$

$$\iota = 0.0716, \quad (313)$$

$$\lambda = \nu\kappa, \quad (314)$$

$$\nu = 16\frac{\sqrt[3]{3}\sqrt[3]{\pi^2}}{\pi}, \quad (315)$$

$$\kappa = 0.004235, \quad (316)$$

$$Z = -0.001667, \quad (317)$$

$$\phi(r) = \theta(r) - Z, \quad (318)$$

$$\theta(r) = \frac{1}{1000}\frac{2.568 + \Xi r + \Phi r^2}{1 + \Lambda r + \Upsilon r^2 + 10\Phi r^3}, \quad (319)$$

$$\Xi = 23.266, \quad (320)$$

$$\Phi = 0.007389, \quad (321)$$

$$\Lambda = 8.723, \quad (322)$$

$$\Upsilon = 0.472, \quad (323)$$

$$T = [0.031091, 0.015545, 0.016887], \quad (324)$$

$$U = [0.21370, 0.20548, 0.11125], \quad (325)$$

$$V = [7.5957, 14.1189, 10.357], \quad (326)$$

$$W = [3.5876, 6.1977, 3.6231], \quad (327)$$

$$X = [1.6382, 3.3662, 0.88026], \quad (328)$$

$$Y = [0.49294, 0.62517, 0.49671], \quad (329)$$

$$P = [1, 1, 1], \quad (330)$$

$$\begin{aligned}\varepsilon(\alpha,\beta) &= e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1) \\ &\quad - \frac{e(r(\alpha,\beta),T_3,U_3,V_3,W_3,X_3,Y_3,P_3)\,\omega(\zeta(\alpha,\beta))\left(1-(\zeta(\alpha,\beta))^4\right)}{c} \\ &\quad + (e(r(\alpha,\beta),T_2,U_2,V_2,W_2,X_2,Y_2,P_2) \\ &\quad - e(r(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1))\,\omega(\zeta(\alpha,\beta))\,(\zeta(\alpha,\beta))^4,\end{aligned} \tag{331}$$

$$r(\alpha,\beta) = 1/4\,\sqrt[3]{3}\,4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{332}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{333}$$

$$\omega(z) = \frac{(1+z)^{4/3}+(1-z)^{4/3}-2}{2\sqrt[3]{2}-2}, \tag{334}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1+ur)\ln\left(1+1/2\,\frac{1}{t\left(v\sqrt{r}+wr+xr^{3/2}+yr^{p+1}\right)}\right), \tag{335}$$

$$c = 1.709921, \tag{336}$$

$$C(d,\alpha,\beta) = K(Q,\alpha,\beta) + M(Q,\alpha,\beta), \tag{337}$$

$$M(d,\alpha,\beta) = 0.5\,\nu\,(\phi(r(\alpha,\beta)) - \kappa - 3/7\,Z)\,d^2 e^{-335.9789467\,\frac{3^{2/3}d^2}{\sqrt[3]{\pi^5\rho}}}, \tag{338}$$

$$K(d,\alpha,\beta) = 0.2500000000\,\lambda^2\ln\left(1 + 2\,\frac{\iota\left(d^2+N(\alpha,\beta)\,d^4\right)}{\lambda\,(1+N(\alpha,\beta)\,d^2+(N(\alpha,\beta))^2 d^4)}\right)\iota^{-1}, \tag{339}$$

$$N(\alpha,\beta) = 2\,\iota\,\lambda^{-1}\left(e^{-4\,\frac{\iota\,\varepsilon(\alpha,\beta)}{\lambda^2}} - 1\right)^{-1}, \tag{340}$$

$$Q = 1/12 \frac{\sqrt{\sigma(ss)}\sqrt[3]{2}3^{5/6}}{\sqrt[6]{\pi^{-1}}\rho^{7/6}}. \tag{341}$$

C.27 PBEX: PBE Exchange Functional

$$g = 1/2E(2\rho(s)), \tag{342}$$

$$G = 1/2E(2\rho(s)), \tag{343}$$

$$E(n) = -3/4 \frac{\sqrt[3]{3}\sqrt[3]{\pi^2}n^{4/3}F(S)}{\pi}, \tag{344}$$

$$S = 1/12 \frac{\chi(s)6^{2/3}}{\sqrt[3]{\pi^2}}, \tag{345}$$

$$F(S) = 1 + R - R\left(1 + \frac{\mu S^2}{R}\right)^{-1}, \tag{346}$$

$$R = 0.804, \tag{347}$$

$$\mu = 1/3\,\delta\,\pi^2, \tag{348}$$

$$\delta = 0.066725. \tag{349}$$

C.28 PBEXREV: Revised PBE Exchange Functional

Changes the value of the constant R from the original PBEX functional

$$g = 1/2E(2\rho(s)), \tag{350}$$

$$G = 1/2E(2\rho(s)), \tag{351}$$

$$E(n) = -3/4 \frac{\sqrt[3]{3}\sqrt[3]{\pi^2}n^{4/3}F(S)}{\pi}, \tag{352}$$

$$S = 1/12 \frac{\chi(s)6^{2/3}}{\sqrt[3]{\pi^2}}, \tag{353}$$

$$F(S) = 1 + R - R\left(1 + \frac{\mu S^2}{R}\right)^{-1}, \tag{354}$$

$$R = 1.245, \tag{355}$$

$$\mu = 1/3\,\delta\,\pi^2, \tag{356}$$

$$\delta = 0.066725. \tag{357}$$

C.29 `PW86`: .

GGA Exchange Functional.

$$g = 1/2\,E\,(2\rho\,(s)), \tag{358}$$

$$E(n) = -3/4\,\sqrt[3]{3}\sqrt[3]{\pi^{-1}}n^{4/3}F(S), \tag{359}$$

$$F(S) = \left(1 + 1.296\,S^2 + 14\,S^4 + 0.2\,S^6\right)^{1/15}, \tag{360}$$

$$S = 1/12\,\frac{\chi(s)\,6^{2/3}}{\sqrt[3]{\pi^2}}, \tag{361}$$

$$G = 1/2\,E\,(2\rho\,(s)). \tag{362}$$

C.30 `PW91C`: Perdew-Wang 1991 GGA Correlation Functional

$$f = \rho\,(\varepsilon\,(\rho\,(a), \rho\,(b)) + H\,(d, \rho\,(a), \rho\,(b))), \tag{363}$$

$$G = \rho\,(\varepsilon\,(\rho\,(s), 0) + C\,(Q, \rho\,(s), 0)), \tag{364}$$

$$d = 1/12\,\frac{\sqrt{\sigma}3^{5/6}}{u\,(\rho\,(a), \rho\,(b))\,\sqrt[6]{\pi^{-1}}\rho^{7/6}}, \tag{365}$$

$$u(\alpha,\beta) = 1/2\,(1+\zeta(\alpha,\beta))^{2/3} + 1/2\,(1-\zeta(\alpha,\beta))^{2/3}, \tag{366}$$

$$H(d,\alpha,\beta) = L(d,\alpha,\beta) + J(d,\alpha,\beta), \tag{367}$$

$$L(d,\alpha,\beta) = 1/2\,(u(\rho(a),\rho(b)))^3\lambda^2 \ln\left(1 + 2\,\frac{\iota\left(d^2 + A(\alpha,\beta)\,d^4\right)}{\lambda\,(1 + A(\alpha,\beta)\,d^2 + (A(\alpha,\beta))^2 d^4)}\right)\iota^{-1}, \tag{368}$$

$$J(d,\alpha,\beta) = \nu\,(\phi(r(\alpha,\beta)) - \kappa - 3/7\,Z)\,(u(\rho(a),\rho(b)))^3 d^2 e^{-\frac{400}{3}\frac{(u(\rho(a),\rho(b)))^4 3^{2/3} d^2}{\sqrt[3]{\pi^5\rho}}}, \tag{369}$$

$$A(\alpha,\beta) = 2\,\iota\lambda^{-1}\left(e^{-2\,\frac{\iota\,\varepsilon(\alpha,\beta)}{(u(\rho(a),\rho(b)))^3\lambda^2}} - 1\right)^{-1}, \tag{370}$$

$$\iota = 0.09, \tag{371}$$

$$\lambda = \nu\,\kappa, \tag{372}$$

$$\nu = 16\,\frac{\sqrt[3]{3}\sqrt[3]{\pi^2}}{\pi}, \tag{373}$$

$$\kappa = 0.004235, \tag{374}$$

$$Z = -0.001667, \tag{375}$$

$$\phi(r) = \theta(r) - Z, \tag{376}$$

$$\theta(r) = \frac{1}{1000}\,\frac{2.568 + \Xi\,r + \Phi\,r^2}{1 + \Lambda\,r + \Upsilon\,r^2 + 10\,\Phi\,r^3}, \tag{377}$$

$$\Xi = 23.266, \tag{378}$$

$$\Phi = 0.007389, \tag{379}$$

$$\Lambda = 8.723, \tag{380}$$

$$\Upsilon = 0.472, \tag{381}$$

$$T = [0.031091, 0.015545, 0.016887], \tag{382}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{383}$$

$$V = [7.5957, 14.1189, 10.357], \tag{384}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{385}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{386}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{387}$$

$$P = [1, 1, 1], \tag{388}$$

$$\begin{aligned}\varepsilon(\alpha,\beta) &= e(r(\alpha,\beta), T_1, U_1, V_1, W_1, X_1, Y_1, P_1) \\ &\quad - \frac{e(r(\alpha,\beta), T_3, U_3, V_3, W_3, X_3, Y_3, P_3)\, \omega(\zeta(\alpha,\beta))\left(1 - (\zeta(\alpha,\beta))^4\right)}{c} \\ &\quad + (e(r(\alpha,\beta), T_2, U_2, V_2, W_2, X_2, Y_2, P_2) \\ &\quad - e(r(\alpha,\beta), T_1, U_1, V_1, W_1, X_1, Y_1, P_1))\, \omega(\zeta(\alpha,\beta))\, (\zeta(\alpha,\beta))^4,\end{aligned} \tag{389}$$

$$r(\alpha,\beta) = 1/4\, \sqrt[3]{3}\, 4^{2/3} \sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{390}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{391}$$

$$\omega(z) = \frac{(1+z)^{4/3} + (1-z)^{4/3} - 2}{2\sqrt[3]{2} - 2}, \tag{392}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1 + ur)\ln\left(1 + 1/2\, \frac{1}{t\left(v\sqrt{r} + wr + xr^{3/2} + yr^{p+1}\right)}\right), \tag{393}$$

$$c = 1.709921, \tag{394}$$

$$C(d,\alpha,\beta) = K(Q,\alpha,\beta) + M(Q,\alpha,\beta), \quad (395)$$

$$M(d,\alpha,\beta) = 0.5\,\nu\,(\phi(r(\alpha,\beta)) - \kappa - 3/7\,Z)\,d^2 e^{-335.9789467\,\frac{3^{2/3} d^2}{\sqrt[3]{\pi^5 \rho}}}, \quad (396)$$

$$K(d,\alpha,\beta) = 0.2500000000\,\lambda^2 \ln\left(1 + 2\,\frac{\iota\left(d^2 + N(\alpha,\beta)\,d^4\right)}{\lambda\,(1 + N(\alpha,\beta)\,d^2 + (N(\alpha,\beta))^2 d^4)}\right)\iota^{-1}, \quad (397)$$

$$N(\alpha,\beta) = 2\,\iota\,\lambda^{-1}\left(e^{-4\,\frac{\iota\,\varepsilon(\alpha,\beta)}{\lambda^2}} - 1\right)^{-1}, \quad (398)$$

$$Q = 1/12\,\frac{\sqrt{\sigma(ss)}\,\sqrt[3]{2}\,3^{5/6}}{\sqrt[6]{\pi^{-1}}\,\rho^{7/6}}. \quad (399)$$

C.31 `PW91X`: Perdew-Wang 1991 GGA Exchange Functional

$$g = 1/2\,E(2\rho(s)), \quad (400)$$

$$G = 1/2\,E(2\rho(s)), \quad (401)$$

$$E(n) = -3/4\,\frac{\sqrt[3]{3}\,\sqrt[3]{\pi^2}\,n^{4/3}F(S)}{\pi}, \quad (402)$$

$$S = 1/12\,\frac{\chi(s)\,6^{2/3}}{\sqrt[3]{\pi^2}}, \quad (403)$$

$$F(S) = \frac{1 + 0.19645\,S\,\mathrm{arcsinh}(7.7956\,S) + \left(0.2743 - 0.1508\,e^{-100\,S^2}\right)S^2}{1 + 0.19645\,S\,\mathrm{arcsinh}(7.7956\,S) + 0.004\,S^4}. \quad (404)$$

C.32 PW92C: Perdew-Wang 1992 GGA Correlation Functional

Electron-gas correlation energy.

$$T = [0.031091, 0.015545, 0.016887], \tag{405}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{406}$$

$$V = [7.5957, 14.1189, 10.357], \tag{407}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{408}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{409}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{410}$$

$$P = [1, 1, 1], \tag{411}$$

$$f = \rho\,\varepsilon(\rho(a), \rho(b)), \tag{412}$$

$$\begin{aligned}\varepsilon(\alpha,\beta) &= e(r(\alpha,\beta), T_1, U_1, V_1, W_1, X_1, Y_1, P_1) \\ &\quad - \frac{e(r(\alpha,\beta), T_3, U_3, V_3, W_3, X_3, Y_3, P_3)\,\omega(\zeta(\alpha,\beta))\left(1 - (\zeta(\alpha,\beta))^4\right)}{c} \\ &\quad + (e(r(\alpha,\beta), T_2, U_2, V_2, W_2, X_2, Y_2, P_2) \\ &\quad - e(r(\alpha,\beta), T_1, U_1, V_1, W_1, X_1, Y_1, P_1))\,\omega(\zeta(\alpha,\beta))(\zeta(\alpha,\beta))^4,\end{aligned} \tag{413}$$

$$r(\alpha,\beta) = 1/4\,\sqrt[3]{3}\,4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{414}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{415}$$

$$\omega(z) = \frac{(1+z)^{4/3} + (1-z)^{4/3} - 2}{2\sqrt[3]{2} - 2}, \tag{416}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1+ur)\ln\left(1 + 1/2\,\frac{1}{t\left(v\sqrt{r} + wr + xr^{3/2} + yr^{p+1}\right)}\right), \tag{417}$$

$$c = 1.709921. \tag{418}$$

C.33 STEST: Test for number of electrons

$$g = \rho(s). \tag{419}$$

C.34 TH1: Tozer and Handy 1998

Density and gradient dependent first row exchange-correlation functional.

$$t = [7/6, 4/3, 3/2, 5/3, 4/3, 3/2, 5/3, \frac{11}{6}, 3/2, 5/3, \frac{11}{6}, 2, 3/2, 5/3, \frac{11}{6}, 2, 7/6, 4/3, 3/2, 5/3, 1], \tag{420}$$

$$u = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0], \tag{421}$$

$$v = [0,0,0,0,1,1,1,1,2,2,2,2,0,0,0,0,0,0,0,0,0], \tag{422}$$

$$w = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0], \tag{423}$$

$$\begin{aligned}\omega = [& \\ & -0.728255, 0.331699, \\ & -1.02946, 0.235703, \\ & -0.0876221, 0.140854, 0.0336982, \\ & -0.0353615, 0.00497930, \\ & -0.0645900, 0.0461795, \\ & -0.00757191, \\ & -0.00242717, 0.0428140, \\ & -0.0744891, 0.0386577, \\ & -0.352519, 2.19805, \\ & -3.72927, 1.94441, 0.128877],\end{aligned} \tag{424}$$

$$n = 21, \tag{425}$$

$$\begin{aligned}R_i = & (\rho(a))^{t_i} \\ & + (\rho(b))^{t_i},\end{aligned} \tag{426}$$

$$S_i = \left(\frac{\rho(a) - \rho(b)}{\rho}\right)^{2u_i}, \tag{427}$$

$$X_i = 1/2\,\frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3\,v_i}}, \tag{428}$$

$$Y_i = \left(\frac{\sigma(aa) + \sigma(bb) - 2\sqrt{\sigma(aa)}\sqrt{\sigma(bb)}}{\rho^{8/3}}\right)^{w_i}, \tag{429}$$

$$f = \sum_{i=1}^{n} \omega_i R_i S_i X_i Y_i, \tag{430}$$

$$G = \sum_{i=1}^{n} 1/2\,\omega_i (\rho(s))^{t_i} \left(\sqrt{\sigma(ss)}\right)^{v_i} \left(\frac{\sigma(ss)}{(\rho(s))^{8/3}}\right)^{w_i} \left((\rho(s))^{4/3\,v_i}\right)^{-1}. \tag{431}$$

C.35 TH2:.

Density and gradient dependent first row exchange-correlation functional.

$$t = [\frac{13}{12}, 7/6, 4/3, 3/2, 5/3, \frac{17}{12}, 3/2, 5/3, \frac{11}{6}, 5/3, \frac{11}{6}, 2, 5/3, \frac{11}{6}, 2, 7/6, 4/3, 3/2, 5/3], \quad (432)$$

$$u = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1], \quad (433)$$

$$v = [0,0,0,0,0,1,1,1,1,2,2,2,0,0,0,0,0,0,0], \quad (434)$$

$$w = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0], \quad (435)$$

$$\omega = [0.678831, -1.75821, 1.27676, -1.60789, 0.365610, -0.181327, 0.146973, 0.147141, -0.0716917, -0.0407167, 0.0214625, -0.000768156, 0.0310377, -0.0720326, 0.0446562, -0.266802, 1.50822, -1.94515, 0.679078], \quad (436)$$

$$n = 19, \quad (437)$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \quad (438)$$

$$S_i = \left(\frac{\rho(a) - \rho(b)}{\rho}\right)^{2u_i}, \quad (439)$$

$$X_i = 1/2\,\frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3\,v_i}}, \quad (440)$$

$$Y_i = \left(\frac{\sigma(aa) + \sigma(bb) - 2\sqrt{\sigma(aa)}\sqrt{\sigma(bb)}}{\rho^{8/3}}\right)^{w_i}, \quad (441)$$

$$f = \sum_{i=1}^{n} \omega_i R_i S_i X_i Y_i, \quad (442)$$

$$G = \sum_{i=1}^{n} 1/2\,\omega_i (\rho(s))^{t_i} \left(\sqrt{\sigma(ss)}\right)^{v_i} \left(\frac{\sigma(ss)}{(\rho(s))^{8/3}}\right)^{w_i} \left((\rho(s))^{4/3\,v_i}\right)^{-1}. \quad (443)$$

C.36 TH3:.

Density and gradient dependent first and second row exchange-correlation functional.

$$t = [7/6, 4/3, 3/2, 5/3, \frac{17}{12}, 3/2, 5/3, \frac{11}{6}, 5/3, \frac{11}{6}, 2, 5/3, \frac{11}{6}, 2, 7/6, 4/3, 3/2, 5/3, \frac{13}{12}], \tag{444}$$

$$u = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0], \tag{445}$$

$$v = [0,0,0,0,1,1,1,1,2,2,2,0,0,0,0,0,0,0,0], \tag{446}$$

$$w = [0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0], \tag{447}$$

$$\omega = [-0.142542, -0.783603, -0.188875, 0.0426830, -0.304953, 0.430407, -0.0997699, 0.00355789, -0.0344374, 0.0192108, -0.00230906, 0.0235189, -0.0331157, 0.0121316, 0.441190, -2.27167, 4.03051, -2.28074, 0.0360204], \tag{448}$$

$$n = 19, \tag{449}$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \tag{450}$$

$$S_i = \left(\frac{\rho(a) - \rho(b)}{\rho}\right)^{2u_i}, \tag{451}$$

$$X_i = 1/2\,\frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3\,v_i}}, \tag{452}$$

$$Y_i = \left(\frac{\sigma(aa) + \sigma(bb) - 2\sqrt{\sigma(aa)}\sqrt{\sigma(bb)}}{\rho^{8/3}}\right)^{w_i}, \tag{453}$$

$$f = \sum_{i=1}^{n} \omega_i R_i S_i X_i Y_i, \tag{454}$$

$$G = \sum_{i=1}^{n} 1/2\,\omega_i (\rho(s))^{t_i} \left(\sqrt{\sigma(ss)}\right)^{v_i} \left(\frac{\sigma(ss)}{(\rho(s))^{8/3}}\right)^{w_i} \left((\rho(s))^{4/3\,v_i}\right)^{-1}. \tag{455}$$

C.37 `TH4:.`

Density an gradient dependent first and second row exchange-correlation functional.

$$t = [7/6, 4/3, 3/2, 5/3, \frac{17}{12}, 3/2, 5/3, \frac{11}{6}, 5/3, \frac{11}{6}, 2, 5/3, \frac{11}{6}, 2, 7/6, 4/3, 3/2, 5/3, \frac{13}{12}], \tag{456}$$

$$u = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0], \tag{457}$$

$$v = [0,0,0,0,1,1,1,1,2,2,2,0,0,0,0,0,0,0,0], \tag{458}$$

$$w = [0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0], \tag{459}$$

$$\omega = [0.0677353, -1.06763, -0.0419018, 0.0226313, -0.222478, 0.283432, -0.0165089, -0.0167204, -0.0332362, 0.0162254, -0.000984119, 0.0376713, -0.0653419, 0.0222835, 0.375782, -1.90675, 3.22494, -1.68698, -0.0235810], \tag{460}$$

$$n = 19, \tag{461}$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \tag{462}$$

$$S_i = \left(\frac{\rho(a) - \rho(b)}{\rho}\right)^{2u_i}, \tag{463}$$

$$X_i = 1/2\,\frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3\,v_i}}, \tag{464}$$

$$Y_i = \left(\frac{\sigma(aa) + \sigma(bb) - 2\sqrt{\sigma(aa)}\sqrt{\sigma(bb)}}{\rho^{8/3}}\right)^{w_i}, \tag{465}$$

$$f = \sum_{i=1}^{n} \omega_i R_i S_i X_i Y_i, \tag{466}$$

$$G = \sum_{i=1}^{n} 1/2\,\omega_i (\rho(s))^{t_i} \left(\sqrt{\sigma(ss)}\right)^{v_i} \left(\frac{\sigma(ss)}{(\rho(s))^{8/3}}\right)^{w_i} \left((\rho(s))^{4/3\,v_i}\right)^{-1}. \tag{467}$$

C.38 THGFC: .

Density and gradient dependent first row exchange-correlation functional for closed shell systems. Total energies are improved by adding DN, where N is the number of electrons and $D = 0.1863$.

$$t = [7/6, 4/3, 3/2, 5/3, 4/3, 3/2, 5/3, \frac{11}{6}, 3/2, 5/3, \frac{11}{6}, 2], \tag{468}$$

$$v = [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2], \tag{469}$$

$$\omega = [\tag{470}$$
$$-0.864448, 0.565130,$$
$$-1.27306, 0.309681,$$
$$-0.287658, 0.588767,$$
$$-0.252700, 0.0223563, 0.0140131,$$
$$-0.0826608, 0.0556080,$$
$$-0.00936227],$$

$$n = 12, \tag{471}$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \tag{472}$$

$$X_i = 1/2 \frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3 v_i}}, \tag{473}$$

$$f = \sum_{i=1}^{n} \omega_i R_i X_i, \tag{474}$$

$$G = \sum_{i=1}^{n} 1/2 \frac{\omega_i (\rho(s))^{t_i} \left(\sqrt{\sigma(ss)}\right)^{v_i}}{\rho^{4/3 v_i}}. \tag{475}$$

C.39 THGFCFO: .

Density and gradient dependent first row exchange-correlation functional. FCFO = FC + open shell fitting.

$$t = [7/6, 4/3, 3/2, 5/3, 4/3, 3/2, 5/3, \frac{11}{6}, 3/2, 5/3, \frac{11}{6}, 2, 3/2, 5/3, \frac{11}{6}, 2, 7/6, 4/3, 3/2, 5/3], \tag{476}$$

$$u = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1], \tag{477}$$

$$v = [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0], \tag{478}$$

$$w = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0], \tag{479}$$

$$\omega = [\quad -0.864448, 0.565130, -1.27306, 0.309681, -0.287658, 0.588767, -0.252700, 0.0223563, 0.0140131, -0.0826608, 0.0556080, -0.00936227, -0.00677146, 0.0515199, -0.0874213, 0.0423827, 0.431940, -0.691153, -0.637866, 1.07565], \tag{480}$$

$$n = 20, \tag{481}$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \tag{482}$$

$$S_i = \left(\frac{\rho(a) - \rho(b)}{\rho}\right)^{2u_i}, \tag{483}$$

$$X_i = 1/2\,\frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3\,v_i}}, \tag{484}$$

$$Y_i = \left(\frac{\sigma(aa) + \sigma(bb) - 2\sqrt{\sigma(aa)}\sqrt{\sigma(bb)}}{\rho^{8/3}}\right)^{w_i}, \tag{485}$$

$$f = \sum_{i=1}^{n} \omega_i R_i S_i X_i Y_i, \tag{486}$$

$$G = \sum_{i=1}^{n} 1/2\,\omega_i\,(\rho(s))^{t_i}\left(\sqrt{\sigma(ss)}\right)^{v_i}\left(\frac{\sigma(ss)}{(\rho(s))^{8/3}}\right)^{w_i}\left((\rho(s))^{4/3\,v_i}\right)^{-1}. \tag{487}$$

C.40 `THGFCO`: .

Density and gradient dependent first row exchange-correlation functional.

$$t = [7/6, 4/3, 3/2, 5/3, 4/3, 3/2, 5/3, \frac{11}{6}, 3/2, 5/3, \frac{11}{6}, 2, 3/2, 5/3, \frac{11}{6}, 2, 7/6, 4/3, 3/2, 5/3], \tag{488}$$

$$u = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1], \tag{489}$$

$$v = [0,0,0,0,1,1,1,1,2,2,2,2,0,0,0,0,0,0,0,0], \tag{490}$$

$$w = [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0], \tag{491}$$

$$\omega = [\quad -0.962998, 0.860233, -1.54092, 0.381602, -0.210208, 0.391496, -0.107660, -0.0105324, 0.00837384, -0.0617859, 0.0383072, -0.00526905, -0.00381514, 0.0321541, -0.0568280, 0.0288585, 0.368326, -0.328799, -1.22595, 1.36412], \tag{492}$$

$$n = 20, \tag{493}$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \tag{494}$$

$$S_i = \left(\frac{\rho(a) - \rho(b)}{\rho}\right)^{2u_i}, \tag{495}$$

$$X_i = 1/2 \frac{\left(\sqrt{\sigma(aa)}\right)^{v_i} + \left(\sqrt{\sigma(bb)}\right)^{v_i}}{\rho^{4/3 v_i}}, \tag{496}$$

$$Y_i = \left(\frac{\sigma(aa) + \sigma(bb) - 2\sqrt{\sigma(aa)}\sqrt{\sigma(bb)}}{\rho^{8/3}}\right)^{w_i}, \tag{497}$$

$$f = \sum_{i=1}^{n} \omega_i R_i S_i X_i Y_i, \tag{498}$$

$$G = \sum_{i=1}^{n} 1/2\, \omega_i (\rho(s))^{t_i} \left(\sqrt{\sigma(ss)}\right)^{v_i} \left(\frac{\sigma(ss)}{(\rho(s))^{8/3}}\right)^{w_i} \left((\rho(s))^{4/3 v_i}\right)^{-1}. \tag{499}$$

C.41 `THGFL`: .

Density dependent first row exchange-correlation functional for closed shell systems.

$$t = [7/6, 4/3, 3/2, 5/3], \tag{500}$$

$$\omega = [\quad -1.06141, 0.898203, -1.34439, 0.302369], \tag{501}$$

$$n = 4, \tag{502}$$

$$R_i = (\rho(a))^{t_i} + (\rho(b))^{t_i}, \tag{503}$$

$$f = \sum_{i=1}^{n} \omega_i R_i. \tag{504}$$

C.42 VSXC: .

$$p = [-0.98, 0.3271, 0.7035], \tag{505}$$

$$q = [-0.003557, -0.03229, 0.007695], \tag{506}$$

$$r = [0.00625, -0.02942, 0.05153], \tag{507}$$

$$t = [-0.00002354, 0.002134, 0.00003394], \tag{508}$$

$$u = [-0.0001283, -0.005452, -0.001269], \tag{509}$$

$$v = [0.0003575, 0.01578, 0.001296], \tag{510}$$

$$\alpha = [0.001867, 0.005151, 0.00305], \tag{511}$$

$$g = (\rho(s))^{4/3} F(\chi(s), zs, p_1, q_1, r_1, t_1, u_1, v_1, \alpha_1) + ds\,\varepsilon(\rho(s), 0) F(\chi(s), zs, p_2, q_2, r_2, t_2, u_2, v_2, \alpha_2), \tag{512}$$

$$G = (\rho(s))^{4/3} F(\chi(s), zs, p_1, q_1, r_1, t_1, u_1, v_1, \alpha_1) + ds\,\varepsilon(\rho(s), 0) F(\chi(s), zs, p_2, q_2, r_2, t_2, u_2, v_2, \alpha_2), \tag{513}$$

$$f = F(x, z, p_3, q_3, r_3, t_3, u_3, v_3, \alpha_3)(\varepsilon(\rho(a), \rho(b)) - \varepsilon(\rho(a), 0) - \varepsilon(\rho(b), 0)), \tag{514}$$

$$x = (\chi(a))^2 + (\chi(b))^2, \tag{515}$$

$$zs = \frac{\tau(s)}{(\rho(s))^{5/3}} - cf, \tag{516}$$

$$z = \frac{\tau(a)}{(\rho(a))^{5/3}} + \frac{\tau(b)}{(\rho(b))^{5/3}} - 2cf, \tag{517}$$

$$ds = 1 - \frac{(\chi(s))^2}{4zs + 4cf}, \tag{518}$$

$$F(x,z,p,q,c,d,e,f,\alpha) = \frac{p}{\lambda(x,z,\alpha)} + \frac{qx^2+cz}{(\lambda(x,z,\alpha))^2} + \frac{dx^4+ex^2z+fz^2}{(\lambda(x,z,\alpha))^3}, \tag{519}$$

$$\lambda(x,z,\alpha) = 1 + \alpha\left(x^2 + z\right), \tag{520}$$

$$cf = 3/5\,3^{2/3}\left(\pi^2\right)^{2/3}, \tag{521}$$

$$T = [0.031091, 0.015545, 0.016887], \tag{522}$$

$$U = [0.21370, 0.20548, 0.11125], \tag{523}$$

$$V = [7.5957, 14.1189, 10.357], \tag{524}$$

$$W = [3.5876, 6.1977, 3.6231], \tag{525}$$

$$X = [1.6382, 3.3662, 0.88026], \tag{526}$$

$$Y = [0.49294, 0.62517, 0.49671], \tag{527}$$

$$P = [1,1,1], \tag{528}$$

$$\varepsilon(\alpha,\beta) = (\alpha+\beta)\Bigg(e(l(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1) - \frac{e(l(\alpha,\beta),T_3,U_3,V_3,W_3,X_3,Y_3,P_3)\,\omega(\zeta(\alpha,\beta))\left(1-(\zeta(\alpha,\beta))^4\right)}{c} + (e(l(\alpha,\beta),T_2,U_2,V_2,W_2,X_2,Y_2,P_2) - e(l(\alpha,\beta),T_1,U_1,V_1,W_1,X_1,Y_1,P_1))\,\omega(\zeta(\alpha,\beta))\,(\zeta(\alpha,\beta))^4\Bigg), \tag{529}$$

$$l(\alpha,\beta) = 1/4\,\sqrt[3]{3}\,4^{2/3}\sqrt[3]{\frac{1}{\pi(\alpha+\beta)}}, \tag{530}$$

$$\zeta(\alpha,\beta) = \frac{\alpha-\beta}{\alpha+\beta}, \tag{531}$$

$$\omega(z) = \frac{(1+z)^{4/3} + (1-z)^{4/3} - 2}{2\sqrt[3]{2} - 2}, \tag{532}$$

$$e(r,t,u,v,w,x,y,p) = -2t(1 + ur)\ln\left(1 + 1/2\,\frac{1}{t\left(v\sqrt{r} + wr + xr^{3/2} + yr^{p+1}\right)}\right), \tag{533}$$

$$c = 1.709921. \tag{534}$$

C.43 `VWN3`: Vosko-Wilk-Nusair (1980) III local correlation energy

VWN 1980(III) functional

$$x = 1/4\,\sqrt[6]{3}4^{5/6}\sqrt[6]{\frac{1}{\pi\rho}}, \tag{535}$$

$$\zeta = \frac{\rho(a) - \rho(b)}{\rho}, \tag{536}$$

$$f = \rho e, \tag{537}$$

$$k = [0.0310907, 0.01554535, -1/6\,\pi^{-2}], \tag{538}$$

$$l = [-0.409286, -0.743294, -0.228344], \tag{539}$$

$$m = [13.0720, 20.1231, 1.06835], \tag{540}$$

$$n = [42.7198, 101.578, 11.4813], \tag{541}$$

$$e = \Lambda + z(\lambda - \Lambda), \tag{542}$$

$$y = \frac{9}{8}(1 + \zeta)^{4/3} + \frac{9}{8}(1 - \zeta)^{4/3} - 9/4, \tag{543}$$

$$\Lambda = q(k_1, l_1, m_1, n_1), \tag{544}$$

$$\lambda = q(k_2, l_2, m_2, n_2), \tag{545}$$

$$\begin{aligned} &q(A, p, c, d) \\ &\quad = A \left(\ln\left(\frac{x^2}{X(x,c,d)}\right) \right. \\ &\qquad + 2c \arctan\left(\frac{Q(c,d)}{2x+c}\right) (Q(c,d))^{-1} \\ &\qquad - cp \left(\ln\left(\frac{(x-p)^2}{X(x,c,d)}\right) \right. \\ &\qquad + 2(c \\ &\qquad\qquad \left.\left. + 2p) \arctan\left(\frac{Q(c,d)}{2x+c}\right) (Q(c,d))^{-1}\right) (X(p,c,d))^{-1} \right), \end{aligned} \tag{546}$$

$$\begin{aligned} &Q(c,d) \\ &\quad = \sqrt{4d - c^2}, \end{aligned} \tag{547}$$

$$\begin{aligned} &X(i,c,d) \\ &\quad = i^2 \\ &\qquad + ci \\ &\qquad + d, \end{aligned} \tag{548}$$

$$z = 4 \frac{y}{9\sqrt[3]{2} - 9}. \tag{549}$$

C.44 `VWN5`: Vosko-Wilk-Nusair (1980) V local correlation energy

VWN 1980(V) functional. The fitting parameters for $\Delta\varepsilon_c(r_s, \zeta)_V$ appear in the caption of table 7 in the reference.

$$x = 1/4 \sqrt[6]{3} 4^{5/6} \sqrt[6]{\frac{1}{\pi\rho}}, \tag{550}$$

$$\zeta = \frac{\rho(a) - \rho(b)}{\rho}, \tag{551}$$

$$f = \rho e, \tag{552}$$

$$k = [0.0310907, 0.01554535, \\ -1/6\pi^{-2}], \tag{553}$$

$$l = [\\ -0.10498, \\ -0.325, \\ -0.0047584], \tag{554}$$

$$m = [3.72744, 7.06042, 1.13107], \tag{555}$$

$$n = [12.9352, 18.0578, 13.0045], \tag{556}$$

$$e = \Lambda + \alpha y \left(1 + h\zeta^4\right), \tag{557}$$

$$y = \frac{9}{8}(1 + \zeta)^{4/3} + \frac{9}{8}(1 - \zeta)^{4/3} - 9/4, \tag{558}$$

$$h = 4/9 \frac{\lambda - \Lambda}{\left(\sqrt[3]{2} - 1\right)\alpha} - 1, \tag{559}$$

$$\Lambda = q(k_1, l_1, m_1, n_1), \tag{560}$$

$$\lambda = q(k_2, l_2, m_2, n_2), \tag{561}$$

$$\alpha = q(k_3, l_3, m_3, n_3), \tag{562}$$

$$q(A, p, c, d) = A\left(\ln\left(\frac{x^2}{X(x,c,d)}\right) + 2c \arctan\left(\frac{Q(c,d)}{2x+c}\right)(Q(c,d))^{-1} - cp\left(\ln\left(\frac{(x-p)^2}{X(x,c,d)}\right) + 2(c + 2p)\arctan\left(\frac{Q(c,d)}{2x+c}\right)(Q(c,d))^{-1}\right)(X(p,c,d))^{-1}\right), \tag{563}$$

$$Q(c,d) = \sqrt{4d - c^2}, \tag{564}$$

$$X(i,c,d) = i^2 + ci + d. \tag{565}$$

D License information

The Molpro source code contains some external code which is listed in this section. Molpro binaries may, in some instances, contain compiled versions of this code also.

D.1 BLAS

Copies of some of the netlib BLAS routines are included in the Molpro source code. The source code is identical except for the addition of a revision control header, and renaming of the file suffix from .f to .fh. These routines are only used if the user does not provide an external BLAS library. In addition, as suggested in the comments, the STOP statement in subroutine XERBLA has been modified.

D.2 LAPACK

Copies of some of the netlib LAPACK routines are included in the Molpro source code. The source code is identical except for the addition of a revision control header, and renaming of the file suffix from .f to .fh. These routines are only used if the user does not provide an external LAPACK library. The license follows.

D.3 Boost

A complete unmodified copy of the boost directory from the Boost C++ library is included. The license follows.

Index