Firefly 8.0.0 manual

Version 2013-11-12

Table of Contents

Introduction

About this manual

The goal of this manual is to provide users with a description of all capabilities of the Firefly QC software package. It consists of two parts: a multi-chapter document that describes Firefly's core functionality and a list of all keywords that are available in Firefly.

The main document provides an overview of all important functionality present in Firefly. It deals with several theories and calculation types and, in addition, explains also how to set up Firefly in both Windows and Linux environments. It is intended for users new to Firefly (or new to a specific feature in Firefly) and assumes a basic to moderate level of QC knowledge. However, as this document contains several tips, users with a higher level of knowledge might also find this document useful. Note that as only an overview of Firefly's functionality is provided, many of the less important (but not necessarily less useful) keywords are not discussed here. Instead, these can be found in the list of keywords.

The list of keywords provides a complete listing of all options accessible in Firefly. This document contains very specific information for each feature in Firefly and will foremost be useful to the more experienced users. However, beginning users are also encouraged to read through the list – they might come across some useful keywords not mentioned in the main document. In the list, keywords are organized by the keyword group (*e.g.* $CONTRL, $SYSTEM, etc.) they belong to. The list is also available as a separate PDF document for the quick lookup of keywords.

This manual is freely available from the Firefly website and falls under the copyright policy of the Firefly website. It may not be printed or republished without the explicit permission of the copyright holder.

Overview of capabilities

The table below summarizes the current main capabilities of Firefly:

Wavefunction	RHF	UHF	ROHF	GVB	MCSCF
Semiempirical SCF	dm	dm	dm	dm	-
SCF energy	cdp	cdp	cdp	cdp	cdp
SCF analytical gradient	cdp	cdp	cdp	cdp	cdp
SCF analytical Hessian	cdp*	-	cdp*	cdp*	-
DFT energy	cdp	cdp	cdp	-	-
DFT analytical gradient	cdp	cdp	cdp	-	-
CIS energy	cdp	-	-	-	-

CIS analytical gradient	cdp	-	-	-	-
TDHF (RPA) energy	cdp	-	-	-	-
TDDFT energy	cdp	-	-	-	-
MP2 energy	cdp	cdp	cdp	-	cdmp
MP2 analytical gradient	cdp	-	-	-	-
MP3 energy	cdm	-	-	-	-
MP4 energy	cdmp*	-	-	-	-
CI energy	cdp	-	cdp	cdp	cdp
CI analytical gradient	cd*	-	-	-	-

Legend: c - conventional
d - direct/semidirect
m - multithreaded
p - parallel
* - additional notes:
- CI analytical gradients and SCF analytical Hessians are programmed for spd basis sets only (note that CIS gradients are available for any supported basis sets);
- For GVB, SCF analytical Hessians are available only for a selected subset of possible GVB-type wavefunctions;
- The MP4(SDQ) code is multithreaded but not parallel while the MP4(SDTQ) code is both multithreaded and parallel for (T) part.

Firefly and GAMESS (US)

Due to historical reasons, Firefly and GAMESS (US) share a lot of functionality. When the Firefly project started in 1993 at MSU, it was based on the source code of the GAMESS (US) package developed by members of Mark Gordon's group at ISU. Modifications to the source code were made by Alex Granovsky (who at that time was working at MSU), and the modified package was known as “PC GAMESS”. Initially, PC GAMESS was only available locally at MSU, but in 1997 it became available outside of MSU as part of the public GAMESS (US) distribution.

Up until 1999, features added to GAMESS (US) were also incorporated in PC GAMESS. The latest release of GAMESS (US) of which the source code was used is the October 25th, 1999 release. After this release, the development of PC GAMESS and GAMESS (US) became independent. However, PC GAMESS did remain part of the GAMESS (US) distribution and a certain degree of compatibility between the two packages was maintained. With the release of version 7.1.C in 2008, PC GAMESS became completely forked from GAMESS (US) as an independent package and the name was changed to “Firefly” (though the PC GAMESS name was also used till the release of version 7.1.G).

An overview of similarities and differences between Firefly and GAMESS (US) is given below.

<u>Similarities with GAMESS (US)</u>
Firefly supports all functionalities of GAMESS (US) up to the October 25th, 1999 release of GAMESS (US). Many additional features added to GAMESS (US)

since 1999 are supported as well. *Ab initio* SCF wavefunctions for RHF, UHF, ROHF, GVB and MCSCF cases are available. Correlation corrections include configuration interaction (CI) and Møller-Plesset (MP) perturbation theory. For MCSCF wavefunctions, correlation corrections can be calculated with MCQDPT2. Excited electronic states can also be described with configuration interaction with singles (CIS), RPA (*i.e.* TDHF), and TDDFT. Support for effective core potentials and a variety of density functional (DFT) functionals is available. Geometry optimizations can be performed with analytical gradients at the HF, MP2, CI, DFT, and MCSCF levels. Analytical Hessians can be computed for RHF, ROHF and GVB wavefunctions while numerical Hessians are available for all methods that support analytic gradients. Finally, numerical gradients and double-numerical Hessians are available for all methods.

Differences with GAMESS (US)

Firefly provides fast and efficient algorithms for Møller-Plesset second, third, and fourth order energy corrections. The DFT code in the Firefly is completely different with respect to that of GAMESS (US) and the two programs support different density functionals. In Firefly, MCSCF gradients can be calculated semi-numerically from state-averaged orbitals, whereby state-tracking is supported. MCSCF can be used for the location of conical intersections and interstate crossings. MCQDPT2 calculations in Firefly can be sped up through a Resolvent fitting technique, and an improved theory, namely XMCQDPT2, is available. Finally, Firefly has different engines for geometry optimizations and relaxed surface scans.

Firefly lacks some features that have been implemented in later versions of GAMESS (US). For example, Firefly does not support coupled cluster and fragment molecular orbitals methods. ORMAS-type CI and MCSCF calculations are not possible. Also, it is not possible to do all-electron relativistic calculations with Firefly. Note that both Firefly and GAMESS (US) lack support for h and higher angular momentum basis functions.

Release history

Firefly 8.0.0, build #8240. Official Firefly binaries released to the public September 5, 2013

System-wide changes:

- New hierarchical eXtreme Parallel (xp) parallel execution model
- Improved performance of threaded code with all presently known issues fixed
- Improved memory management which is capable to allocate more memory and to handle it in a more intelligent way
- Improved support of Lustre and other non-local filesystems with all presently known compatibility issues resolved
- Support of Intel's AVX-enabled processors
- Support of Intel's AVX2-enabled processors and FMA3
- Support of AMD Bulldozer processors and FMA4
- New high-performance intelligent numerical gradients mode

- New, updated documentation for Firefly. This documentation combines information taken from the old manual, the various README files, and the forums, and also many new chapters

Changes to HF and DFT code:

- Improved DIIS code with lots of additional modes of operation, configurable using dedicated input options
- Better parallel scalability of DFT code due to faster DIIS
- Improved accuracy of DFT quadratures
- DFT-D empirical dispersion correction
- New GGA and hybrid GGA functionals
- Support of double-hybrid functionals
- New HFX parameter allowing users to modify the fraction of the exact HF exchange used in hybrid functionals. Helpful in TDDFT calculations of Rydberg and charge-transfer states.

Changes to MP2 code:

- SCS and SOS MP2
- More intelligent handling of I/O errors while running in parallel

Changes to CUDA-enabled MP4 code:

- Changes for CUDA 4.x and 5.x

Changes to CASSCF, MCQDPT, and XMCQDPT2 code:

- New high-performance determinant-based CASCI/CASSCF code with better parallel scalability and efficiency, and very modest memory demands
- Multiple functional and performance improvements and extensions to MCQDPT and XMCQDPT module
- Improved conical intersection location code now capable to deal with arbitrary averaging of multiple states in SA-CASSCF

Other changes:

- CPCM solvent model
- Fully variational DPCM and CPCM-like extended solvent models
- PCM is enabled for UHF/UDFT and ROHF/RODFT
- Extended restart capabilities
- Improvements and extensions to surface scan module
- Multiple bugfixes and minor improvements

7.1.G (*i.e.*, 7.1.16), build #5618. Official Firefly binaries released to the public December 4, 2009.

Main new features are:

- Multiple minor bugfixes and improvements.
- More aggressive set of default settings for better performance and stability.

- Improved performance on AMD Phenom/Phenom II/Barcelona/Shanghai/Istanbul processors.
- Improved performance of MP2 Energy/Energy Gradient method=1 code.
- Improved DLC engine.
- Improved code for relaxed PES scans using DLCs.
- Improvements to P2P communication interface.
- Completely redesigned threading model for better multithreaded performance and CUDA interoperability.
- Improved MP4 code with CUDA support.
- Program name changed to Firefly.

7.1.F (*i.e.*, 7.1.15), build #5211. Official PC GAMESS/Firefly binaries released to the public February 24, 2009.

This is mainly the maintenance/bugfix release over PC GAMESS/Firefly version 7.1.E. Main new features and bugfixes are:

- Support of Mac OS X/Intel platform.
- Better multithreaded performance on some platforms/hardware.
- Workaround for nasty CPUID bug.
- New fully dynamically linked MPICH/NPTL binaries for Linux.
- Faster GVB/MCSCF gradients.
- State-Specific gradient code for State-Averaged MCSCF allows use of PCM solvation model for excited states optimization and location of Conical Intersections.

7.1.E (*i.e.*, 7.1.14), build #5190. Official PC GAMESS/Firefly binaries released to the public January 11, 2009.

This is mainly the maintenance/bugfix release over PC GAMESS/Firefly version 7.1.C. However, it adds some new important features:

- Completely new DLC engine and improved default geometry optimizer.
- Better support of Core i7 (as well as other Nehalem core processors).
- More efficient I/O under Windows Vista/Windows Server 2008 R1.
- Support of httfix option under Linux.
- New mpich2-linked binaries for Linux.
- Fixes for compatibility with the most recent Linux distributions.

7.1.C (*i.e.*, 7.1.12), build #5014. Official PC GAMESS/Firefly binaries released to the public October 17, 2008.

Accumulates multiple bugfixes, performance improvements, and new functionality introduced since the release of PC GAMESS v. 7.1.5:

- Massive internal engine rewrites.
- Extended P2P interface with support of up to 1024 nodes.
- New eXtended MultiConfiguration QuasiDegenerate Perturbation Theory (XMCQDPT) code (see http://classic.chem.msu.su/gran/gamess/xmcqdpt.pdf).

- Improved CPHF=AO code for analytical RHF second derivatives with reduced memory requirements and better parallel scalability. Note it is now turned on by default!
- Improved, much more stable and faster SOSCF code available for RHF/ROHF/UHF/GVB/MCSCF wavefunctions optimization
- Improved METHOD=GDIIS geometry optimizations.
- Updated NBO version 5.G module.
- Relaxed 1D and 2D PES scans over bonds, angles, or torsions in DLCs (list of new available options is to be published soon in "Manuals" section).
- Added RM1 parameterization to MOPAC code.
- Improved stability of the ALDET's Davidson CI diagonalization code.
- Improved state-averaged MCSCF gradient code.
- Improved MCSCF state-tracking feature.
- New code for location of conical intersections (CI).
- Improved DIIS code with reduced memory demands.
- More accurate memory demands estimation for CIS/TD code.
- New, much more stable code to convert internals to Cartesians.
- DFT gradients are now slightly more accurate.
- Support of Gaussian-style O3LYP functional.
- Support of Intel Atom processors.
- Initial optimization for Intel Core i7 (codename Nehalem) microarchitecture.
- Improved compatibility with some buggy Linux versions.
- Linux/MPICH version of PC GAMESS now uses SSH by default.

7.1.6 - 7.1.11 (*i.e.*, 7.1.B), internal builds.

7.1.5, build #4630, December 23, 2007.

Available as the set of the update patches to the original PC GAMESS version 7.1 build # 4471 from the Downloads section of the PC GAMESS homepage at MSU. Incorporates multiple bugfixes and performance improvements (especially for AMD Phenom/Barcelona and Intel Core 2 processors), as well as some new functionality.

7.1.4, November 2007, internal build.

7.1.3, October 2007, internal build.

7.1.2, October 2007, internal build.

7.1.1, September 2007, internal build.

7.1, Official PC GAMESS binaries released to the public, build #4471, September 5, 2007.

This version accumulates all the changes introduced in intermediate releases versions 7.0.2-7.0.7, as well as state-specific analytic gradients for state-averaged MCSCF and completely new state of the art serial and parallel semidirect MP2 gradient program with excellent performance, scalability and very modest memory demands.

7.1 final RC, build #4400, June 27, 2007.

New PC GAMESS version features completely new state of the art parallel semidirect MP2 gradient program with excellent performance, scalability and very modest memory demands. The updated PC GAMESS binaries were available from MSU for registered PC GAMESS users for testing purposes upon request.

7.0.7, build #4194, April 23, 2007.

Dynamically linked Linux binaries of PC GAMESS are now fully nptl compatible and were carefully tested with MPICH, MPICH-GM, MPICH-MX, OpenMPI, Scali MPI, HP-MPI, Intel MPI 3.0 and Infinipath MPI implementations. The updated PC GAMESS binaries were available from MSU for registered PC GAMESS users for testing purposes upon request.

7.0.6, April 2007, internal build.

7.0.5, March 2007, internal build.

7.0.4, build #4102, February 16, 2007, internal build.

- More numerically stable PCM energy and gradient code
- Serious improvements of parallel scalability of ALDET CASCI and CASSCF code The updated PC GAMESS binaries were available from MSU for registered PC GAMESS users for testing purposes upon request.

7.0.3, build #4063, January 21, 2007, internal build.

- Shared memory version of P2P interface (Windows)
- support of OPTX, OLYP, and O3LYP XC functionals
- Parallel PCM with enlarged PCM dimensions

The updated PC GAMESS binaries were available from MSU for registered PC GAMESS users for testing purposes upon request.

7.0.2, build #4020, October 8, 2006, internal build.

Added completely new, much faster MCQDPT2 code developed at MSU. The updated PC GAMESS binaries were available from MSU for registered PC GAMESS users for testing purposes upon request.

7.0.1, build #3970, August 17, 2006

- Better optimization for Intel Core 2 (Woodcrest/Conroe/Merom processors) microarchitecture
- Minor bugfixes and improvements

7.0, build #3910, May 11, 2006

- Initial optimization for Intel Core 2 (Woodcrest/Conroe/Merom processors) microarchitecture
- Time-dependent Hartree-Fock (TDHF) and density functional theory (TDDFT)
- Configuration Interaction Singles (CIS)
- Static and dynamic hyperpolarizabilities via TDDFT
- Large-scale direct and conventional parallel MCSCF
- Semidirect MCQDPT
- Support of general contraction basis sets
- Faster PCM
- Improved Linux compatibility
- Unified support of different MPI implementations under Windows

6.5, Release Candidate #9, January 2005, was available to beta testers only

- Support for spherical basis functions
- Quantum Fast Multipole Method and Linear Exchange for linear scaling HF and DFT
- Faster direct SCF/DFT code
- New 2-e integrals code
- Much faster MCQDPT code with SMP support

6.4, March 2004

- Fast hybrid DFT
- Support of AMD Opteron and Intel Pentium M processors
- Improved performance and stability
- Dynamic load balancing on the top of P2P interface
- Improvements to P2P parallel MP2 energy code
- Better support of SMP including HTT environment
- Several other enhancements and extensions (cube facility etc...)

6.3, June 2003

- Optimized binaries for AMD Athlon and Intel Pentium 4 processors
- New parallel mode P2P communication interface
- Efficient parallel MP2 energy code based on P2P model allowing thousands of basis functions
- Various bugfixes and enhancements

6.2, 2001

- Optimized binaries for AMD Athlon
- Cumulative bugfixes and improvements

6.1, 2001, Internal release

6.0, May 2000

- PC GAMESS is now based on updated GAMESS (US) sources
- Pentium/Pentium Pro/Pentium II/Pentium III-optimized native Linux SMP-aware PC GAMESS ELF binaries
- Pentium Pro/Pentium II/Pentium III-optimized OS/2-based SMP & JFS-aware binaries
- Large files are supported via transparent file splitting on all 32-bit file systems (2 or 4 GB limit)
- Enhanced convergence options for solving MP2 CPHF equations
- PG GAMESS can now write out cube files for visualization
- Advanced surface scan interface
- NBO module is incorporated into PC GAMESS
- Support of direct MP3/MP4
- GDIIS method for geometry optimization

5.4, 1999

- Changes for faster MP3/MP4
- Changes to better support SMP throughout the code
- OS and CPU autodetection

5.3, 1999, Internal release

- Changes to use updated MKL libraries
- Changes to support parallel execution

5.2, May 1999

- New RHF MP4(SDTQ) (*i.e.*, full MP4) energy module is added
- The SMP scaling properties of MP3/MP4 calculations are improved considerably
- The speed of MP2 gradient calculations for non-abelian symmetry groups is increased
- The speed of Conjugated Gradient solver (used mainly during calculations of analytical second derivatives) is improved significantly
- The Finite Field module is changed to avoid reevaluations of 2-e integrals if possible
- The ECP integrals module is rewritten to avoid numerical instability problems
- It is now possible to run the PC GAMESS under Linux using customized Wine

5.1, Build #1519, October 1998

- RHF MP3/MP4(SDQ) energy modules are added.
- Memory management is improved again (Win32 and OS/2-specific).
- The CPHF solver used during MP2-level geometry optimization/hessian runs is now faster.
- GAMESS bugs concerning PCM-level geometry optimization/hessian runs are fixed.
- The precision of the HF density matrix generated by SOSCF converger is enhanced

5.0, August 1998

- Based on May 8, 1998 GAMESS code
- Faster TDHF, MCQDPT, MP2 energy, and MP2 energy gradient calculations
- The disk usage is reduced for non-FORS GUGA CI jobs
- Improved support of SMP (Windows NT specific)

4.5, Internal Release

4.4, March 1998

- Completely new RHF/ROHF/UHF MP2 energy program
- Additional GUGA CI Hamiltonian packing
- Improved memory management (Windows NT specific)

4.3, October 1997

- Updated MP2, MP2 gradient, and CI gradient codes
- Large direct access files (> 2 GB) are supported under Windows NT
- Ctrl-C and Ctrl-Break signal handling is implemented

4.2, Internal Release

4.1, Build number 1220

- The fast (non-Fortran) file I/O as well as the AO integrals and GUGA CI Hamiltonian packing are implemented
- Large sequential access files (> 2 GB) are supported under Windows NT

4.0, Build number 1080

- First public release based on the original GAMESS sources dated March 18, 1997

Versions 1.0 - 3.0 were used locally at MSU.

Citing Firefly

You must use the following Firefly reference in your publications:

> *Alex A. Granovsky, Firefly version 8.0.0, www http://classic.chem.msu.su/gran/firefly/index.html*

This reference should be explicitly given in the appropriate section of your paper just like any other regular references are. Specifically, this should not be just a reference given in pass within the main body of paper.

The recommended form is as follows:

> *Firefly QC package [1], which is partially based on the GAMESS (US) [2] source code.*
>
> *1. Alex A. Granovsky, Firefly version 8.0.0, www http://classic.chem.msu.su/gran/firefly/index.html*
>
> *2. M.W.Schmidt, K.K.Baldridge, J.A.Boatz, S.T.Elbert, M.S.Gordon, J.H.Jensen, S.Koseki, N.Matsunaga, K.A.Nguyen, S.Su, T.L.Windus, M.Dupuis, J.A.Montgomery J.Comput.Chem. 14, 1347-1363 (1993)*

Installing and Running Firefly

General information

Firefly is freely available and can be obtained from the Firefly website. The Windows and Linux versions of the program are distributed as a compressed archive – to be precise, as a RAR file inside a ZIP file. The RAR file is password protected; one first has to register in order to obtain the password which is necessary for decompression. Instructions for the registration can be found inside the ZIP file. The Mac OS X version is as an encrypted installation image (the password will be asked during installation process).

Before downloading, one has to decide which version to download. First of all, Firefly can be obtained for Windows, Linux, and OS X. The Windows version can run in serial and parallel mode, and is compatible with various MPI implementations. The Linux version can also work with various MPI implementations, however, a separate download is provided for each MPI type. Archives for MPI implementations not listed in the download section on the Firefly website can be provided upon request.

Up until version 8.0.0, one could also choose from various compiled/optimized versions of Firefly, designed to run with maximum efficiency on a certain CPU architecture. However, as of Firefly 8.0.0 only the so-called “P4” version is available from the Firefly website. The P4 version is optimized for Intel Pentium 4, Intel Pentium D, Intel Xeon, Intel Core 2, and Intel Core i3/i5/i7 processors as well as for AMD Phenom, AMD “Barcelona” Opteron, and newer AMD processors. It is also possible to run this version on older processors provided that they support SSE2, but performance might not be optimal in such a case. Binaries optimized for older processors are available upon request. Note that a 64-bit version of Firefly is currently not available.

Specifics regarding the installation and execution of Firefly on Windows and Linux are given in the following sections.

An important bit of information which applies to all versions of Firefly is that the execution of a job results in the creation of many new files. Some of these contain formatted data (such as the PUNCH file), others are temporary files (such as the DICTNRY file). Firefly will refuse to start a new job if the formatted data file(s) from a previous job are still in the working directory (as they may contain something useful), these should therefore either be deleted or moved to a different directory. Temporary files don’t necessarily have to be deleted, though it is advisable to do so (unless a job is restarted).

Windows

The installation of Firefly under Windows is quite straightforward. The first step is to extract the contents of the downloaded RAR archive to a directory for which Firefly has read and write permissions. This is all that is required for executing a single instance of Firefly in serial mode. Running multiple, separate jobs is possible, the only requirement is that each instance of Firefly runs from its own directory.

When executing Firefly, the program will look for input in a file called "input" in the directory which contains the firefly.exe executable, though it is also possible to specify an input file with the “-i” command line option. By default, the output is written to the “stdout” device (*i.e.*, it will appear onscreen if no I/O redirection is used). Therefore, if one would like the output to be saved to a file, an I/O redirection has to be used. When executing Firefly on an NT-based version of Windows, one could for example specify:

```
firefly.exe >test.out 2>&1
```

This will cause the output to appear in the file test.out. The addition of “2>&1” will cause most severe error messages normally sent to console also to be send to test.out instead. (Firefly high-level error messages will be written to test.out anyway). This however does not work on Windows 95/98/ME as command interpreter's I/O redirection is more limited on these operating systems. In these cases, one can only specify:

```
firefly.exe >test.out
```

If one is executing Firefly from the Windows Powershell, it is also possible to send the output simultaneously to the screen and to a file by using the tee command. This is not recommended though, as the output file produced in this way cannot easily be opened by other programs like ChemCraft due to an ASCII/binary problem.

Additionally, it is also possible to tell Firefly to write the output to a file of your choosing with the “-o” command line option:

```
firefly.exe –o test.out
```

Windows MPI implementations

The instructions in the previous section cover the basic steps needed to get Firefly installed and running in serial mode. Running Firefly in parallel mode however requires some additional work. Specifically, one has to choose an MPI implementation, even if Firefly will be run on a standalone SMP/multicore system (*i.e.* a single multicore computer).

The Windows version of Firefly is supplied with a series of DLL files which after extraction of the RAR archive will be in a directory named BINDINGS. Each of these files makes Firefly compatible with a particular MPI implementation. The list of supported MPI implementations and corresponding DLL files is as follows:

* MPIBIND.SEQ.DLL - no MPI implementation is used, this file converts Firefly into a purely sequential program

* MPIBIND.NT-MPICH-SMP.DLL - to be used with mpich_smp.dll library (included into the Firefly distribution, taken from NT-MPICH version 1.3) on standalone SMP/multicore systems

* MPIBIND.NT-MPICH.DLL - to be used with NT-MPICH

* MPIBIND.DEINOMPI.DLL - to be used with Deino MPI

* MPIBIND.INTELMPI.DLL - to be used with Intel MPI

* MPIBIND.MPICH.NT.DLL - to be used with MPICH.NT

* MPIBIND.MPICH2.DLL - to be used with MPICH2

* MPIBIND.WMPI-1.3.DLL - to be used with WMPI v. 1.3 (included into the Firefly distribution)

* MPIBIND.WMPI-1.6.DLL - to be used with WMPI v. 1.6 and above

* MPIBIND.WMPI-II.DLL - to be used with WMPI-II

* MPIBIND.MPIPRO.DLL - to be used with MPI Pro

* MPIBIND.MSMPI.DLL - to be used with Microsoft MPI library for Windows Compute Cluster Edition/Compute Cluster Server

The recommended MPI implementation for Firefly on pure SMP/multicore systems is NT-MPICH-SMP. For self-made (non CCS-based) Windows clusters the recommended implementation is NT-MPICH. Finally, for Windows CCS systems, the recommended MPI implementation is MS MPI.

Making Firefly compatible with a certain MPI implementation requires one to copy the appropriate DLL file from the BINDINGS directory into the Firefly installation directory and rename it to “mpibind.dll” (after deleting the old version of mpibind.dll). Note that because Firefly is 32-bit program you will need to use 32-bit MPI libraries, even when you are is using a 64-bit version of Windows.

Getting Firefly to run in parallel on a single SMP/multicore system is quite straightforward – the mpich_smp.dll library, which is needed for NT-MPICH-SMP implementation, is included in the downloaded Firefly RAR archive and will already be present in the Firefly directory if the full archive was extracted. The only thing that has to be done is to copy and rename the file mpibind.nt-mpich-smp.dll. A job can then be started with the following command:

```
firefly.exe -i c:\jobs\job1.inp -o c:\jobs\job1.out -t c:\ff\ -np 2
```

Here, the -i and -o switches give the input and output file names. The -t command line switch specifies a directory in which Firefly will place its temporary files. To be more specific, Firefly will create a separate direc-

tory in c:\ff\ for each instance of Firefly in which the temporary files of that instance will be placed. It should hereby be noted that it is possible to run multiple jobs in a single directory without any filename conflicts. Formatted data files such as the PUNCH file will be placed in the first of these automatically created directories. Finally, the -np switch gives the amount of instances of Firefly that should be started.

When your computer system has more than one physical hard drive, it is advisable to let Firefly use all of them in order to spread the I/O workload. For such a case, you can explicitly specify working directories as follows:

firefly.exe DIR_0 DIR_1 DIR_2 ... DIR_N -np number_of_cpu_cores_to_use

where DIR_0 is the master directory (where the formatted data files will be created, and where the input file is assumed to be). Prior to execution, all of the directories specified should be manually created. As an example, when launching a single job on two cores and two separate hard disks, one could write:

firefly.exe d:\mydir\wrk0 "e:\my dir\wrk1" -np 2

Command line switches such as -i and -o should precede the list of directories. It is also possible to use relative paths; these will be relative to the master directory.

Detailed instructions for setting up Firefly with other MPI implementations are not included in this documentation as the documentation for each MPI implementation itself is already a good source of information (we recommend reading this before attempting to setup up Firefly in an MPI environment). However, some general tips can be given for a few MPI implementations:

WMPI 1.3

The installation package for this implementation is distributed with Firefly and is one of few that supports Windows 98 and Me. After installation, the cvwmpi.dll file should be somewhere in your path, and the wp4 daemon (wmpi_daemon.exe, Windows 98/Me) or service (wmpi_service.exe, Windows NT/2000,XP, etc...) has to be running on all systems. Note, the -p4dialog option is currently not supported by the Firefly and that this will not be changed in the future. The simplest command line for executing Firefly with WMPI 1.3 is as follows:

FIREFLY.EXE DIR_0 DIR_1 DIR_2 ... DIR_N < wp4 options >

where “wp4 options” are optional wp4-specific options (see the WMPI documentation for the list). For example, you can use something like the following:

firefly.exe d:\mydir\wrk0 e:\mydir\wrk1 "f:\my dir\wrk2" -p4gm 10000000

In this example, the -p4gm 10000000 option sets the size of the global memory used by the WMPI libraries and the wp4 device to 10 MB.

Finally, it's a good idea to look at the more detailed instructions written by Prof. Ernst Schumacher at http://www.chemsoft.ch/qc/pcgamess.htm

NT-MPICH

Setup requires the installation of the 32-bit NT-MPICH package (the part of MP-MPICH), including MPI dynamic link libraries and cluster management service, on all computers that will be used to run Firefly in parallel. Also create an appropriate machinefile (machines.txt) following directions in the NT-MPICH documentation. Afterwards, a Firefly job is executed with mpiexec. A simple command line is as follows:

mpiexec.exe < mpiexec options > FIREFLY.EXE DIR_0 DIR_1 DIR_2 ... DIR_N

An example:

mpiexec.exe -n 3 -account local/administrator C:\FIREFLY\FIREFLY.EXE -o C:\FIREFLY\MP2LARGE\mp2.out C:\FIREFLY\MP2LARGE D:\FIREFLY\MP2LARGE E:\FIREFLY\MP2LARGE

MPICH.NT

Setup requires the installation of the 32-bit MPICH.NT package on all computers to be used to run Firefly in parallel. After installation, an appropriate config file has to be created. Below is an example:

```
exe C:\FIREFLY\FIREFLY.EXE
args -o C:\WORK\chk.out D:\FIREFLY\1 C:\FIREFLY\1 C:\FIREFLY\2
hosts
P4 1
DUATH 2
```

In this example, D:\FIREFLY\1 must exist on host P4, while C:\FIREFLY\1 and C:\FIREFLY\2 must exist on host DUATH2 prior to execution of the Firefly job. The job can be executed with the mpirun program.

Finally, some general advice that applies to all MPI implementations can be given. First of all, it is important to keep in mind that, when running Firefly in parallel, it is possible to experience performance degradation due to simultaneous I/O operations. In such a case, the use of a high-quality RAID setup, or the use of a separate physical disk for each instance of Firefly can help. If the problem persist, one solution is to use direct computation methods which require much less disk I/O.

It should be noted that the default value for AOINTS is DUP. This is probably optimal for low-speed networks (10 and 100 Mbps Ethernet). On the other hand, for faster networks and SMP systems the optimal value could be AOINTS=DIST which distributes the AO integral file across all nodes. One can change this behavior through the AOINTS keyword in the $SYSTEM group.

In the case of MPI-related problems, there are four keywords in the $SYSTEM group which might be of help. These are MXBCST, MPISNC, MXBNUM, and LENSNC. For a description of these keywords, please see the list with keywords. It

is recommended not to modify the default values unless you are absolutely sure that this is needed.

Windows and CUDA

Documentation on the use of CUDA under Windows OSs will be provided in the future. Some information on this topic can be found in the forums on the Firefly website.

Linux

As we have seen in the previous section, the Windows version of Firefly can be made compatible with a certain MPI implementation by copying the appropriate binding DLL into the main Firefly directory. Things however work a bit different for the Linux version of Firefly as here the compatibility with a certain MPI type is incorporated in the main Firefly executable. Therefore, an individual download is available for each MPI implementation. Additionally, for the MPICH version, one can choose between statically and dynamically linked libraries as well as between rsh and ssh supporting libraries.

Getting Firefly up and running in serial mode under Linux is very straightforward. The easiest is to download and decompress the Linux MPICH version that is statically linked with MPICH and dynamically linked with other libraries, as this version does not require an MPI implementation to be installed for serial runs. After decompression, the main firefly binary has to be made executable with the following command:

```
chmod a+x ./firefly
```

This is all that is required for executing Firefly in serial mode. Running multiple, separate jobs is possible. Note that with Firefly 8.0.0, it is possible to run multiple jobs in a single directory without any filename conflicts.

When executing Firefly, the program will look for input in a file called "input" in the directory which contains the firefly executable, though it is also possible to specify an input file with the “-i” command line option. By default, the output is written to the “stdout” device. Therefore, if one would like the output to be saved to a file, an I/O redirection has to be used, for example:

```
./firefly >test.out 2>&1
```

This will cause the output to appear in the file test.out. The addition of “2>&1” will cause some severe error messages also to be send to test.out (Firefly high-level error messages will be written to test.out anyway). Additionally, it is also possible to tell Firefly to write the output to a specific file by using the “-o” command line option:

```
./firefly -o test.out
```

Linux MPI implementations

For Linux, Firefly binaries are available for the following MPI implementations. Note that all fully dynamically linked versions are NPTL-based.

- MPICH (using ssh as remote shell by default), statically linked with MPICH and dynamically linked with other libraries.
- MPICH (using rsh as remote shell by default), statically linked with MPICH and dynamically linked with other libraries.
- MPICH, fully dynamically linked
- MPICH2, dynamically linked
- LAM/MPI version 7.1.X, dynamically linked
- OpenMPI version 1.4.X, dynamically linked
- OpenMPI version 1.6.X, dynamically linked
- MVAPICH, dynamically linked
- MVAPICH2, dynamically linked
- MPICH-MX, dynamically linked
- MPICH-GM, dynamically linked
- Intel MPI, dynamically linked
- Scali MPI, dynamically linked
- Parastation, dynamically linked
- Infinipath MPI, dynamically linked
- HP-MPI, dynamically linked
- Platform MPI, dynamically linked

The MPICH binaries exist in two versions, namely one statically linked with MPICH library only, and one fully dynamically linked. All other binaries are fully dynamically linked and you therefore need to compile and/or install the particular MPI implementation you want to use with support of the shared libraries enabled. Note that because all the currently available Firefly versions are 32-bit executables, so you need 32-bit shared libraries even when using a 64-bit system!

In addition to the above MPI implementations, Firefly has also been successfully linked and tested with other MPI implementations, including LAM/MPI v6.5.9, INTEL MPI v1.x-2.x, MPICH-GM (statically linked), MVAPICH/libvapi, etc. If you would like to obtain Firefly binaries linked with these (or other) MPI implementations, please register for the MPICH version of Firefly, then contact us to obtain customized binaries.

Detailed instructions for setting up Firefly with these MPI implementations are not included in this documentation as the documentation of the MPI implementation itself is already a good source of information (we recommend reading this before attempting to setup up Firefly in an MPI environment). However, some general pointers can be given for the statically linked MPICH version of Firefly, which is probably the easiest to set up for a single multicore node as it does not need the installation of MPICH.

The first step is to unpack the statically linked MPICH version of Firefly into a directory of your choosing. Let’s assume its path is /home/alex/firefly. Next, after unpacking Firefly, you will need to make a procgroup file (filename: “procgrp”) in Firefly’s main directory. Details on how to do this are in the MPICH documentation. Unfortunately, since the

inception of MPICH2, the original MPICH documentation is somewhat hard to find. Pedro Silva is therefore gracefully hosting an old MPICH manual on

http://www2.ufp.pt/~pedros/procgroup.htm

For a single node with only 2 CPU cores, the procgroup file can be kept simple. It only has to contain the string “local 1”. If the single node has 4 cores, the string should be “local 3”. Etcetera.

Then, assuming your input file is in /home/alex/firefly, Firefly can be executed as follows:

/home/alex/firefly/firefly -r -f -p -stdext -i /home/alex/test.inp -o /home/alex/test.out -ex /home/alex/firefly -t /tmp/scratch/test -p4pg /home/alex/firefly/procgrp &

Finally, it is recommended to install the “cleanipcs” script, which is a part of MPICH1 distribution but which can also be found separately on the web after some googling (this script is also included into some Linux distributions).

For running Firefly on an Infiniband interconnect based cluster, we recommend using MVAPICH or recent versions of Intel MPI.

Please see the Windows MPI section for some general advice that applies to all MPI implementations.

Linux and CUDA

Documentation on the use of CUDA under Linux-based OSs will be provided in the future. Some information on this topic can be found in the forums on the Firefly website.

Installing Firefly on an InfiniBand network 64-bit Linux cluster with Intel MPI v. 3.x

Because of the popularity of Linux computing clusters with InfiniBand interconnect, this section contains instructions specifically on how to set up Intel MPI version of Firefly on such a cluster. In the following we assume that Intel MPI is already installed on the cluster.

To get maximum performance of Firefly on a cluster with InfiniBand interconnect, we need to configure our system and the Firefly to use Infiniband. In addition, some parts of Firefly will benefit if IP over InfiniBand (IPoIB) is enabled and functioning. Keep in mind that 32-bit libraries are required to setup Firefly on 64-bit cluster. Most clusters use a Linux-based OS. Therefore, all information below corresponds to the Linux-based OS. Requirements:

- A cluster with preinstalled Linux OS.
- A high speed adapter InfiniBand installed on it.
- Intel MPI v. 3.x

If you are not sure whether Infiniband adapters are installed on your cluster, type at the command prompt (commands to type are preceded by a '$' sign):

$ /sbin/ifconfig -a

If you see a string beginning with "ib" then you have IB installed. In the case where several devices are installed on the cluster (ib0, ib1 ..), check all of them carefully. Only one device should normally be marked as active. Also, we need to check whether or not 32-bit dat and dapl libraries are installed. These libraries are used by Intel MPI to communicate with the IB device.
All examples and outputs below pertain to a particular cluster. The outputs you get may differ in some details, and so may the scripts and configuration files to use in your environment.

```
$ /sbin/ldconfig -p | grep dat
libicudata.so.34 (libc6,x86-64) => /usr/lib64/libicudata.so.34
libdat.so.1 (libc6,x86-64) => /usr/lib64/libdat.so.1
libdat.so.1 (libc6) => /usr/lib/libdat.so.1
libdat.so (libc6,x86-64) => /usr/lib64/libdat.so
libdat.so (libc6) => /usr/lib/libdat.so
libboost_date_time.so.1.33.1 (libc6,x86-64) =>
/usr/lib64/libboost_date_time.so.1.33.1
$
```

As we can see, 32-bit and 64-bit dat libraries are installed on the cluster. As the system-wide settings of Intel MPI installed on a 64-bit cluster are to use 64-bit libraries, we need to override the defaults and force Intel MPI to use 32-bit dapl libraries. In order to achieve this, we need to create our own dat configuration file "dat.conf". This file contains a user defined list of dapl providers for the IB adapter and can be created anywhere within a user's folder. It is good idea to create user's "dat.conf" file based on the system wide one.

```
$ cp /etc/dat.conf ~/
$ cat ~/dat.conf
# This is example of the dat.con file
OpenIB-cma u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 "ib0 0" ""
OpenIB-cma-1 u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 "ib1 0" ""
OpenIB-cma-2 u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 "ib2 0" ""
OpenIB-cma-3 u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 "ib3 0" ""
OpenIB-bond u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 "bond0 0"
""
$
```

The first word of the strings ("OpenIB-cma", "OpenIB-cma-1", etc.) is a name of a dapl provider. Now, we need to find our device (remember the output from "ifconfig" above). In our case it should be ib0. A part of the string "libdaplcma.so.1" is a name of the library that should be used by IB. Of course, on a 64-bit cluster it is a 64-bit library by default. We need to redefine that library and make sure it points out on 32-bit library.

Now, we need to check whether or not the 32-bit library is installed on our cluster:

$ /sbin/ldconfig -p | grep libdaplcma
libdaplcma.so.1 (libc6,x86-64) => /usr/lib64/libdaplcma.so.1
libdaplcma.so.1 (libc6) => /usr/lib/libdaplcma.so.1 # 32-bit library
libdaplcma.so (libc6,x86-64) => /usr/lib64/libdaplcma.solibdaplcma.so
(libc6) => /usr/lib/libdaplcma.so # 32-bit library

Libraries that do not contain *x86-64* are 32-bit. Based on that, we conclude that we have 32-bit libraries installed. Now we need to change the string "libdaplcma.so.1" in our "dat.conf" file to "/usr/lib/libdaplcma.so.1”

The new file dat.conf looks as follows:

$ cat ~/dat.conf
OpenIB-cma u1.2 nonthreadsafe default /usr/lib/libdaplcma.so.1 dapl.1.2
"ib0 0" ""

To start Firefly *via* PBS, we need to add a couple of extra strings to a pbs script. We need to use the 32-bit version of the Intel MPI libraries. In the case they are not installed system-wide, we need to inform the dynamic loader where to look for them. You need to know where exactly they are installed. You may ask your system administrator or use the “locate” or “find” commands.

Let’s assume we found that the 32-bit libraries are installed here:
/opt/intel/ict32/impi/3.2.1.009/lib

Now, we need to modify the LD_LIBRARY_PATH environment variable accordingly (note the line below is a single wrapped line):

export LD_LIBRARY_PATH=/opt/intel/ict32/impi/3.2.1.009/lib:$LD_LIBRARY_PATH

To call Intel MPI's mpirun/mpiexec we need to add the directory "bin" within the Intel MPI installation to the PATH environment variable:

export PATH=/opt/intel/ict32/impi/3.2.1.009/bin:$PATH

By default, the dat layer uses a system wide dat configuration file such as "/etc/dat.conf". We can override these settings by defining a DAT_OVERRIDE environment variable:

export DAT_OVERRIDE=$HOME/dat.conf

Now it points to our version of the dat.conf file.

Below you will find a sample PBS script that can be used to run Intel MPI version of Firefly over an InfiniBand network.

```
#!/bin/sh -x
#PBS -N jobname
#PBS -l nodes=3:ppn=8 # we are going to use 3 nodes and
                      # 8 CPU's on each node
#PBS -l mem=500mb
#PBS -l walltime=6:00:00
#PBS -e jobname.e
#PBS -o jobname.o

### COMMON SETTINGS ###
export IMPI_HOME=/soft/intel/ict32/impi/3.2.2.006
export PATH=$IMPI_HOME/bin:$PATH
export LD_LIBRARY_PATH=$IMPI_HOME/lib:$LD_LIBRARY_PATH
export TMP_DIR=/scratch1/$USER/$PBS_JOBID
export FFHOME=$HOME/bin
export WORK_DIR=$PBS_O_WORKDIR

#### Intel MPI ####
# I_MPI_DEVICE=< device >:< provider >
# see user manual for Intel MPI
# "rdssm" - Combined sockets + shared memory + DAPL
# (for clusters with SMP nodes and RDMA-capable network fabrics)
# Provider is "OpenIB-cma" as defined in our  dat.conf file.
export I_MPI_DEVICE="rdssm:OpenIB-cma"
# Just in case, let us setup extra output information for IntelMPI
# This should not however be used in production runs
export I_MPI_DEBUG=10
#
# Below we need to add appropriate commands to start Firefly.
# It is recommended to read the Quick Start guide for IntelMPI.


mkdir $TMP_DIR
cd $WORK_DIR

mpiexec -n $NCPUS $FFHOME/firefly -r -f -p -stdext -ex $FFHOME -i
$WORK_DIR/filename.inp -o $WORK_DIR/filename.out -b $WORK_DIR/basis.lib -t
$TMP_DIR
# or
mpirun -np $NCPUS $FFHOME/firefly -r -f -p -stdext -ex $FFHOME -i
$WORK_DIR/filename.inp -o $WORK_DIR/filename.out -b $WORK_DIR/basis.lib -t
$TMP_DIR

cd $WORK_DIR && rm -rf $TMP_DIR
rm -rf $TMP_DIR.*
```

The script can be queued by the command:

```
$ qsub myscript.pbs
```

The next step is to obtain the IP address and network mask of the ib0 interface, which will be used to direct the P2P interface to use fast IPoIB network. Note that this is usually not required on modern clusters.

Start by typing

$ /sbin/ifconfig -a

Go to the part "ib0" and look at the strings "inet addr … " and "Mask …." These should be something like:

"inet addr:192.168.5.5 Mask:255.255.0.0"

or

```
$ cat /etc/sysconfig/network/ifcfg-ib0 | grep IPADDR IPADDR=192.168.5.5
$
```

Now we first need to convert the IP address and Mask to hexadecimal. After that, we need to perform a logical AND of the IP address with the Mask. The result is the network address of the IpoIB network. To speed up P2P, we need to put extra settings directly into Firefly INPUT file:

```
 $P2P BIND=.t. NET=<hexadecimal value of the IPoIB network address>
  MASK=<hexadecimal value for the IPoIB network mask> $END
 $SYSTEM MXBCST=-1   $end
 $MPI MXGSUM=1048576 $end
 $MPI MNPDOT=1000000 $end
```

Note that you need to specify leading zeros, if any (*e.g.* net=0a100000 instead of net=a100000).

Command line options

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

All Firefly versions can take optional command line arguments. The supported format of command line is as follows:

firefly [optional command line options in any order] [optional list of working directories]

The Firefly accepts both -option and /option syntax. All options are case-insensitive. A list of the most widely used command line options is given below:

The following list is currently incomplete and will be completed in the future.

-i <filename>	Specifies input file to use. Filename is the complete absolute name of the file to be used as input. If no path is specified, Firefly will assume the file resides in the initial working directory from which Firefly was launched. The file will be read in, optionally preprocessed (in the case it contains any preprocessing directives), and result will be copied into the assigned working directory of the master Firefly process using the plain "INPUT" filename. If this option is omitted, the file named "INPUT" will be searched for in the assigned working directory of the master Firefly process, and used directly as the input file to Firefly.
-o <filename>	Specifies output file to use. Filename is the complete absolute name of the file to be used as output. If this option is omitted, output will be sent to standard output (stdout) device.
-b <path or filename>	Specifies the complete absolute location and/or name of the external basis set library file. If the filename is given, it will be used to read in the external basis sets. If the filename was not found, Firefly tries to interpret it as path and looks for file called "BASIS.LIB" at this location.
-t <path>	Directs the Firefly to use the specified path as the template to create per-instance unique temporary working directories to be used to store all the intermediate working files.
-ex <path>	Linux specific: directs the Firefly to copy runtime extension files (pcgp2p.ex, fastdiag.ex, and optionally p4stuff.ex) into its working directory(ies) from the location specified by path.
-r	Directs the Firefly to remove all the temporary scratch files opened using FSF routines at the end of the job.
-f	Forces the Firefly execution in the presence of the old PUNCH or IRCDATA files. Windows specific: also forces execution if the assigned working directory is on the network drive or is the root directory of the volume (*e.g.*, C:\). This option can also be given three times followed by an -o option (i.e., "-f -f -f -o OUTPUT.out") in order to force Firefly to overwrite the output file specified with -o.
-p	Redirects all the text output files (PUNCH, IRCDATA, etc...) from working directory to the directory where the main output file resides.
-stdext	Changes naming convention used for PUNCH and IRCDATA files. Provided that output filename was set using -

	o option, the last extension of this name, if any, will be removed and then .dat/.irc extension will be added to the end to form the filename to use instead of plain PUNCH or IRCDATA names.
-daf <mode>	Specifies the mode used to work with primary dictionary file DICTNRY. Mode is a number, it can be one of 0, 1, or 2. <mode> = 0 Default strategy is used to handle requests to DICTNRY file. <mode> = 1 Increases the size of messages used to broadcast contents of DICTNRY file to the slave processes. This may reduce overhead of communications on some interconnects and/or for some MPI implementations resulting in better performance and scalability. <mode> = 2 Allows DICTNRY file to be replicated over all slaves. This mode completely eliminates overhead caused by all DICTNRY-related communications by the cost of performing some extra I/O. Typically, this is the fastest and the best scalable mode.
-ncores <number>	Allows one to override the automatically detected number of physical cores per CPU. Useful when running the Firefly on buggy processors, or in the virtualized environment like Hyper-V etc... . Passing correct value (in the case it was not properly detected automatically) will generally improve the Firefly performance.
-nthreads <number>	Allows one to override the automatically detected number of active logical cores per single physical core. Useful when running the Firefly on buggy processors, or in the virtualized environment like Hyper-V etc... . Passing correct value (in the case it was not properly detected automatically) will generally improve the Firefly performance.
-lp	Windows specific: Allows Firefly to use large memory pages. The hardware and OS must support large pages, and user account must have enough rights to allow Firefly to allocate large pages. Otherwise, this option will be ignored. Running in large pages mode, Firefly prints information message on their use at the beginning of its output.
-nocheck	Disables validity check of the command line arguments (*e.g.*, check for valid names of files and directories, etc.) passed to the Firefly.
-nompi	Forces purely sequential execution avoiding any MPI calls even if initially launched in parallel.
-v	Directs the Firefly to trace each MPI call on each node. This will create additional pseudo- output

	files for all slaves.
-prof	Directs the Firefly to gather some real-time profiling statistics. This option is useful while fine-tuning the performance and provides some insight on the possible performance bottlenecks.
-xp or -xp=<number>	Given without <number>, this option enables the XP parallel mode of execution. With <number>, it enables the extended XP parallel mode of execution, where <number> defines the amount of processes in each group. This option may also be written in the form of -xp:<number>
-legacy	Causes output to be more in the style of older version of Firefly (useful when a third-party program has difficulties parsing Firefly 8.0.0 output).
-run	Forces the job to be an actual run, regardless of the value of EXETYP in $CONTRL.
-check	Forces the job to be a check run, regardless of the value of EXETYP in $CONTRL.
-prealloc:<number>	Instructs Firefly to preallocate <number> MW of memory in the virtual address space at the very beginning of the job initialization. Such behavior is different from Firefly's normal behavior, which is to allocate memory *after* MPI initialization. This can be of help when trying to run Firefly with a large amount of memory (400 – 500 MW) as the memory allocated has to be a single continuous address range, but MPI initialization might cause the virtual address space to be fragmented. There is no guarantee that the pre-allocation will be successful though.

The optional list of working directories, if any, must follow the list of command-line options. It has no effect if the -t < path > option was already specified. However, if this option is not given, one must provide the list of working directories when running the Firefly in parallel mode. The format of this list is very simple:

$$DIR_0\ DIR_1\ DIR_2\ \ldots\ DIR_{n-1}$$

One must specify exactly the same number of directories as the total number of Firefly instances is. For example, to run the Firefly in parallel on 16 cores, one has to specify sixteen directories. The master instance of the parallel Firefly process will use DIR_0, the first slave will use DIR_1, and so on. Unlike temporary directories created using -t < path > option, the explicitly passed directories must exist and have proper access rights prior to the Firefly execution, as they will not be created automatically! In most cases, it is much more convenient to use -t < path > syntax; however, there are some situations when the old-style "directories list" syntax is very helpful.

Creating an input file

Input file structure

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Input to Firefly may be in upper or lower case. There are three types of input groups in Firefly:

1. A pseudo-namelist, free format, keyword driven group. Almost all input groups fall into this first category.

2. A free format group which does not use keywords. The only examples of this category are $DATA, $ECP, $POINTS, and $STONE.

3. Formatted data. This data is never typed by the user, but rather is generated in the correct format by some earlier Firefly run.

All input groups begin with a $ sign in column 2, followed by a name identifying that group. The group name should be the only item appearing on the input line for any group in category 2 or 3.

All input groups terminate with a $END. For any group in category 2 and 3, the $END must appear beginning in column 2, and thus is the only item on that input line.

Type 1 groups may have keyword input on the same line as the group name, and the $END may appear anywhere.

Because each group has a unique name, the groups may be given in any order desired. In fact, multiple occurrences of category 1 groups are permissible.

Most of the groups can be omitted if the program defaults are adequate. An exception is $DATA, which is always required. A typical free format $DATA group is

```
 $DATA
STO-3G test case for water
CNV      2

OXYGEN       8.0
    STO  3

HYDROGEN     1.0    -0.758       0.0     0.545
```

```
              STO  3

            $END
```

Here, position is important. For example, the atom name must be followed by the nuclear charge and then the x,y,z coordinates. Note that missing values will be read as zero, so that the oxygen is placed at the origin.
The zero Y coordinate must be given for the hydrogen, so that the final number is taken as Z.
The free format scanner code used to read $DATA is adapted from the ALIS program. Note that the characters ;>! mean something special to the free format scanner, and so use of these characters in $DATA and $ECP should probably be avoided.

Because the default type of calculation is a single point (geometry) closed shell SCF, the $DATA group shown is the only input required to do a RHF/STO-3G water calculation.

* * *

As mentioned, the most common type of input is a namelist-like, keyword driven, free format group. These groups must begin with the $ sign in column 2, but have no further format restrictions. You are not allowed to abbreviate the keywords, or any string value they might expect. They are terminated by a $END string, appearing anywhere. The groups may extend over more than one physical card. In fact, you can give a particular group more than once, as multiple occurrences will be found and processed. We can rewrite the STO-3G water calculation using the keyword groups $CONTRL and $BASIS as

```
 $CONTRL SCFTYP=RHF RUNTYP=ENERGY $END
 $BASIS GBASIS=STO NGAUSS=3 $END
 $DATA
STO-3G TEST CASE FOR WATER
CNV 2

Oxygen       8.0     0.0         0.0     0.0
Hydrogen     1.0    -0.758       0.0     0.545
 $END
```

Keywords may expect logical, integer, floating point, or string values. Group names and keywords never exceed 6 characters. String values assigned to keywords never exceed 8 characters. Spaces or commas may be used to separate items:

```
$CONTRL MULT=3 SCFTYP=UHF,TIMLIM=30.0 $END
```

Floating point numbers need not include the decimal, and may be given in exponential form, i.e. TIMLIM=30, TIMLIM=3.E1, and TIMLIM=3.0D+01 are all equivalent.

Numerical values follow the FORTRAN variable name convention. All keywords which expect an integer value begin with the letters I-N, and all keywords

which expect a floating point value begin with A-H or O-Z. String or logical keywords may begin with any letter.

Some keyword variables are actually arrays. Array elements are entered by specifying the desired subscript:

```
$SCF NO(1)=1 NO(2)=1 $END
```

When contiguous array elements are given this may be given in a shorter form:

```
$SCF NO(1)=1,1 $END
```

When just one value is given to the first element of an array, the subscript may be omitted:

```
$SCF NO=1 NO(2)=1 $END
```

Logical variables can be .TRUE. or .FALSE. or .T. or .F. The periods are required. Logical variables may also be input as 1 or 0.

The program rewinds the input file before searching for the namelist group it needs. This means that the order in which the namelist groups are given is immaterial, and that comment cards may be placed between namelist groups.

Furthermore, the input file is read all the way through for each free-form namelist so multiple occurrences will be processed, although only the LAST occurrence of a variable will be accepted. Comment fields within a free-form namelist group are turned on and off by an exclamation point (!). Comments may also be placed after the $END's of free format namelist groups. Usually, comments are placed in between groups,

```
$CONTRL SCFTYP=RHF RUNTYP=GRADIENT $END
--$CONTRL EXETYP=CHECK $END
$DATA
molecule goes here...
```

The second $CONTRL is not read, because it does not have a blank and a $ in the first two columns. Here a careful user has executed a CHECK job, and is now running the real calculation. The CHECK card is now just a comment line.

* * *

The final form of input is the fixed format group.
The formatted groups are $VEC, $HESS, $GRAD, $DIPDR, and $VIB. Each of these is produced by some earlier Firefly run, in exactly the correct format for reuse. Thus, the format by which they are read is not documented in this manual.

* * *

Each group is described in the 'Input Description' chapter.

Fixed format groups are indicated as such, and the conditions for which each group is required and/or relevant are stated.

Chemical control data

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The most important settings are provided through the $CONTRL group, which can be used to specify the calculation type, wave function type, multiplicity, charge, etc.

Computer related control data

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Settings that pertain to the computer operations are mostly part of the $SYSTEM, $P2P, $LP2P, and $SMP groups. A few very specific settings can be found in various other groups (for example, I/O control switches for the MP2 program are part of the $MP2 group).

Formatted input sections

A run can produce one or more formatted input groups, depending on the type of run and the settings used during the run. Below is an overview of formatted input groups which are used by Firefly. Note that some groups allow one to use the first line as a title card, while other groups are very strict about whether such a group can be used.

Name	Content	Location	Title card?
$VEC	Orbitals	PUNCH and MCQD* files	Allowed
$HESS	Force constant matrix	PUNCH file	Required
$GRAD	Gradient vector	PUNCH file	Required
$DIPDR	Dipole derivative tensor	PUNCH file	Not allowed
$ALPDR	Alpha polarizability	PUNCH file	Required
$VIB	Restart data for a numerical RUNTYP=HESSIAN and RUNTYP=RAMAN runs	PUNCH and IRCDATA files	Not allowed
$CISVEC	CIS orbitals	PUNCH file	Allowed
$TDVEC	TDHF and TDDFT orbitals (from a CITYP=TDHF/TDDFT run)	PUNCH file	Allowed
$TWOEI	Transformed two-electron coulomb and exchange integrals	PUNCH file	Not allowed

Input checking

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Because some of the data in the input file may not be processed until well into a lengthy run, a facility to check the validity of the input has been provided. If EXETYP=CHECK is specified in the $CONTRL group, Firefly will run without doing much real work so that all the input sections can be executed and the data checked for correct syntax and validity to the extent possible. The one-electron integrals are evaluated and the distinct row table is generated. Problems involving insufficient memory can be identified at this stage. To help avoid the inadvertent absence of data, which may result in the inappropriate use of default values, Firefly will report the absence of any control group it tries to read in CHECK mode. This is of some value in determining which control groups are applicable to a particular problem.

The use of EXETYP=CHECK is HIGHLY recommended for the initial execution of a new problem.

Input preprocessing

Starting from version 7.1, Firefly has a useful preprocessing function that retrieves parts of the input from one or more external files. The idea behind this function is to keep the input file as small as possible while giving flexible and versatile control over large blocks in the input file, examples of such blocks being the specification of the basis set and effective core potential (ECP), the orbital data (the $VEC group), the Hessian data (the $HESS group), etc. In order to activate the preprocessing function, Firefly should be run with the -i <filename> command line parameter, where <filename> is the name of the main input file.

In the main input, the notation "@filename" is used to retrieve the contents of a specific external file and insert its content as a block into the input file. Here, "filename" is a plain text file containing the information of interest. By default, Firefly will look for the specified file in the working directory of Firefly's master process. It is, however, possible to specify additional directories with the "#libdir ./directory" directive. To clarify this, let us consider the following example:

```
#libdir /home/ff/ECP
#libdir /home/ff/basis
#libdir /home/ff/job1/guess
 $CONTRL DFTTYP=PBE0 RUNTYP=OPTIMIZE SCFTYP=RHF EXETYP=RUN
         COORD=UNIQUE NZVAR=39 ICHARG=0 MULT=1 MAXIT=500
         ECP=READ D5=.T. $END
 $CONTRL EXETYP=CHECK $END
 $ZMAT   DLC=.T. AUTO=.T.
         NONVDW(1)= 1,2 1,15
         IFZMAT(1)= 3,15,1,2,11
         SYMREP=-1 $END
```

```
 $SCF    NCONV=5 DIRSCF=.T. $END
 $STATPT NSTEP=100 OPTTOL=0.0001 METHOD=GDIIS $END
 $GUESS GUESS=MOREAD NORB=200 $END
 $SYSTEM TIMLIM=6000 MWORDS=10  $END
 $BASIS EXTFIL=.T. GBASIS=SVP $END
 $DATA
 RhP3NOF9
CN        3

RH  45.0   0.000000000     0.000000000     0.569919578  @Rh_svp.bas
P   15.0   1.414965727     1.465467346    -0.382720422  @P_svp.bas
F   9.0    2.431892386     1.029806465    -1.546074422  @F_svp.bas
F   9.0    0.861955322     2.810425007    -1.062902422  @F_svp.bas
F   9.0    2.477768601     2.138622865     0.615240578  @F_svp.bas
O   8.0    0.000000000     0.000000000     3.566541578  @O_svp.bas
N   7.0    0.000000000     0.000000000     2.392496578  @N_svp.bas
 $END
 $ECP
@Rh_def2_SVP.ecp
P-ECP NONE
P-ECP
P-ECP
F-ECP NONE
F-ECP
F-ECP
F-ECP
F-ECP
F-ECP
F-ECP
F-ECP
F-ECP
O-ECP NONE
N-ECP NONE
 $END
@MO_to_read.txt
```

At the top, the three #libdir preprocessing directives define three directories (*ECP*, *basis* and *guess*) from which Firefly will retrieve all files specified with the '@' marker. These files can be named as desired. In the example, the "basis" directory contains files with basis sets (Rh_svp.bas, P_svp.bas, F_svp.bas etc.), the "ECP" directory contains the file Rh_def2_SVP.ecp with ECP data for the Rh atom, and the "guess" directory contains the file (MO_to_read.txt) with molecular vectors extracted from the PUNCH file of a previous run. Since information from @filename is copied directly into the input file (the inserted text is not preprocessed), special attention should be paid to the format of these files – their contents should fit seamlessly into the input file. In the above example, the files should start immediately with the basis set or ECP definition (no atom name is allowed) and should be terminated by empty line in case of the basis set data. Group names such as $DATA and $BASIS should not be given as they're already in the input file. However, for the file with the MO vectors, the $VEC and $END keywords must be present since these are not yet specified in the input file – the whole block with MO vectors has to be inserted.

Note that the path specified with "#libdir" can be absolute or relative to scratch directory. It is usually recommended to use absolute paths as this is the most convenient.

Performance

Introduction

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

This chapter discusses a few important settings that are important for Firefly's performance (i.e. speed).

64-bit processing support

Though a fully 64-bit version of Firefly is not yet available, the current version of Firefly is able to use some 64-bit CPU instructions. More precisely, the base 32-bit code can call some 64-bit computational kernels for faster processing. The use of 64-bit code is especially important when using newer processors (Intel Core 2 and newer, AMD Barcelona core and newer). We recommend all Firefly users running Firefly on these processors to use a 64-bit operating system in order to allow the use of the 64-bit code.

The use of 64-bit code is enabled through the CALL64 keyword of the $SMP group:

```
 $SMP CALL64=.T. $END
```

In Firefly 8.0.0, this option is enabled by default. For older version of Firefly, it is by default enabled for the Windows binaries, but disabled for the Linux binaries.

The P2P communication interface

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Starting from the PC GAMESS version 6.3, the support of the new proprietary parallel mode communication interface (which is called P2P interface) was implemented as a part of the software. This interface is very flexible and was specifically designed to overwhelm the limitations of MPI & DDI interfaces. The Firefly specific parallel MP2 energy and energy gradient method=1 modules supports P2P communication model. It is expected that in the future more and more computational methods and algorithms in the Firefly will support P2P.
To take advantages of this interface, you need:

1. Firefly running in parallel mode over MPI as usually.

2. The dynamic library in which the P2P interface is implemented. It is called pcgp2p.dll (Win32) or pcgp2p.ex (Linux). The Firefly distribution contains the library that is suitable for your OS. It should be placed into the Firefly home directories on each computing node in the case of Windows OS and into the Firefly working directories (note not into the Firefly home directories, or /usr/lib, etc...) on each node in the case of Linux OS. It should be renamed to be all lowercase (Linux).

3. You should activate the P2P interface adding $P2P P2P=.T. $END to the input file.

Starting from the PC GAMESS v. 6.4, the DLB (dynamic load balancing) functionality was added to P2P interface. To activate DLB, add the following line to your input:

```
 $P2P P2P=.T. DLB=.T. $END
```

Many of the parallel-aware Firefly parts transparently use DLB over P2P interface if DLB is enabled, including 2-e part of direct SCF and DFT, etc...

For some jobs you may find that extended DLB model (XDLB) results in slightly better performance than standard DLB model. XDLB can be activated by specifying:

```
 $P2P P2P=.T. XDLB=.T. $END
```

The difference between these DLB and XDLB modes is in the additional thread(s) created by Firefly to handle P2P requests. In DLB mode, a single additional thread is created which operates in multiplexed mode, serving both P2P requests and DLB requests In XDLB mode, two additional threads are created, one of which serves to handle DLB requests exclusively. XDLB mode requires more system and program resources but may result in better load balancing. This is most frequently observed for parallel MP2 METHOD=1 runs.

Details on the MP2 energy code and P2P interface implementation in the Firefly can be found here. Information on how the DLB affects performance can be found on this page.

Windows specific:
The file pcgp2psm.dll contains implementation of the P2P interface that is specific to shared memory SMP/multicore systems. If you run Firefly on a standalone SMP/multicore system, rename this file to pcgp2p.dll and replace the default P2P library in the Firefly home directory. This will provide better performance than the default library (which uses TCP/IP rather than shared memory).
For better efficiency of shared memory implementation of P2P, it is recommended to use the following additional P2P settings in the input files:

```
 $P2P MXBUF=2048 $END
```

Below is the sample Firefly input file which uses P2P for DLB-driven direct SCF calculations in parallel mode:

```
 $CONTRL SCFTYP=RHF RUNTYP=ENERGY UNITS=ANGS $END
 $SYSTEM TIMLIM=600 MEMORY=3000000 $END
! to activate P2P inteface and DLB:
 $P2P P2P=.T. DLB=.T. $END
 $BASIS  GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
! to speed up Huckel guess:
 $GUESS  GUESS=HUCKEL KDIAG=0 $END
 $SCF DIRSCF=.T. $END
 $DATA
6-31G*//RHF/3-21G* Silacyclobutane
CS

SILICON    14.0    -0.081722       1.055710
CARBON      6.0    -0.081722      -0.395331       1.217568
CARBON      6.0     0.319935      -1.329102
HYDROGEN    1.0    -1.222554       1.998369
HYDROGEN    1.0     1.168317       1.848237
HYDROGEN    1.0     0.604981      -0.419640       2.052727
HYDROGEN    1.0    -1.077445      -0.641554       1.572232
HYDROGEN    1.0     1.388834      -1.500162
HYDROGEN    1.0    -0.184517      -2.285408
 $END
```

The XP and extended XP parallel modes of execution

Firefly version 8.0 and above supports two new parallel modes of execution, namely the standard eXtreme Parallel (XP) and extended XP modes. The idea behind these modes is to efficiently utilize the multi-level parallelism inherent to many typical QC calculations by splitting the entire job into quasi-independent pieces which can be processed at high level in parallel. In the standard XP mode, each instance of the entire parallel Firefly super-process most of the time acts as if it were a separate master process working on its own task, in serial or using multi-threading. In the extended XP mode, there are several groups of processes, each group consisting of its own local master and slave processes. The global master process is at the same time the local master of the first group. The members of each group are working together on the task or tasks which are specifically assigned to them. For instance, in runs involving numerical gradients, the energy at each displaced geometry used in finite differencing can be calculated independently. If the underlying theoretical method is not programmed to run in parallel, it is then natural to use the standard XP mode to run multiple serial energy calculations at once and in parallel. If it is, any of three parallel modes (standard parallel, standard XP, and extended XP) could be used, with extended XP being the most scalable and most highly-performing alternative. For example, if a typical single point energy computation for a particular combination of QC method and model system is scalable to e.g. 64 cores, and the numerical gradient code requires ca. 100 reevaluations of energy, the calculations of numerical gradients in extended XP modes would be scalable up to 64*100=6400 cores and would be extremely fast.

The current implementation of XP modes in Firefly supports two levels of parallelism. This will be changed in the future to support the arbitrary number of parallelism levels provided there will be a need for this. Presently, any job involving numerical gradients can be run in either standard XP or extended XP modes, or both. Other jobs types are not currently allowed to run in XP modes. This limitation will be relaxed in the future by allowing semi-numerical Hessian computations, semi-analytic Raman activities computations, surface scans and some other types of jobs to run in XP modes as well.

The XP or extended XP modes are invoked by running Firefly in parallel using dedicated command line options. There is no way to force XP modes using information in the input file as preparations for these modes of operation need to be done at the very early stages of Firefly's initialization. Namely, to launch Firefly in standard XP mode, use the -xp command line option. Similarly, to launch Firefly in extended XP mode, the -xp:N (or, alternatively, -xp=N) command line options can be used. Here, N is the number of processes belonging to each process group. For instance, launching Firefly on 32 cores using -xp=4 (or -xp:4) command line option will create eight groups of processes, with each group consisting of four separate processes.

Firefly's P2P interface and dynamic load balancing over P2P can be used in any of three parallel modes of execution. Working in the extended XP mode, Firefly supports two levels of P2P communications and dynamic load balancing. The high-level one is global for the entire parallel Firefly process, and serves primarily for the communications between the local masters of separate groups. For instance, high-level dynamic load balancing is used to distribute the high-level jobs between individual process groups. The global P2P interface is controlled by the usual $P2P control group of the input file. Each separate group of processes can be interconnected via its own local P2P interface which is virtually identical to the global one but is limited in its scope to the members of its group only. The behavior of the second-level, local P2P interfaces is controlled using the new $LP2P control group of the input file. The $LP2P input group has exactly the same keywords as the standard $P2P input group.

Utilizing HyperThreading

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

HyperThreading is a CPU technology that causes each physical CPU core to be seen as two virtual cores. For some calculations types, Firefly is able to make use of these extra cores in order to provide a modest speedup. Typically, Firefly will automatically detect the presence of virtual cores and use them in an intelligent manner. However, if one finds that Firefly mistakenly uses more than one virtual core on one physical core without exhausting all physical cores, one can correct this behavior by specifying:

```
$SMP HTTFIX=.F. $END
```

Which disables explicit binding to cores at all. Alternatively, you can use:

```
$SMP HTTPAR=.T. $END
```

to allow the binding of each process to different logical processors for parallel runs.

Finally, in the case of two independent jobs on one SMP system, you should add:

```
$SMP HTTALT=.T. $END
```

to the second input file to resolve binding conflict between two Firefly instances.

CUDA

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The MP4 code in Firefly can be executed using CUDA. CUDA is not supported by any other functionality in Firefly.

The Fastdiag dynamic library

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

The Fastdiag dynamic library (fastdiag.dll for Windows Firefly distributions, fastdiag.ex for Linux distributions) contains fast optimized modern algorithms of symmetric matrix diagonalization and inversion and is intended to improve the performance of initial guess generation, DIIS extrapolation, as well as some other computationally-intensive steps. Windows users should put this library into the folder where the Firefly executables reside, while Linux users should put it into the Firefly working/scratch directories. Under Linux, the name of the file should be all lower-case.

There are three related options in $SYSTEM and $GUESS groups, namely:

$SYSTEM KDIAG= < one of 3,2,1,0,-1,-2 > $END - Controls the system-wide diagonalization routine used.

$SYSTEM NOJAC= < N > $END - Instructs Firefly to never use Jacobi diagonalization for matrices of size NxN and above, even if Jacobi code was explicitly requested.

$GUESS KDIAG= < one of 3,2,1,0,-1,-2 > $END - Controls the diagonalization routine used during initial guess generation.

The values of 3, 2, 1 have the same meaning as in the regular GAMESS (US), the values of 0,-1,-2 are the Firefly specific Fastdiag values.

KDIAG=0 selects very stable, fast and precise diagonalization routine based on the Divide and Conquer (DC) algorithm. However, DC-based code requires large amount of extra memory.

KDIAG=-1 selects a potentially less stable, less precise but even faster diagonalization routine based on the Relatively Robust Representation (RRR) approach. This method requires less memory than DC code.

Finally, KDIAG=-2 selects a combination of RRR and DC methods, which falls back to DC in the cases when RRR fails completely resulting in NANs and INFs. It is usually as fast as KDIAG=-1 but requires as much memory as KDIAG=0. However, as in the most cases results are obtained using RRR approach, KDIAG=-2 is still less precise than purely DC-based kdiag=0.

Note, KDIAG=-1 and KDIAG=-2 options are currently considered as the experimental ones and are intended mainly for large MOPAC jobs. Otherwise, they should not normally be used! The most serious issue one can encounter using RRR-based code is the sporadic program hangs inside RRR-based diagonalization code. This seems to be the intrinsic property of the RRR approach and there does not seem to exist any solution to this problem.

The default value of KDIAG found in $SYSTEM and $GUESS groups is 0 (as -1 and -2 are the experimental options at present), which is reasonable. For better compatibility with GAMESS (US), the default value of NOJAC is -1, meaning that this variable has no effect at all. It is generally recommended to set NOJAC to some small value, e.g., 30 or so, especially if the number of basis functions is large enough.

Note that fast diagonalization routines use extra memory which is not taken into account during check runs! Hence it is recommended to reserve some additional amount of memory for diagonalization routines. For example, if the system of interest has ca. 1000 basis functions, it is a good idea to add about 2.5-3 MW of memory for diagonalization purposes. If the amount of memory is not enough to use fast routines, the slower built-in routine will be called.

Finally, it should be noted that fastdiag is not compatible with and is not used by the generic Pentium Firefly versions.

Fast two-electron integrals code

The ‘fastints’ 2-electron integrals/fock matrix build/integral transformation modules are intended to speed up direct HF/DFT/CIS/TDHF/TDDFT/MCSCF runs. They are presently implemented for direct RHF/UHF/ROHF/CIS/TDHF/TDDFT/MCSCF-type calculations only. The performance gain as compared with standard GAMESS (US)-based direct SCF implementation depends on the particular basis set type and the processor architecture used and usually varies from 50% to 200-400%. The only situation where old integral code can be faster than fastints is in the case of a pure L-shell basis set while using the Pople integral package. Even for the 6-31* basis

set, which contains a relatively large number of L shells per atom, the new code is considerably faster than the old code (mainly due to the presence of d functions), especially on Pentium 4-type processors. Note that in many situations the new direct SCF modules are faster than the corresponding conventional SCF. The precision of the two-electron integrals calculated by the new code is comparable with that of GAMESS (US)'s old INTTYP=HONDO 2-e integral code package. By code design, the time required for Fock matrix formation using new routines depends strictly quadratically on the number of atoms in molecule for sufficiently large molecular systems.

The new code can be run in parallel using both static and dynamic load balancing modes, though the latter is preferred. For very large molecular systems and HF/DFT, the new code can be used in conjunction with the linear scaling QFMM code – see the QFMM section for additional information on QFMM implementation in Firefly. Options specific to MCSCF calculations are documented elsewhere. Generic input is described below.

There are three options related to the new code in the $CONTRL group:

FSTINT=.TRUE./.FALSE. Enables (default)/disables the use of the new direct SCF code.

REORDR=.TRUE./.FALSE. Enables (default)/disables shells reordering for even better direct SCF performance.

GENCON=.TRUE./.FALSE. Enables (default)/disables the use of the special version of the fastints code designed for general contraction (GC) type basis sets. It is mainly intended to dramatically speedup calculations involving large GC-type basis sets like ANO basis sets by Roos et al (the example of pure GC basis sets), and to some degree cc-pVXZ basis sets (which are only partially of GC type), and many others. The code is very efficient, but requires some additional amount of memory and has minor addition computational overhead for setup. It can result in slightly different energies than the standard fastints code using the same value of ICUT and ITOL parameters, and does not improve performance for pure segmented contraction basis sets at all. This is why the gencon code automatically disables itself if the basis set is not of the GC type. At present, it has no effect on QFMM calculations.

Quantum fast multipole method

New modules implementing linear scaling methods based on QFMM were added to the Firefly in order to speed up large-scale direct HF and DFT runs. The QFMM code is partially based on optimized and bugfixed GAMESS (US) QFMM code (refs. 3 and 4 below), as well as on new modules developed at MSU. It is currently implemented for RHF/UHF/ROHF-type calculations only. A CI or MP stage is allowed to be performed after the QFMM calculation stage, and conventional gradients (not QFMM-based) are available. The QFMM code can be run in parallel using both static and dynamic load balancing modes, the latter is preferred in most cases.

QFMM calculations consist of two or possibly three different steps, depending on whether exact HF exchange is required (for HF and hybrid DFT) or not (for pure DFT). These steps are:

1. Calculation of the so-called Coulomb (J) far-field contribution to the Fock matrix. This step is performed using the FMM (fast multipole method) technique. For this step, Firefly uses a set of routines based on the original GAMESS (US) sources which were bugfixed and tuned for better performance.

2. Calculation of the so-called Coulomb (J) near-field contribution to the Fock matrix. This step is performed using two-electron integral and modified direct SCF-like routines. At present, there are two algorithms implemented in Firefly to perform this step. The first one is based on bugfixed and performance tuned GAMESS (US) code, the second approach is completely different and is based on the new fastints code.

3. The so-called linear-scaling exact exchange (K) contribution to the Fock matrix (also known as LEX or linK). This step is also performed using two-electron integral and modified direct SCF-like routines. At present, there are three algorithms implemented in Firefly to perform this step. The first one is based on bugfixed and performance tuned GAMESS (US) code, the second and third approaches are completely different and are based on the fastints code.

The QFMM input in Firefly is compatible with that of GAMESS (US). QFMM is turned on by the logical variable QFMM in the $INTGRL group (its default value is .FALSE., *i.e.*, no QFMM calculations). You must select DIRSCF=.TRUE. in $SCF and SCHWRZ=.TRUE. (default) in $INTGRL if you use this option. Most of the QFMM-related options are controlled by the corresponding $FMM group. Some keywords in the $CONTRL group affect QFMM as well, namely ICUT, ITOL, FSTINT and REORDR. Another keyword affecting the performance of all linear exchange routines is the RCRIT value in the $MOORTH group, which controls the density matrix pruning. If RCRIT is greater than zero, all matrix elements of the density matrices will be set to zero if the distance between two orbital centers is greater than RCRIT. This option can speed up LEX (the routine used to calculate HF exchange terms), but should be used with a caution, especially for conjugated systems, metal clusters, etc. For alkanes, RCRIT=25 a.u. seems to be safe enough. The default is zero.

Note that the default values of the keywords of the $FMM group are quite reasonable, so there is usually no need to alter them.

Some additional comments:

1. Near-field J and linear exchange routines require more CPU time than direct SCF in the case of small and even medium-size systems due to additional logic and computational overhead. Thus, QFMM should be used for large systems only and it is usually a good idea to check what the fastest method is in your particular case.

2. There is no or little use of molecular symmetry during QFMM runs. Thus, direct SCF with fastints code can be faster than QFMM for very large symmetrical systems (like fullerenes, etc.)

3. Time required for QOPS far-field J FMM is usually much smaller than that of near-field J, especially on the first SCF iterations. The time used by LEX is usually comparable with or larger than that of near-field J, especially on the very first SCF iterations. There is some additional overhead in near-field J routines if HF exchange is required as well. Thus, the speedup of pure DFT calculations due to QFMM is more serious than that of HF and hybrid DFT.

4. There is an EXETYP=QFMM option in the $CONTRL group which is used to get the timing statistics of the various QFMM stages during SCF.

Selected QFMM references:

1. E.O.Steinborn, K.Ruedenberg Adv.Quantum Chem. 7, 1-81 (1973)
2. L.Greengard "The Rapid Evaluation of Potential Fields in Particle Systems" (MIT, Cambridge, 1987)
3. C.H.Choi, J.Ivanic, M.S.Gordon, K.Ruedenberg J.Chem.Phys. 111, 8825-8831 (1999)
4. C.H.Choi, K.Ruedenberg, M.S.Gordon J.Comput.Chem. 22, 1484-1501 (2001)
5. C.H.Choi J.Chem.Phys. 120, 3535-3543 (2004)

Output

Main output

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The main output is by default printed onscreen. One can send the output to a file as follows:

```
firefly.exe >test.out 2>&1
```

Alternatively, one can use the –o parameter:

```
firefly.exe –o test.out
```

The amount of information printed can be changed through various keywords in Firefly. The most important of these is the NPRINT keyword in $CONTRL.

The PUNCH file

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The PUNCH file contains various data produced by the calculation. Much of this data is of the formatted type and can be cope/pasted to the input of a second run. Examples of data punched is geometry information, basis set information, orbitals, the force constant matrix, etc.

The IRCDATA file

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The IRCDATA file serves multiple functions. For example, it contains molecular geometries found during an IRC run. Also, it contains restart data from a numerical Hessian calculations and Raman calculations.

The MCQD files

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The MCQD files contain data from (X)MCQDPT2 runs. MCQD63 contains (X)MCQDPT2 MOs, MCQD64 contains information on the CSFs used.

Restart capabilities

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

The program checks for CPU time, and will stop if time is running short. Restart data are printed and punched out automatically, so the run can be restarted where it left off.

At present all SCF modules will place the current orbitals in the punch file if the maximum number of iterations is reached. These orbitals may be used in conjunction with the GUESS=MOREAD option to restart the iterations where they quit. Also, if the TIMLIM option is used to specify a time limit just slightly less than the job's batch time limit, Firefly will halt if there is insufficient time to complete another full iteration, and the current orbitals will be punched.

When searching for equilibrium geometries or saddle points, if time runs short, or the maximum number of steps is exceeded, the updated Hessian matrix is punched for restart. Optimization runs can also be restarted with the dictionary file. See $STATPT for details.

Coordinate types

Introduction

Firefly is capable of working with various types of coordinates. Possible input coordinates are Cartesian coordinates, Hilderbrandt style internals and Z-Matrix internal coordinates (Gaussian and MOPAC style). In addition, Firefly can internally work with 'plain' Z-matrix coordinates, symmetrized Z-matrix coordinates (e.g. natural internals), and delocalized internal coordinates (DLCs).

The input coordinate type can be set with the COORD keyword in the $CONTRL group. Possible values are:

- UNIQUE: only the symmetry unique atoms will be given, in Cartesian coordinates (this is the default; use this unless you have a good reason to use COORD=CART);
- HINT: only the symmetry unique atoms will be given, in Hilderbrandt style internals;
- CART: Cartesian coordinates will be input (this option is generally not recommended, see below);
- ZMT: GAUSSIAN style internals will be input;
- ZMTMPC: MOPAC style internals will be input;
- FRAGONLY: this means no part of the system is treated by ab initio means, hence $DATA is not given. The system is to be fully specified by $EFRAG.

The CART, ZMT, and ZMTMPC choices require input of all atoms in the molecule. These three also orient the molecule, and then determine which atoms are unique. The reorientation is very likely to change the order of the atoms from what you input. When the point group contains a 3-fold or higher rotation axis, the degenerate moments of inertia often cause problems choosing correct symmetry unique axes, in which case you must use COORD=UNIQUE rather than Z-matrices. It is also important to realize that the reorientation into principal axes is done only for atomic coordinates, and is not applied to the axis dependent data of the groups $VEC, $HESS, $GRAD, $DIPDR, and $VIB, nor to Cartesian coordinates of effective fragments in $EFRAG. COORD=UNIQUE avoids reorientation, and is thus the safest way to read these.

Note that the choices CART, ZMT, ZMTMPC require the use of a $BASIS group to define the basis set. The first two choices might or might not use $BASIS, as you wish.

Furthermore, it is important to note that the choice COORD=CART currently forces the job to be a check run (*i.e.*, EXETYP=CHECK). The reason for this is that it is generally not recommended to use this coordinate type. For users who need this coordinate input type and know what they’re doing, COORD=CART jobs can be forced to run by using the "-run" command line switch.

For Cartesian coordinates, the distance unit is set with the keyword UNITS in the $CONTRL group. Possible values are ANGS (for Ångstroms, the default) and BOHR (for Bohr atomic unites).

In the next three sections, Cartesian coordinates, Z-Matrix coordinates, and the use of DLCs will be discussed separately. In addition, there will be two sections devoted to the use of symmetry and isotopic substitution, respectively. Hilderbrandt internals will not be discussed in detail. Instead, for this coordinate type, we would like to refer the reader to the list of keywords as well as to the following reference:

R.L. Hilderbrandt, J. Chem. Phys. 51, 1654 (1969).

Cartesian coordinates

Cartesian coordinates should be input in the form of:

```
NAME    NUCLEAR_CHARGE    X-COORD    Y-COORD    Z-COORD
```

Here, NAME is an arbitrary name. One could use the elements name or symbol, but in principal any name is allowed, e.g. CAT, MOUSE, or, if in the case of a heavier element, ELEPHANT. NUCLEAR_CHARGE is the atom's nuclear charge. An example:

```
 $DATA
 Water
C1
OXYGEN      8.0     -0.708955260     -0.940298490      0.000000000
HYDROGEN    1.0      0.251044740     -0.940298490      0.000000000
HYDROGEN    1.0     -1.029409850     -0.035362660      0.000000000
 $END
```

Coordinates may be omitted when they equal 0, starting from Z coordinate. The above example when oriented differently:

```
 $DATA
 Water
C1
OXYGEN      8.0     -0.062007499
HYDROGEN    1.0      0.721968395      0.554059380
HYDROGEN    1.0     -0.845983394      0.554059380
 $END
```

Note that optimizations in Cartesian coordinates have a reputation of converging slowly. This is largely due to the fact that translations and rotations are usually left in the problem. Numerical problems caused by the small eigenvalues associated with these degrees of freedom are the source of this poor convergence. The methods in Firefly project the Hessian matrix to eliminate these degrees of freedom, which should not cause a problem.

Nonetheless, Cartesian coordinates are in general the most slowly convergent coordinate system.

Z-Matrix and natural internal coordinates

Internal coordinates are able to provide convergence faster than with Cartesian coordinates. They can be specified with NZVAR in $CONTRL and with $ZMAT group. An simple input example (taken from the example job EXAM06):

```
$CONTRL NZVAR=3 COORD=ZMT $END
$DATA
Methylene
Cnv  2

C
H 1 rCH
H 1 rCH 2 aHOH

rCH=1.09
aHOH=99.0
 $END
 $ZMAT   IZMAT(1)=1,1,2,   1,1,3,   2,2,1,3  $END
```

Benefits of this coordinate type are the elimination of the six rotational and translational degrees of freedom and that the GUESS Hessian is able to use empirical information about bond stretches and bends. On the other hand, there are many possible choices for the internal coordinates, some of which may lead to much poorer convergence of the geometry search than others. Particularly poorly chosen coordinates may not even converge at all.

One thing to keep in mind is that internal coordinates are frequently strongly coupled. A very common example to illustrate this might be a bond length in a ring, and the angle on the opposite side of the ring. Clearly, changing one changes the other simultaneously. A more mathematical definition of "coupled" is to say that there is a large off-diagonal element in the Hessian. In this case convergence may be unsatisfactory, especially with a diagonal GUESS Hessian, where a "good" set of internals is one with a diagonally dominant Hessian. Of course, if you provide an accurately computed Hessian, it will have large off-diagonal values where those are truly present. Even so, convergence may be poor if the coordinates are coupled through large 3rd or higher derivatives. The best coordinates are therefore those which are the most "quadratic".

One very popular set of internal coordinates is the usual "model builder" Z-matrix input, where for N atoms, one uses N-1 bond lengths, N-2 bond angles, and N-3 bond torsions. The popularity of this choice is based on its ease of use in specifying the initial molecular geometry. Typically, however, it is the worst possible choice of internal coordinates, and in the case of rings, is not even as good as Cartesian coordinates.

However, Firefly does not require this particular mix of the common types. Firefly's only requirement is that you use a total of 3N-6 coordinates,

chosen from these 3 basic types, or several more exotic possibilities. (Of course, we mean 3N-5 throughout for linear molecules.) These additional types of internal coordinates include linear bends for 3 collinear atoms, out of plane bends, and so on. There is no reason at all why you should place yourself in a straightjacket of N-1 bonds, N-2 angles, and N-3 torsions. If the molecule has symmetry, be sure to use internals which are symmetrically related.

For example, the most effective choice of coordinates for the atoms in a four membered ring is to define all four sides, any one of the internal angles, and a dihedral defining the ring pucker. For a six membered ring, the best coordinates seem to be 6 sides, 3 angles, and 3 torsions. The angles should be every other internal angle, so that the molecule can "breathe" freely. The torsions should be arranged so that the central bond of each is placed on alternating bonds of the ring, as if they were pi bonds in Kekule benzene. For a five membered ring, we suggest all 5 sides, 2 internal angles, again alternating every other one, and 2 dihedrals to fill in. The internal angles of necessity skip two atoms where the ring closes. Larger rings should generalize on the idea of using all sides but only alternating angles. If there are fused rings, start with angles on the fused bond, and alternate angles as you go around from this position.

Rings and more especially fused rings can be tricky. For these systems, especially, we suggest the Cadillac of internal coordinates, the "natural internal coordinates" of Peter Pulay. For a description of these, see:

P. Pulay, G. Fogarosi, F. Pang, J. E. Boggs, J. Am. Chem. Soc. 101, 2550-2560 (1979)
G. Fogarasi, X. Zhou, P. W. Taylor, P. Pulay J.Am.Chem.Soc. 114, 8191-8201 (1992)

These are linear combinations of local coordinates, except in the case of rings. The examples given in these two papers are very thorough.

An illustration of natural internal coordinates is given in the example job EXAM25. This is a nonsense molecule, designed to show many kinds of functional groups. It is defined using standard bond distances with a classical Z-matrix input, and the angles in the ring are adjusted so that the starting value of the unclosed OO bond is also a standard value. Using Cartesian coordinates is easiest, but takes a very large number of steps to converge. This however, is better than using the classical Z-matrix internals given in $DATA, which is accomplished by setting NZVAR to the correct 3N-6 value. The geometry search changes the OO bond length to a very short value on the 1st step, and the SCF fails to converge. (Note that if you have used dummy atoms in the $DATA input, you cannot simply enter NZVAR to optimize in internal coordinates, instead you must give a $ZMAT which involves only real atoms).

The third choice of internal coordinates in EXAM25, natural internal coordinates, is the best set which can be made from the simple coordinates. It follows the advice given above for five membered rings, and because it includes the OO bond it has no trouble with crashing this bond. It takes 20 steps to converge, so the trouble of generating this $ZMAT can be worth it when compared to the use of Cartesians. Natural internal coordinates are

defined in the final group of input. The coordinates are set up first for the ring, including two linear combinations of all angles and all torsions within the ring. After this the methyl is hooked to the ring as if it were a NH group, using the usual terminal methyl hydrogen definitions. The H is hooked to this same ring carbon as if it were a methine. The NH and the CH2 within the ring follow Pulay's rules exactly. The amount of input is much greater than a normal Z-matrix. For example, 46 internal coordinates are given, which are then placed in 3N-6=33 linear combinations. Note that natural internals tend to be rich in bends, and short on torsions.

The energy results for the three coordinate systems which converge are as follows:

NSERCH	Cart	good Z-mat	nat. int.
0	-48.6594935049	-48.6594935049	-48.6594935049
1	-48.6800538676	-48.6806631261	-48.6838361406
2	-48.6822702585	-48.6510215698	-48.6874045449
3	-48.6858299354	-48.6882945647	-48.6932811528
4	-48.6881499412	-48.6849667085	-48.6946836332
5	-48.6890226067	-48.6911899936	-48.6959800274
6	-48.6898261650	-48.6878047907	-48.6973821465
7	-48.6901936624	-48.6930608185	-48.6987652146
8	-48.6905304889	-48.6940607117	-48.6996366016
9	-48.6908626791	-48.6949137185	-48.7006656309
10	-48.6914279465	-48.6963767038	-48.7017273728
11	-48.6921521142	-48.6986608776	-48.7021504975
12	-48.6931136707	-48.7007305310	-48.7022405019
13	-48.6940437619	-48.7016095285	-48.7022548935
14	-48.6949546487	-48.7021531692	-48.7022569328
15	-48.6961698826	-48.7022080183	-48.7022570260
16	-48.6973813002	-48.7022454522	-48.7022570662
17	-48.6984850655	-48.7022492840	
18	-48.6991553826	-48.7022503853	
19	-48.6996239136	-48.7022507037	
20	-48.7002269303	-48.7022508393	
21	-48.7005379631		
22	-48.7008387759		
	...		
50	-48.7022519950		

from which you can see that the natural internals are actually the best set. The $ZMAT exhibits upward burps in the energy at step 2, 4, and 6, so that for the same number of steps, these coordinates are always at a higher energy than the natural internals.

The initial Hessian generated for these three columns contains 0, 33, and 46 force constants. This assists the natural internals, but is not the major reason for its superior performance. The computed Hessian at the final geometry of this molecule, when transformed into the natural internal coordinates is almost diagonal. This almost complete uncoupling of coordinates is what makes the natural internals perform so well. The conclusion is of course that not all coordinate systems are equal, and natural internals are the best. As another example, we have run the ATCHCP molecule, which is a popular geometry optimization test, due to its two fused rings:

H. B. Schlegel, Int. J. Quantum Chem., Symp. 26, 253-264 (1992)
T. H. Fischer and J. Almlof, J. Phys. Chem. 96, 9768-9774 (1992)
J. Baker, J. Comput. Chem. 14, 1085-1100 (1993)

Here, we have compared the same coordinate types, using a guess Hessian, or a computed Hessian. The latter set of runs is a test of the coordinates only, as the initial Hessian information is identical. The results show clearly the superiority of the natural internals, which like the previous example, give an energy decrease on every step:

	HESS=GUESS	HESS=READ
Cartesians	65	41 steps
good Z-matrix	32	23
natural internals	24	13

A final example is phosphinoazasilatrane, with three rings fused on a common SiN bond, in which 112 steps in Cartesian space became 32 steps in natural internals. The moral here is: "A little brain time can save a lot of CPU time".

Delocalized coordinates

A relatively new type of internal coordinate is the delocalized internal coordinate (DLC), which generally provides fast convergence and is easier to set up than Z-matrix coordinates. It is described by J. Baker, A. Kessi, and B. Delley (J. Chem. Phys. 1996, 105, 192-212), although the implementation in Firefly is not exactly the same. Bonds are kept as independent coordinates while angles are placed in linear combination by the DLC process. There are some interesting options for applying constraints, and other options to assist the automatic DLC generation code by either adding or deleting coordinates.

It is simple to use DLCs in their most basic form. One has to specify:

```
 $CONTRL NZVAR=value $END
 $ZMAT DLC=.T. AUTO=.T. $END
```

where the value of NZVAR is nonzero. As with ZMAT input, setting NZVAR=0 disables the use of DLCs. This can be used as an easy way of switching between internal and Cartesian coordinates without the need to remove of comment out additional DLC related directives from the input.

Because of the popularity of DLCs, it is also possible to enable them using a 'shortcut'. If NZVAR is nonzero, input coordinates are in Cartesian format, and no $ZMAT group is given, then Firefly will act as if the input contains $ZMAT DLC=.T. AUTO=.T. $END with a nonzero value of NZVAR.

Though the quality of DLCs are not as good as explicitly constructed natural internals (which benefit from human chemical knowledge), they are almost always better than carefully crafted $ZMATs using only the primitive

internal coordinates. Because of their relative ease to use, we recommend their use highly.

As mentioned earlier, DLCs are generated from Cartesian or Z-matrix coordinates by Firefly's DLC generator. The generation process, however, is not entirely black box as there are cases in which the generator can fail. In many such cases it might help to set:

```
 $ZMAT DLCTOL=1D-7 ORTTOL=1D-7 $END
```

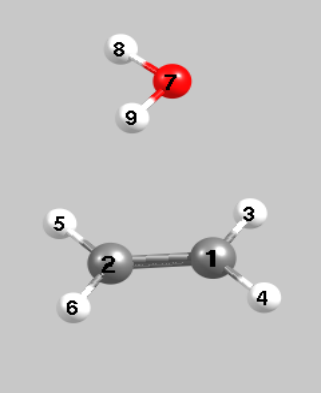
which lowers the threshold used to check the quality/completeness of DLCs.

Another possible remedy might be to choose a different method for freezing internals in DLCs with the IFDMOD keyword in $ZMAT. Possible values are 0 (the most stable method), 1 (a less stable method), and 2 (an experimental method). Setting IFDMOD=2 has been found to help in a number of cases.

There are, however, a few important general cases in which these settings do not help. Important examples of these are multimolecular systems and systems in which four or more atoms lie on a straight line (in some cases, three atoms on one line may also pose a problem). For such cases, one has to manually define one or more additional bonds through the NONVDW keyword in the $ZMAT group in order to reach the necessary amount of linearly independent coordinates. NONVDW should be given as an array that describes atom pairs. For example, NONVDW(1)=2,3,5,6 describes bonds between atoms 2 and 3, and between atoms 5 and 6. You may add as many bonds as you want through NONVDW. However, beware that the addition of too many bonds may degrade the performance of the optimization algorithm.

For clarity, let us consider two basic examples.

The first example is a system consisting of water and ethene, as depicted in the first image. Here, one or more additional bonds have to be added through the NONVDW array. Typically, these additional bonds should be (by order of importance):



a) bonds that are expected to be broken (or formed) in the process under study

b) if the system under investigation includes more than one fragment, at least one "inter-fragment"

For this particular system, one should specify one of the following:

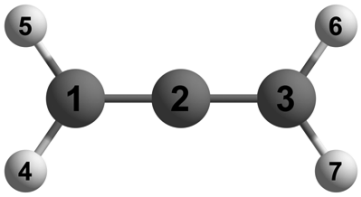
```
 $ZMAT NONVDW(1)= 1,7  $END          or
 $ZMAT NONVDW(1)= 1,8  $END          or
 $ZMAT NONVDW(1)= 1,9  $END
```

Again, it is not recommended to add too many bonds. In this example, addition of two explicit bonds, rather than one, increased the number of optimization steps from around 50 to almost 90.

However, if you want to study the addition of the water to the ethylene above, your NONVDW group should read:

```
 $ZMAT NONVDW(1)=1,7,2,8,2,9  $END
```

since the addition of the water to the ethylene may (depending on your basis set and theory level) immediately cause one of the O-H bonds to break and the migration of that H nucleus to carbon number 2.



As a second example, let us consider the allene molecule, which is depicted in the second image. Because of the 180 degree bond angle between atoms 1, 2, and 3, at least one additional bond has to be specified. The specification of

```
 $ZMAT NONVDW(1)=4,2 $END
```

allows the DLC generator the find enough linearly independent coordinates. A different atom pair (such as 5,2) can also be specified to get the same result.

Utilizing symmetry

The symmetry group of the system under investigation can be specified in the $DATA group, below the title card, in the form:

```
      GROUP NAXIS
```

Here, GROUP is the Schoenflies symbol of the symmetry group. You may choose from:

C1, CS, CI, CN, S2N, CNH, CNV, DN, DNH, DND, T, TH, TD, O, OH

NAXIS is the order of the highest rotation axis, and must be given when the name of the group contains an N. For example, "CNV 2" is C2v. "S2N 3" means S6. For linear molecules, choose either CNV or DNH, and enter NAXIS as 4. Enter single atoms as DNH with NAXIS=2 (see also input example "EXAM16").

When group C1 is specified, the atom input can directly start on the line after "C1".

```
$DATA
 Water
C1
OXYGEN      8.0     -0.708955260     -0.940298490      0.000000000
HYDROGEN    1.0      0.251044740     -0.940298490      0.000000000
HYDROGEN    1.0     -1.029409850     -0.035362660      0.000000000
 $END
```

When a different point group is specified, two additional input lines have to be given prior to the atom input. These two cards specify the coordinates and orientation of the axis/planes of symmetry. As input structures

are nowadays usually prepared with external molecule building programs (which are able to orient the system correctly with respect to axis/planes of symmetry), these two input lines have little use. They can be skipped over by providing a single blank line, as in the example below:

```
$DATA
 H2O
CNV 2

OXYGEN      8.0      0.000000000          0.000000000         -0.066188278
HYDROGEN    1.0     -0.751549274          0.000000000          0.525227863
 $END
```

Armed with only the name of the group, Firefly is able to exploit the molecular symmetry throughout almost all of the program, and thus save a great deal of computer time. Firefly does not require that you know very much else about group theory, although a deeper knowledge (character tables, irreducible representations, term symbols, and so on) is useful when dealing with the more sophisticated wavefunctions.

It should finally be noted that the use of symmetry can be disabled/enabled through the NOSYM keyword in the $CONTRL group. By default, symmetry is enabled (NOSYM=0). When NOSYM=1 is specified, symmetry is only used to build the molecule and not in any calculations. Disabling symmetry is necessary

- for GVB and MCSCF runs in which the charge density is not fully symmetric;
- for polarizability calculations with RUNTYP=TDHF;
- for effective fragment potential calculations;
- for many DRC runs;
- in some cases when rotating alpha and beta HOMO and LUMO orbitals in the initial guess with $GUESS MIX=.T $END.

Isotopic substitution

Isotopic substitution can be controlled with the AMASS keyword in the $MASS group. AMASS is an array that specifies the atomic masses of elements in $DATA, in amu. The default is to use the mass of the most abundant isotope. Masses through element 104 are stored. For example:

```
 $MASS AMASS(3)=2.0140 $END
```

will make the third atom in the molecule a deuterium. Masses affect only the frequencies and normal modes of vibration.

Basis sets

Introduction

Firefly uses Gaussian basis functions for the construction of the molecular orbitals. It can handle basis sets of the segmented contraction and of the general contraction type, with support for s, p, d, f, and g functions, as well as Pople style sp functions. Higher functions such as h and i are not supported, which is important to keep in mind when using basis sets that include these functions (such as Dunning's cc-pV5Z set). Additionally, there is support for effective core potentials (i.e. pseudopotentials).

This chapter will start with a description of the basis sets included in Firefly, followed by a brief explanation on the use of spherical or Cartesian functions. It will then discuss ways in which one can manually specify a basis set, making it possible to either modify a basis set or use a basis set that is not included in Firefly. The specification of effective core potentials will also be discussed. Finally, the chapter will conclude with a short discussion on how to handle partial linear dependence in a basis set.

Built-in basis sets

The following built-in basis sets can be requested through the GBASIS keyword. References for these basis sets can be found at the end of this section.

```
* GBASIS=STO - Pople's STO-NG minimal basis set

      Available H-Xe, for NGAUSS=2,3,4,5,6

* GBASIS=N21 - Pople's N-21G split valence basis set

      Available H-Xe, for NGAUSS=3
      Available H-Ar, for NGAUSS=6

* GBASIS=N31 - Pople's N-31G split valence basis set

      Available H-Ne,P-Cl for NGAUSS=4
      Available H-He,C-F for NGAUSS=5
      Available H-Ar, for NGAUSS=6
      For Ga-Kr, N31 selects the BC basis

* GBASIS=N311 - Pople's "triple split" N-311G basis set

      Available H-Ne, for NGAUSS=6
      Selecting N311 implies MC for Na-Ar
```

* GBASIS=MC - McLean/Chandler "triple split" basis

      (12s,9p)/[6s,5p] for Na-Ar
      Selecting MC implies 6-311G for H-Ne

The first four basis sets are designed by Pople and coworkers, the last basis set is by McLean and Chandler and is often used together with 6-311G. Specifically, if N-311G is requested on a second row atom, the McLean Chandler basis set is used on this atom instead (and in the same manner, when McLean/Chandler is requested on H, He, or a first row atom, 6-311G will be used). For the N-31G basis sets, the 'BC' basis set (a double-zeta valence basis set by Binning and Curtiss) will be used on Ga-Kr. For all four Pople basis sets, the number of primitive gaussians to be used for core orbitals should be set with the NGAUSS keyword. Popular choices are to use STO-3G, 3-21G, and 6-31G.

* GBASIS=MINI - Huzinaga's 3 gaussian minimal basis set

      Available H-Rn.

* GBASIS=MIDI - Huzinaga's 21 split valence basis set

      Available H-Rn.

These two basis sets were designed by Huzinaga and coworkers. The MINI basis consists of three gaussian expansions of each atomic orbital. The exponents and contraction coefficients are optimized for each element, and s and p exponents are not constrained to be equal. As a result these bases give much lower energies than does STO-3G. The valence MINI orbitals of main group elements are scaled by factors while transition metal MINI bases are not scaled.

The MIDI bases are derived from the MINI sets by decontracting the outer valence function, thus making them of the split valence type. MIDI bases are not scaled by Firefly. The transition metal bases are taken from the lowest SCF terms in the s**1,d**n configurations.

Note that nowadays multiple basis sets carry the MINI or MIDI name. The MINI and MIDI basis sets incorporated in Firefly are in literature commonly referred to as MINI-1 and MIDI-1.

* GBASIS=DH - Dunning/Hay "double zeta" basis set

      (3s)/[2s] for H
      (9s,4p)/[3s,2p] for Li
      (9s,5p)/[3s,2p] for Be-Ne
      (11s,7p)/[6s,4p] for Al-Cl

* GBASIS=DZV - "double zeta valence" basis set

      A synonym for DH for H, Li, Be-Ne, Al-Cl
      (14s,9p,3d)/[5s,3p,1d] for K-Ca

(14s,11p,5d/[6s,4p,1d] for Ga-Kr (= the BC basis)

* GBASIS=TZV - "triple zeta valence" basis set

(5s)/[3s] for H
(10s,3p)/[4s,3p] for Li
(10s,6p)/[5s,3p] for Be-Ne
A synonym for MC for Na-Ar
(14s,9p)/[8s,4p] for K-Ca
(14s,11p,6d)/[10s,8p,3d] for Sc-Zn

The DZV and DH sets are identical for H, Li, Be-Ne, and Al-Cl – both use the double zeta set by Dunning and Hay. The only difference is that DZV can also be used on K-Ca and Ga-Kr. This basically makes the DH set redundant; it is actually only included in Firefly for backwards compatibility reasons. The TZV set uses a triple zeta set by Dunning for H and Li-Ne, the McLean/Chandler set for Na-Ar, and Wachters' bases for K-Ca and Sc-Zn.

* GBASIS=SBKJC - Stevens/Basch/Krauss/Jasien/Cundari valence
basis set, for Li-Rn. This choice implies an
unscaled -31G basis for H-He.

* GBASIS=HW - Hay/Wadt valence basis. This is a -21 split,
available for Na-Xe, except for the transition metals.
This implies a 3-21G basis for H-Ne.

These two options request valence only basis sets, meant to be used in combination with effective core potentials (which describe the core orbitals). Effective core potentials are discussed in a later section.

* GBASIS=MNDO
* GBASIS=AM1
* GBASIS=PM3
* GBASIS=RM1

These four options do not request a basis set, but request the use of a semi-empirical method. They are discussed in separate chapter. Requesting one of these methods causes all other keywords in $BASIS to be ignored.

The addition of polarization functions to the basis set can be requested with the NDFUNC, NPFUNC, and NFFUNC keywords. NDFUNC specifies the amount of d functions on 'heavy' atoms, except for MINI/MIDI where it requests additional p functions. For the STO, HW, and N21 sets 'heavy' means Na and heavier, for other basis sets it means Li and heavier. NPFUNC specifies the amount of p functions that should be added to H and He. Finally, NFFUNC specifies the amount of f functions that is to be added to Li-Cl. NDFUNC and NPFUNC may not exceed 3. The only permitted values for NFFUNC are 0 and 1.

Five different sets of polarization exponents are available in Firefly. Which set of polarization exponents is used is determined by the keyword POLAR, which can be set to following values:

- POPLE (chooses exponents designed for GBASIS=STO, N21, N31, SBKJC, HW)
- POPN311 (chooses exponents designed for GBASIS=N311, MC)
- DUNNING (chooses exponents designed for GBASIS=DH, DZV)
- HUZINAGA (chooses exponents designed for GBASIS=MINI, MIDI)
- HONDO7 (chooses exponents designed for GBASIS=TZV)

It is normally not necessary to specify POLAR as Firefly will automatically pick the set of polarization exponents that matches the basis set chosen with GBASIS. A detailed list of exponents as well as references is given later on in this section.

Two more keywords that pertain to the specification of polarization functions are SPLIT2 and SPLIT3. These keywords specify the splitting factors when NDFUNC and/or NPFUNC is chosen as >1. In such a case, the 1d and/or 1p single values are split according to the chosen values. For example, SPLIT2=2.0,0.5 means to double and halve the single polarization exponent. The default values (SPLIT2=2.0,0.5 and SPLIT3=4.00,1.00,0.25) are from the Pople school, and as they were derived with correlation in mind they are probably too far apart for Hartree-Fock. The default SPLIT2 value will usually cause an >increase< over the 1d energy at the HF level for hydrocarbons. For HF, SPLIT2=0.4,1.4 will always lower the SCF energy. For SPLIT3, we might suggest 3.0,1.0,1/3. For more information, see also:

M.J.Frisch, J.A.Pople, J.S.Binkley J.Chem.Phys. 80, 3265-3269 (1984).

The addition of diffuse basis functions to the basis set can be controlled with the DIFFSP and DIFFS keywords (possible values being .T. or .F.). The first keyword requests diffuse functions on ‘heavy atoms’, i.e. Li-F, Na-Cl, Ga-Br, In-I, and Tl-At. The latter keyword requests diffuse functions on H and He. A list of diffuse exponents used in Firefly is given at the end of this section. As opposed to the case with polarization exponents, Firefly contains only a single set of diffusion exponents that is used for all basis sets available through GBASIS.

By default, DIFFSP requests diffuse functions for all heavy atoms present in $DATA. It is however possible to limit the addition of diffuse functions to specific elements. When DIFFSP is set to .T., this can be done with the ELNEG keyword. For example,

```
DIFFSP=.T. ELNEG(1)=7,8,9
```

adds diffuse functions to N, O, and F, but not to other elements. ELNEG only affects heavy atoms, it cannot be used to include or exclude diffuse functions on H and He (this can only be controlled through DIFFS).

This section will conclude with a list of details and references for all included basis sets, polarization functions, and diffuse functions.

STO-NG

H-Ne	Ref. 1 and 2
Na-Ar	Ref. 2 and 3 **
K,Ca,Ga-Kr	Ref. 4
Rb,Sr,In-Xe	Ref. 5
Sc-Zn,Y-Cd	Ref. 6

1) W.J.Hehre, R.F.Stewart, J.A.Pople J.Chem.Phys. 51, 2657-2664(1969).
2) W.J.Hehre, R.Ditchfield, R.F.Stewart, J.A.Pople J.Chem.Phys. 52, 2769-2773(1970).
3) M.S.Gordon, M.D.Bjorke, F.J.Marsh, M.S.Korth J.Am.Chem.Soc. 100, 2670-2678(1978). ** the valence scale factors for Na-Cl are taken from this paper, rather than the "official" Pople values in Ref. 2.
4) W.J.Pietro, B.A.Levi, W.J.Hehre, R.F.Stewart, Inorg.Chem. 19, 2225-2229(1980).
5) W.J.Pietro, E.S.Blurock, R.F.Hout,Jr., W.J.Hehre, D.J. DeFrees, R.F.Stewart Inorg.Chem. 20, 3650-3654(1980).
6) W.J.Pietro, W.J.Hehre J.Comput.Chem. 4, 241-251(1983).

MINI/MIDI

H-Xe	Ref. 7

7) "Gaussian Basis Sets for Molecular Calculations" S.Huzinaga, J.Andzelm, M.Klobukowski, E.Radzio-Andzelm, Y.Sakai, H.Tatewaki Elsevier, Amsterdam, 1984.

3-21G

H-Ne	Ref. 8	(also 6-21G)
Na-Ar	Ref. 9	(also 6-21G)
K,Ca,Ga-Kr,Rb,Sr,In-Xe	Ref. 10	
Sc-Zn	Ref. 11	
Y-Cd	Ref. 12	

8) J.S.Binkley, J.A.Pople, W.J.Hehre J.Am.Chem.Soc. 102, 939-947(1980).
9) M.S.Gordon, J.S.Binkley, J.A.Pople, W.J.Pietro, W.J.Hehre J.Am.Chem.Soc. 104, 2797-2803(1982).
10) K.D.Dobbs, W.J.Hehre J.Comput.Chem. 7, 359-378(1986)
11) K.D.Dobbs, W.J.Hehre J.Comput.Chem. 8, 861-879(1987)
12) K.D.Dobbs, W.J.Hehre J.Comput.Chem. 8, 880-893(1987)

N-31G

references for	4-31G	5-31G	6-31G
H	13	13	13
He	21	21	21
Li	17,22		17

Be	18,22		18
B	15		17
C-F	13	14	14
Ne	21		21
Na-Ga			20
Si			19 **
P-Cl	16		20
Ar			20
K-Zn			23

13) R.Ditchfield, W.J.Hehre, J.A.Pople J.Chem.Phys. 54, 724-728(1971).
14) W.J.Hehre, R.Ditchfield, J.A.Pople J.Chem.Phys. 56, 2257-2261(1972).
15) W.J.Hehre, J.A.Pople J.Chem.Phys. 56, 4233-4234(1972).
16) W.J.Hehre, W.A.Lathan J.Chem.Phys. 56, 5255-5257(1972).
17) J.D.Dill, J.A.Pople J.Chem.Phys. 62, 2921-2923(1975).
18) J.S.Binkley, J.A.Pople J.Chem.Phys. 66, 879-880(1977).
19) M.S.Gordon Chem.Phys.Lett. 76, 163-168(1980)

** - Note that the built in 6-31G basis for Si is not that given by Pople in reference 20. The basis by Mark Gordon gives a better wavefunction for a ROHF calculation in full atomic (Kh) symmetry:

6-31G	Energy	virial
Gordon	-288.828573	1.999978
Pople	-288.828405	2.000280

See the input example "EXAM16" for information on how to run in Kh.

20) M.M.Francl, W.J.Pietro, W.J.Hehre, J.S.Binkley, M.S.Gordon, D.J.DeFrees, J.A.Pople J.Chem.Phys. 77, 3654-3665(1982).
21) Unpublished, copied out of GAUSSIAN82.
22) For Li and Be, 4-31G is actually a 5-21G expansion.
23) V.A.Rassolov, J.A.Pople, M.A.Ratner, T.L.Windus J.Chem.Phys. 109, 1223-1229(1998)

6-311G

24) R.Krishnan, J.S.Binkley, R.Seeger, J.A.Pople J.Chem.Phys. 72, 650-654(1980).

DH / DZV / BC

DH basis	H	Ref. 25
DH basis	Li-Ne	Ref. 25
DH basis	Al-Ar	Ref. 25
DZV basis	K,Ca	Ref. 26
DZV basis	Ga-Kr	Ref. 27 (a.k.a. the BC basis)

25) T.H.Dunning, Jr., P.J.Hay Chapter 1 in "Methods of Electronic Structure Theory", H.F.Shaefer III, Ed. Plenum Press, N.Y. 1977, pp 1-27.
Note that Firefly uses inner/outer scale factors of 1.2 and 1.15 for DH's hydrogen. To get Thom's usual basis, scaled 1.2 throughout:

```
HYDROGEN 1.0 x, y, z
```

DH 0 1.2 1.2

26) J.-P.Blaudeau, M.P.McGrath, L.A.Curtiss, L.Radom J.Chem.Phys. 107, 5016-5021(1997)
27) R.C.Binning, Jr., L.A.Curtiss J.Comput.Chem. 11, 1206-1216(1990)

TZV / MC

H	Ref. 28
Li-Ne	Ref. 28
Na-Ar	Ref. 29 (a.k.a. the MC basis)
K,Ca	Ref. 30
Ga-Kr	Ref. 30**

28) T.H. Dunning, J.Chem.Phys. 55 (1971) 716-723.
29) A.D.McLean, G.S.Chandler J.Chem.Phys. 72,5639-5648(1980).
30) A.J.H. Wachters, J.Chem.Phys. 52 (1970) 1033-1036 (see Table VI, Contraction 3).

** Ga-Kr is taken from HONDO 7 and is Wachters' (14s9p5d) basis (ref. 30) contracted to (10s8p3d) with the following modifications:

1. the most diffuse s removed;
2. additional s spanning 3s-4s region;
3. two additional p functions to describe the 4p;
4. (6d) contracted to (411) from ref. 31, except for Zn where Wachter's (5d)/[41] and Hay's diffuse d are used.

31) A.K. Rappe, T.A. Smedley, and W.A. Goddard III, J.Phys.Chem. 85 (1981) 2607-2611

SBKJC -31G splits, bigger for trans. metals (available Li-Rn)

32) W.J.Stevens, H.Basch, M.Krauss J.Chem.Phys. 81, 6026-6033 (1984)
33) W.J.Stevens, H.Basch, M.Krauss, P.Jasien Can.J.Chem, 70, 612-630 (1992)
34) T.R.Cundari, W.J.Stevens J.Chem.Phys. 98, 5555-5565(1993)

HW -21 splits (sp exponents not shared; transition metals are not built in at present, although they will work if you type them in)

35) P.J.Hay, W.R.Wadt J.Chem.Phys. 82, 270-283 (1985) main group (available Na-Xe)
36) W.R.Wadt, P.J.Hay J.Chem.Phys. 82, 284-298 (1985)
see also
37) P.J.Hay, W.R.Wadt J.Chem.Phys. 82, 299-310 (1985)

Polarization exponents

STO-NG*	ref. 38
3-21G*	ref. 39 (see also ref. 10)
6-31G*	ref. 40 (see also ref. 20)

6-31G** ref. 40 (see also ref. 20)

38) J.B.Collins, P. von R. Schleyer, J.S.Binkley, J.A.Pople J.Chem.Phys. 64, 5142-5151(1976).
39) W.J.Pietro, M.M.Francl, W.J.Hehre, D.J.DeFrees, J.A. Pople, J.S.Binkley J.Am.Chem.Soc. 104,5039-5048(1982)
40) P.C.Hariharan, J.A.Pople Theoret.Chim.Acta 28, 213-222(1973)

Multiple polarization, and f functions

41) M.J.Frisch, J.A.Pople, J.S.Binkley J.Chem.Phys. 80, 3265-3269 (1984).

Polarization exponents built into Firefly are listed in the table below. The values are for d functions unless otherwise indicated. Please note that the names associated with each column are only generally descriptive. For example, the column marked "POPLE" contains a value for Si with which John Pople would not agree, and the Ga-Kr values in this column are actually from the Huzinaga "green book". The exponents for K-Kr under "DUNNING" are from Curtiss, et al., not Thom Dunning. And so on. A blank means the value equals the "POPLE" column.

	POPLE	POPN311	DUNNING	HUZINAGA	HONDO7
H	1.1(p)	0.75(p)	1.0(p)	1.0(p)	1.0(p)
He	1.1(p)	0.75(p)	1.0(p)	1.0(p)	1.0(p)
Li	0.2	0.200		0.076(p)	
Be	0.4	0.255		0.164(p)	0.32
B	0.6	0.401	0.70	0.388	0.50
C	0.8	0.626	0.75	0.600	0.72
N	0.8	0.913	0.80	0.864	0.98
O	0.8	1.292	0.85	1.154	1.28
F	0.8	1.750	0.90	1.496	1.62
Ne	0.8	2.304	1.00	1.888	2.00
Na	0.175			0.061(p)	0.157
Mg	0.175			0.101(p)	0.234
Al	0.325			0.198	0.311
Si	0.395			0.262	0.388
P	0.55			0.340	0.465
S	0.65			0.421	0.542
Cl	0.75			0.514	0.619
Ar	0.85			0.617	0.696
K	0.2		0.260	0.039(p)	
Ca	0.2		0.229	0.059(p)	
Sc-Zn	0.8(f)	N/A	N/A	N/A	N/A
Ga	0.207		0.141		
Ge	0.246		0.202		
As	0.293		0.273		
Se	0.338		0.315		
Br	0.389		0.338		

Kr	0.443	0.318	
Rb	0.11		0.034(p)
Sr	0.11		0.048(p)

Common d polarization used for all basis sets (from the "green book") are as follows:

In	Sn	Sb	Te	I	Xe
0.160	0.183	0.211	0.237	0.266	0.297
Tl	Pb	Bi	Po	At	Rn
0.146	0.164	0.185	0.204	0.225	0.247

Firefly uses the following f polarization functions (these are from reference 41):

Li	Be	B	C	N	O	F	Ne
0.15	0.26	0.50	0.80	1.00	1.40	1.85	2.50
Na	Mg	Al	Si	P	S	Cl	Ar
0.15	0.20	0.25	0.32	0.45	0.55	0.70	--

Diffuse exponents

42) T.Clark, J.Chandrasekhar, G.W.Spitznagel, P. von R. Schleyer J. Comput. Chem. 4, 294-301 (1983)
43) G.W.Spitznagel, Diplomarbeit, Erlangen, 1982.

The following exponents are for L shells, except those for H and He. For H-F, they are taken from ref 42. For Na-Cl, they are taken directly from reference 43. These values may be found in footnote 13 of reference 41. For Ga-Br, In-I, and Tl-At the exponents were optimized for the atomic ground state anion, using ROHF with a flexible ECP basis set, by Ted Packwood at NDSU.

H						He
0.0360						0.0860
Li	Be	B	C	N	O	F
0.0074	0.0207	0.0315	0.0438	0.0639	0.0845	0.1076
Na	Mg	Al	Si	P	S	Cl
0.0076	0.0146	0.0318	0.0331	0.0348	0.0405	0.0483
		Ga	Ge	As	Se	Br
		0.0205	0.0222	0.0287	0.0318	0.0376
		In	Sn	Sb	Te	I
		0.0223	0.0231	0.0259	0.0306	0.0368
		Tl	Pb	Bi	Po	At
		0.0170	0.0171	0.0215	0.0230	0.0294

Additional information about diffuse functions and also Rydberg type exponents can be found in reference 25. The following atomic energies are from UHF calculations (RHF on 1-S states), with p orbitals not symmetry equivalenced, and using the default molecular scale factors. They should be useful in picking a basis of the desired energy accuracy, and estimating the correct molecular total energies.

Atom state	STO-2G	STO-3G	3-21G	6-31G

H	2-S	-.454397	-.466582	-.496199	-.498233
He	1-S	-2.702157	-2.807784	-2.835680	-2.855160
Li	2-S	-7.070809	-7.315526	-7.381513	-7.431236
Be	1-S	-13.890237	-14.351880	-14.486820	-14.566764
B	2-P	-23.395284	-24.148989	-24.389762	-24.519492
C	3-P	-36.060274	-37.198393	-37.481070	-37.677837
N	4-S	-53.093007	-53.719010	-54.105390	-54.385008
O	3-P	-71.572305	-73.804150	-74.393657	-74.780310
F	2-P	-95.015084	-97.986505	-98.845009	-99.360860
Ne	1-S	-122.360485	-126.132546	-127.803825	-128.473877
Na	2-S	-155.170019	-159.797148	-160.854065	-161.841425
Mg	1-S	-191.507082	-197.185978	-198.468103	-199.595219
Al	2-P	-233.199965	-239.026471	-240.551046	-241.854186
Si	3-P	-277.506857	-285.563052	-287.344431	-288.828598
P	4-S	-327.564244	-336.944863	-339.000079	-340.689008
S	3-P	-382.375012	-393.178951	-395.551336	-397.471414
Cl	2-P	-442.206260	-454.546015	-457.276552	-459.442939
Ar	1-S	-507.249273	-521.222881	-524.342962	-526.772151

Atom	state	DH	6-311G	MC	SCF limit *
H	2-S	-.498189	-.499810	--	-0.5
He	1-S	--	-2.859895	--	-2.861680
Li	2-S	-7.431736	-7.432026	--	-7.432727
Be	1-S	-14.570907	-14.571874	--	-14.573023
B	2-P	-24.526601	-24.527020	--	-24.529061
C	3-P	-37.685571	-37.686024	--	-37.688619
N	4-S	-54.397260	-54.397980	--	-54.400935
O	3-P	-74.802707	-74.802496	--	-74.809400
F	2-P	-99.395013	-99.394158	--	-99.409353
Ne	1-S	-128.522354	-128.522553	--	-128.547104
Na	2-S	--	--	-161.845587	-161.858917
Mg	1-S	--	--	-199.606558	-199.614636
Al	2-P	-241.855079	--	-241.870014	-241.876699
Si	3-P	-288.829617	--	-288.847782	-288.854380
P	4-S	-340.689043	--	-340.711346	-340.718798
S	3-P	-397.468667	--	-397.498023	-397.504910
Cl	2-P	-459.435938	--	-459.473412	-459.482088
Ar	1-S	--	--	-526.806626	-526.817528

* M.W.Schmidt and K.Ruedenberg, J.Chem.Phys. 71, 3951-3962(1979). These are ROHF energies in Kh symmetry.

Using spherical functions

Full support of spherical basis functions (also referred to as pure functions) has been implemented in Firefly. By default, spherical functions are disabled. They can be enabled by specifying:

```
$CONTRL D5=.T. $END
```

The $D5 group provides further control through the D5, F7, and G9 keywords, which pertain to d, f, and g functions, respectively. Their default values are .T. meaning that spherical functions are used for these three types of functions, provided of course that spherical functions are enabled in the first place by setting $CONTRL D5=.T.

As an example, if one would like to use Cartesian d functions but spherical f and g functions, one should specify:

```
$CONTRL D5=.T. $END
$D5 D5=.F. $END
```

Whether one should use Cartesian or spherical functions depends on what the basis set used has been designed for. As a general rule, older basis sets are usually designed to use Cartesian d functions, while newer sets should generally be used with spherical d functions. Higher functions (f and g) should almost always be spherical.

The following sets in Firefly should be used with Cartesian d functions:

STO-NG, N-21G, N-31G, SBKJC, HW, DH, DZV, TZV, MINI, and MIDI

As spherical functions are disabled by default, one does not have to specify anything special when augmenting these basis sets with d functions. On the other hand, f functions with these sets are usually spherical. Therefore, when using a basis set such as 6-31G(2df,2p), one should use Cartesian d and spherical f functions (giving input as in the example above).

The N-311G/MC set in Firefly was designed to use spherical polarization functions, so setting $CONTRL D5=.T. for this basis set is a requirement. The same goes for many popular basis sets not incorporated in Firefly such as. Examples are:

- Ahlrichs' SV, TZV, QZV (both def and def2 generation), and all sets derived from these (such as def2-TZVPPD)
- Dunning's cc-pVXZ family and all sets derived from these (such as aug-cc-pVTZ, cc-pwCVTZ, and cc-pVTZ-PP)
- Jensen's pc-X family and all sets derived from these (such as aug-pc-2 and pcS-2)
- LANL2DZdp, LANL2TZ, and LANL08d
- Roos' augmented DZ and TZ ANO, and ANO-RCC
- Sadlej's pVTZ
- The Sapporo family of sets

Not using spherical functions for these sets can result in discrepancies in energies and/or poor SCF convergence.

Note that the current implementation of the D5 option is incompatible with non-standard molecular input frames (i.e. custom orientations of axes).

Using an external basis set file

Firefly can be instructed to get the basis set from an external file by specifying:

```
$BASIS EXTFIL=.T. GBASIS=name $END
```

where “name” is the name of the basis set as specified in the external file. “name” must be a string of 8 or less characters, and should obviously not be identical to any of the internally stored names.

By default, Firefly will assume that the external file is called BASIS.LIB (in uppercase when using Linux) and is present in the same directory as the input file. A different file name (and path) can be specified using the “–b” command line argument, e.g.:

```
firefly -o test.out –b ccpvtz.lib
```

It is also possible to specify only the path to BASIS.LIB:

```
firefly -o test.out –b c:\basis_sets\ (Windows)
firefly -o test.out –b /home/alex/basis_sets/ (Linux)
```

The structure of the external basis set file should be as follows:

```
 element   basis_name
    shell   n_Gauss
      1       exponent   contr_coeffs
      2       exponent   contr_coeffs
      3       exponent   contr_coeffs
      etc...
    shell   n_Gauss
      1       exponent   contr_coeffs
      2       exponent   contr_coeffs
      3       exponent   contr_coeffs
      etc...
<terminate with a blank line>
```

In here,

- **element** is the element’s symbol (as in the periodic table).
- **basis_name** is the basis set name specified with GBASIS.
- **shell** is the shell type. This can be S, P, D, F, G, or L. Here, L defines a Pople style SP shell.
- **n_Gauss** is the number of Gaussian primitives which the shell is made up from.
- **exponent** gives the Gaussian’s exponent.
- **contr_coeffs** are the contraction coefficients. When specifying two or more contraction coefficients for one exponent, the coefficients should be separated by one or more spaces.

As an example, a 6-31+G(d) set on carbon looks as follows:

```
 C  631pGd
   S   6
```

```
     1   3047.5249000    0.0018347
     2    457.3695100    0.0140373
     3    103.9486900    0.0688426
     4     29.2101550    0.2321844
     5      9.2866630    0.4679413
     6      3.1639270    0.3623120
   L   3
     1      7.8682724   -0.1193324    0.0689991
     2      1.8812885   -0.1608542    0.3164240
     3      0.5442493    1.1434564    0.7443083
   L   1
     1      0.1687144    1.0000000    1.0000000
   L   1
     1      0.0438000    1.0000000    1.0000000
   D   1
     1      0.8000000    1.0000000
```

It is also possible to use internally stored names. The above example can, for example, also be written as:

```
C  631pGd
   N31   6
   L   1
     1      0.0438000    1.00000000   1.0000000
   D   1
     1      0.8000000    1.00000000
```

When using internally stored names, the number of Gaussian primitives only has to be specified for Pople sets (i.e. for STO, N21, N31, N311).

Several basis set files ready for use can be downloaded from the downloads section on the Firefly website. Another good source for basis sets is the EMSL Basis Set Exchange, which can be accessed on https://bse.pnl.gov/bse/portal. Here, basis sets obtained in the “GAMESS-US” format are fully compatible with Firefly.

One final tip: though one might choose to make a separate file for each basis set, it is also possible to have several basis sets in the same file, each identified by a unique GBASIS string.

Specifying a basis set in $DATA

In addition to the above basis input methods, it is possible to specify a basis set in $DATA. This can be useful if one would like to use different sets on different atoms of the same element. For example, one may desire to use diffuse functions on one carbon atom, but not on another carbon atom. A $BASIS group should in this case be omitted from the input.

Basis set information in $DATA should be structured in the same way as the external basis set file (see previous section), except that for each atom “element basis_name” should be replaced with the element name (which can be

arbitrary), atomic nuclear charge, and Cartesian coordinates of the atom on which the set is used. As with the external basis file, it is possible to fully specify a set as well use internally stored names.

Note that a basis set input in $DATA can be easily generated by taking the $DATA block in the PUNCH file of an earlier calculation as this block already has the desired format. As an example, if one would like to use a 6-31+G(d,p) basis but remove the diffuse function from one of the carbon atoms, one can first run a test calculation (EXETYP=CHECK) using

```
$BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=1 DIFFSP=.T. $END
```

After this, one can take the $DATA block from the PUNCH file, remove the desired function, and replace the $DATA part in the input file with it (and, as mentioned, the $BASIS group should then be omitted).

Using effective core potentials

Effective core potentials (ECPs), also referred to as pseudopotenials (PPs), can be used instead of basis functions to describe the core part of an element. This has two advantages. First, heavier elements have a lot of core electrons, which would make it necessary to employ a large number of basis functions in order to accurately describe its core. The use of an ECP in such a case will dramatically reduce the number of basis functions of the system. Secondly, from the third row of the periodic table onwards, relativistic effects become increasingly important. However, these cannot be accounted for by a basis set (at least, not without carrying out an additional relativistic calculation). An ECP on the other hand can include relativistic effects, making for a more accurate description for the elements of the lower half of the periodic table. ECPs, together with spin-orbit coupling calculations, are currently the only way to account for relativistic effects as all-electron relativistic calculations such as Douglass-Kroll-Hess and the Zeroth Order Relativistic Approximation (ZORA) are not (yet) possible with Firefly.

Two sets of ECPs are incorporated in Firefly: the Stevens/Basch/Krauss/Jasien/Cundari (SBKJC) potentials which are available for Li to Rn, and the Hay/Wadt (HW) potentials which are available for Na to Xe. These should be used in conjunction with the SBKJC and HW valence-only basis set (e.g. $BASIS GBASIS=SBKJC $END). In addition, it also possible to manually specify ECPs.

Input for ECPs can be given in three different ways.

The first applies only to the SBKJC and HW ECPs included in Firefly. By using the ECP keyword in the group $CONTRL, one of these two ECPs can be chosen. For example:

```
$CONTRL ECP=SBKJC $END
```

The specified potentials will then be used on all atoms given in $DATA they are available for.

If one would prefer more control over which potentials are used on which atoms, one should specify

```
$CONTRL ECP=READ $END
```

and provide an $ECP group. The way the $ECP group should be formatted is explained in detail in the keyword list and will thus not be given fully here. An example for a formic acid molecule looks as follows:

```
$ECP
C-ECP SBKJC
H-ECP NONE
O-ECP SBKJC
O-ECP
H-ECP
$END
```

An ECP should be specified for every (real) atom present in $DATA. “NONE” means that no core electrons will be removed for an element. In the above example, the second oxygen atom uses the same ECP as the first.

Thirdly, it is possible to specify a potential explicitly. For the formaldehyde example above, this could look as follows:

```
$ECP
C-ECP GEN 2 1
1 ----- CARBON U(P) -----
 -0.89371 1 8.56468
2 ----- CARBON U(S)-U(P) -----
  1.92926 0 2.81497
 14.88199 2 8.11296
H-ECP NONE
O-ECP GEN 2 1
1 ----- OXYGEN U(P) -----
 -0.92550 1 16.11718
2 ----- OXYGEN U(S)-U(P) -----
  1.96069 0 5.05348
 29.13442 2 15.95333
O-ECP
H-ECP
$END
```

Here also, the second oxygen copies from the first.

Constructing an $ECP group for a large molecule can be a labor-intensive task, however, there are some scripts which provide help with this. They can be found in the download section on the Firefly website.

Partial linear dependence in a basis set

When using a large basis set, the situation can arise where some basis functions can be written as linear combinations of other basis functions. This happens most often when diffuse functions are used. This situation is referred to as partial linear dependence in the basis set and can be a problem as it can cause numerical instabilities, possibly leading to poor SCF convergence.

When partial linear dependence is detected, Firefly will print the following message:

```
                    * * * WARNING * * *
   THE OVERLAP MATRIX HAS   x EIGENVALUES BELOW 1.0E-05. THE SMALLEST OF
   THESE IS x. THIS INDICATES A PARTIAL LINEAR DEPENDENCE IN YOUR ATOMIC
   BASIS.

   TO OBTAIN SCF CONVERGENCE MAY REQUIRE MORE ACCURATE INTEGRAL EVALUA-
   TION (INTTYP=HONDO, ICUT=11, ITOL=30 IN $CONTRL), MORE ACCURATE DI-
   RECT SCF FOCK MATRIX FORMATION (FDIFF=.FALSE. IN $SCF), OR CHANGING
   CONVERGERS (DIIS=.T. SOSCF=.F. IN $SCF).

   EIGENVALUES BELOW 1.0D-07 PROBABLY WON'T CONVERGE. EIGENVALUES BE-
   TWEEN 1.0D-07 AND 1.0D-06 MAY REQUIRE TIGHTENING OF -NCONV- DENSITY
   CONVERGENCE IN $SCF.

   THE OVERALL DEGREES OF AOS LINEAR INDEPENDENCE ARE:

   --list of values --

   YOU MAY CONSIDER DROPPING ONE OR MORE AOS, STARTING FROM THE END OF
   THIS LIST.
```

When encountering this message, the required course of action depends on severity of the linear dependence. As the output suggests, eigenvalues above 1.0D-07 might converge but require higher accuracy. More accurate integrals ($CONTRL INTTYP=HONDO, ICUT=11) would be a good starting point. As the message suggests one can also disable FDIFF ($SCF FDIFF=.F – this applies to HF, DFT, and GVB convergence), but this should not be necessary as Firefly will disable FDIFF automatically when it encounters difficult convergence. Finally, it is recommended to use tighter convergence criteria as the linear dependence can cause inaccuracies in the results when normal criteria are used. This is important when the calculated wavefunction will be used for further calculations (e.g., gradient after energy, MP2 after HF, TDDFT after DFT).

When the partial linear dependence results in eigenvalues below 1.0D-07, convergence might not be reached. In such a case, the dependence can be decreased by manually removing one or more functions from the basis. Which functions are best to be removed can be seen from the list of values given in the output. Take for example a situation in which the list ends with:

```
   0.2043E-06   -    C  23  S,   SHELL  86,   AO  230
   0.1455E-06   -    C   5  S,   SHELL  20,   AO   62
   0.1147E-06   -    C  12  S,   SHELL  49,   AO  157
```

```
   0.1015E-06   -    C  13  S,   SHELL  53,   AO  170
   0.8695E-07   -    C   1  S,   SHELL   4,   AO   10
```

As can be seen from this list, the smallest eigenvalue originates from shell nr. 4, situated on atom nr. 1. Looking at the atomic basis set printout:

```
      SHELL TYPE PRIM    EXPONENT          CONTRACTION COEFFICIENTS

       C

         1   S    1    3047.524880    0.536345 (  0.001835)
         1   S    2     457.369518    0.989452 (  0.014037)
         1   S    3     103.948685    1.597283 (  0.068843)
         1   S    4      29.210155    2.079187 (  0.232184)
         1   S    5       9.286663    1.774174 (  0.467941)
         1   S    6       3.163927    0.612580 (  0.362312)

         2   L    7       7.868272   -0.399556 ( -0.119332)     1.296082 (
      0.068999)
         2   L    8       1.881289   -0.184155 ( -0.160854)     0.993754 (
      0.316424)
         2   L    9       0.544249    0.516390 (  1.143456)     0.495953 (
      0.744308)

         3   L   10       0.168714    0.187618 (  1.000000)     0.154128 (
      1.000000)

         4   L   11       0.043800    0.068236 (  1.000000)     0.028562 (
      1.000000)
```

it can be seen that shell 4 is a diffuse function on atom 1. The course of action would now be to take the $DATA block from the PUNCH file, remove this function, and restart the calculation with the modified basis. Most likely, convergence will be much better with the function removed. If not, then the whole procedure needs to be repeated until all problematic functions have been eliminated.

Note that if an internally stored basis set was used, the $DATA block from the PUNCH file will use the GBASIS name as opposed to specifying each shell individually (to be more precise, polarization and diffuse functions are specified individually, but the ‘basic’ basis set will be abbreviated as its GBASIS name, for example, as N31 or TZV). This makes it difficult to remove core shells. There currently is no easy way to solve this – one either will have to replace the abbreviation with the individual shells the set consists of, or one should use an external basis set file which does not use abbreviations from the start (hint: Firefly’s basis sets are available from the EMSL basis set exchange, though there is no guarantee that they are exactly the same as the internal sets – be sure to verify this).

Finally, a word of caution: removing functions can of course change the energy, gradient, and other properties of the system under investigation. This should always be kept in mind when resolving a partial linear dependence in such a way. For example, one should be careful comparing the ener-

gies of two isomers when one of the isomers has less functions than the other.

Basis set superposition error (BSSE) correction

A problem inherent to the use of an incomplete basis set with fixed positions (i.e., with basis functions that are centered on the positions of the nuclei) is that the situation can arise where basis functions positioned on one part of the molecular system can be used in the description of other parts of the molecular system. This situation, commonly referred to as basis set superposition error (BSSE), can cause problems as it artificially lowers the total energy of the system, thus causing the calculated energy to be incorrect. This is encountered most often when studying weak interactions, such as van der Waals bonding, between separate molecules. Circumventing the BSSE problem is in practice difficult – as it is the result of basis set incompleteness in combination with fixed positions for basis functions, the only real solutions are to either a) use a basis set close to the complete basis set limit, or b) use a delocalized basis set designed to describe the system as a whole. Both of these solutions are impractical. There are, however, a few ways to correct for the problem. These will be discussed in this section.

The most simple correction that can be applied with Firefly is the counterpoise correction scheme proposed by Boys and Bernardi[1]. In this scheme, the energy error is evaluated by a set of separate energy calculations. Let us, for example, consider a water dimer with water molecules 'A' and 'B'. Because of BSSE, basis functions on A can be used in the description of B and *vice versa*, causing the calculated total energy of the dimer A+B to be too low. The counterpoise correction on the total energy of the dimer can be obtained through four energy calculations. All of these use the geometries of the two water molecules as they are in the dimer (so, one should not reoptimize them). The four energy calculations are:

- the energy of A+B with a basis set on both molecules, but with ghost atoms specified for B; E(A)ab
- the energy of A+B with a basis set on both molecules, but with ghost atoms specified for A; E(B)ab
- the energy of A, using the geometry of A as in A+B; E(A)a
- the energy of B, using the geometry of B as in A+B; E(B)b

Here, a ghost atom is an atom with basis functions but without charge. A ghost atom can be specified by adding a negative sign to the charge as is shown in the example below (where the three atoms of water molecule B are ghost atoms):

```
 O   8.0   -1.411087   -1.022605   -1.001415
 H   1.0   -2.196224   -0.454859   -1.004828
 H   1.0   -1.111711   -1.052794   -1.922420
 O  -8.0    0.616682    0.473774    0.420136
 H  -1.0    0.973421   -0.026190    1.167563
 H  -1.0   -0.051557   -0.112528    0.012392
```

The use of COORD=UNIQUE in $CONTRL is highly recommended.

With the four energy calculations carried out, the correction itself can be obtained as follows:

```
 Ecorr = E(A)ab - E(A)a + E(B)ab - E(B)b
```

Finally, the corrected energy of the dimer can then obtained as follows:

```
 E(AB)ab_corr = E(AB)ab + Ecorr
```

A second approach to correcting for BSSE is the so-called SCF-MI method [2]. This method is a modification of the Roothaan equations that avoids BSSE in intermolecular interaction calculations by expanding each monomer's orbitals using only its own basis set. As a result, the resulting orbitals are not orthogonal. The use of this method is triggered by the presence of the $SCFMI group in the input. Its current implementation is limited to two monomers and only works with restricted Hartree-Fock calculations. Energies and gradients are available, analytical Hessians are not. Furthermore, this type of calculation cannot be run in parallel.

An SCF-MI run requires that the following four keywords are specified:

```
 NA = number of doubly occupied MOs on fragment A
 NB = number of doubly occupied MOs on fragment B
 MA = number of basis functions on fragment A
 MB = number of basis functions on fragment B
```

Note that, in $DATA, all atoms of monomer A must be given before the atoms of monomer B. Additional keywords belonging to the $SCFMI group are discussed in the list of keywords.

A third way of correcting for BSSE is to calculate the BSSE correction by means of a Morokuma energy decomposition calculation[3]. This calculation is requested by specifying RUNTYP=MOROKUMA in $CONTRL and MOROKM=.T. in $MOROKM. In addition, BSSE=.T. and CTPSPL=.T. should be specified in the $MOROKM. During a MOROKM=.T. decomposition, a basis set superposition error is automatically generated by the RVS scheme. Note however that this is not the full Boys counterpoise correction, as is explained in the reference. More information on this type of calculation can be found in the section on Morokuma energy decompositions.

(1) S. F. Boys and F. Bernardi, Mol. Phys. 19, 553 (1970)
(2) "Modification of Roothan Equations to Exclude BSSE from Molecular Interaction Calculations" E. Gianinetti, M. Raimondi, E. Tornaghi Int. J. Quantum Chem. 60, 157 (1996)
(3) R.Cammi, R.Bonaccorsi, J.Tomasi Theoret.Chim.Acta 68, 271-283(1985) and "Energy decomposition analysis for many-body interactions, and application to water complexes" W.Chen, M.S.Gordon J.Phys.Chem. 100, 14316-14328(1996)

Starting orbitals

All methods in Firefly require the selection of a set of starting orbitals. The selection of starting orbitals is controlled through the $GUESS input group. The GUESS keyword selects the type of starting orbitals.

GUESS=HUCKEL (the default for all runs except restarts) and GUESS=HCORE select methods that automatically generate starting orbitals. HUCKEL performs an extended Hückel calculation using a Huzinaga MINI-1 basis that is projected onto the current basis set, and is available for all elements up to Rn. With HCORE, the one electron Hamiltonian is diagonalized in order to obtain the initial guess, a method that can be used for any element. In general, HUCKEL works better, but HCORE might give better results for systems containing transition metals. This is because HUCKEL does not always treat ECPs on transition metals properly.

Orbitals from earlier calculations can be used through GUESS=MOREAD, GUESS=MOSAVED, GUESS=RDMINI, or GUESS=SKIP. MOREAD requests that the starting orbitals are read in from the input file (from a $VEC group). MOSAVED requests that orbitals are read in from the DICTNRY file of an earlier run; this value of GUESS is the default for restarts. RDMINI can be used to read in the $VEC deck from a converged calculation that used GBASIS=MINI without any polarization functions, and project these orbitals onto the current basis. Note that this option should not be used if the current basis involves ECP basis sets. Finally, SKIP skips the initial orbitals selection. Instead, Firefly will assume that the initial orbitals and density matrix are in the DICTNRY file.

All GUESS types except SKIP carry out an orthonormalization of the orbitals, and generate the correct initial density matrix. However, the initial density matrix cannot be generated for CI and MCSCF calculations, so property restarts for these wavefunctions will require 'SKIP'. Other possible uses for SKIP are for a EXETYP=CHECK job, or a RUNTYP=HESSIAN job where the Hessian is supplied. Apart from these scenarios, SKIP is a seldom used option (that may not always work correctly).

Note that all GUESS types (again except SKIP) permit reordering of the orbitals through the NORDER and IORDER keywords. NORDER is used to enable the reordering (through NORDER=1) while IORDER is an array that supplies reordering instructions. As an example, if one would like to switch the positions of orbitals 9 and 11, the reordering instructions can be as follows:

```
NORDER=1 IORDER(9)=11,10,9
```

Alternatively, one could also write

```
NORDER=1 IORDER(9)=11 IORDER(11)=9
```

Finally, it is possible to give the above reorder instructions as:

```
IORDER(9)=-11    or    IORDER(11)=-9
```

which have the same effect.

For UHF orbitals, the IORDER keyword pertains to the alpha orbitals only. Reordering of the beta orbitals can be specified with the JORDER keyword.

Finally, it should be noted that it is possible to project orbitals obtained with a small basis set upon a larger basis set. This is achieved through the EXTRA keyword which instructs Firefly that the larger basis has extra functions. The NEXTRA keyword in $EXTRAF should hereby be used to provide projection instructions.

An example of projection onto a larger basis is given below for a RHF calculation on water. Here, the orbitals of a converged calculation with the cc-pVDZ basis are projected onto an aug-cc-pVDZ basis. The NEXTRA keyword provides a list of the number of Cartesian functions that will be added for each atom (where atom 1 is oxygen, atom 2 is hydrogen, etc.). In the case of symmetry, this array should still be specified for all atoms, as if the point group is set as C1.

In this example, Firefly is instructed that 10 Cartesian functions will be added for oxygen (s, p, and d) and 4 will be added for hydrogen (s and p). Note that, despite the D5=.T. option (which enables the use of spherical functions), 6 functions are requested for the oxygen's d function. This is because Firefly does not explicitly work with spherical basis functions – it works with Cartesian components but limits the entire Cartesian AO space to its pure spherical subspace. In other words, in the case of a d function, all six Cartesian functions are required to form five spherical functions.

```
 $CONTRL SCFTYP=RHF D5=.T. $END
 $SYSTEM TIMLIM=1 MEMORY=500000 $END

 $GUESS GUESS=MOREAD NORB=25 EXTRA=.T. $END
 $EXTRAF NEXTRA(1)=10,4,4 $END

 $DATA
RHF/aug-cc-pVDZ run that uses converged RHF/cc-pVDZ orbitals
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 S   8
  1   11720.00000         0.710000000E-03
  2    1759.00000         0.547000000E-02
  3     400.80000         0.278370000E-01
  4     113.70000         0.104800000
  5      37.03000         0.283062000
  6      13.27000         0.448719000
  7       5.02500         0.270952000
  8       1.01300         0.154580000E-01
 S   8
```

```
 1    11720.0000         -0.160000000E-03
 2     1759.0000         -0.126300000E-02
 3     400.80000         -0.626700000E-02
 4     113.70000         -0.257160000E-01
 5      37.03000         -0.709240000E-01
 6      13.27000         -0.165411000
 7       5.02500         -0.116955000
 8       1.01300          0.557368000
S   1
 1       0.30230          1.000000000
P   3
 1      17.70000          0.430180000E-01
 2       3.85400          0.228913000
 3       1.04600          0.508728000
P   1
 1       0.27530          1.000000000
D   1
 1       1.18500          1.000000000
S   1
 1       0.07896          1.000000000
P   1
 1       0.06856          1.000000000
D   1
 1       0.33200          1.000000000

H           1.0   0.0000000000   0.7565140024   0.1397092302
S   3
 1      13.01000          0.196850000E-01
 2       1.96200          0.137977000
 3       0.44460          0.478148000
S   1
 1       0.12200          1.000000000
P   1
 1       0.72700          1.000000000
S   1
 1       0.02974          1.000000000
P   1
 1       0.14100          1.000000000

 $END
 $VEC
---orbitals from a converged RHF/cc-pVDZ run---
 $END
```

In addition, it is possible to reduce the basis set size by removing functions. This can be requested with the DELETE keyword in $GUESS. The DELIST keyword in $AODEL should hereby be used to specify deletion instructions.

General accuracy switches

There are a few accuracy options that are universal amongst all calculation types. These are discussed in this chapter.

First of all, the choice of integral code is controlled by two keywords the in $CONTRL group: INTTYP and FSTINT. INTTYP selects the integral routine that is used by the 'old' parts of Firefly's code. Two values are possible, namely INTTYP=POPLE and INTTYP=HONDO. POPLE (the default) selects the use of fast Pople-Hehre routines for s, p, and sp integrals whereby HONDO/Rys code is used for all other integrals. HONDO selects the use of the HONDO/Rys code for all integrals. HONDO is somewhat slower than POPLE, but it is also more accurate.

There are a number of scenarios where the use of HONDO is important because POPLE can lead to inaccurate results. One such a scenarios is when diffuse functions are used, as the error in SCF (meaning anything from RHF to MCSCF) energies with POPLE is generally small, but the error in computations that occupy the virtual orbitals may be much larger. In fact, energy errors up to 1D-4 Hartree have been observed for MP2 energy calculations when diffuse functions were used. It is thus recommended to set INTTYP=HONDO for any calculation that uses diffuse functions. A second scenario where the use of INTTYP=HONDO is recommended is in case of a partial linear dependence in the basis.

The FSTINT keyword can be used to select to select the fastints/gencon code, which is a new, faster integral code available for direct runs only. It can be set for many calculation types (for a list, see the Performance chapter) and has an accuracy similar to INTTYP=HONDO, but has not been incorporated throughout all of Firefly yet. That means that many methods that use fastints/gencon will occasionally use the old code specified by INTTYP. As a result, for these runs, INTTYP still remains a relevant keyword and INTTYP=HONDO is still recommended for the scenarios mentioned above.

Then, two other important keywords of the $CONTRL group are ICUT, the cutoff used in deciding which integrals to discard, and ITOL, the cutoff used in deciding which primitives to skip. The default value for ICUT is 9, which is usually fine but not high enough in cases that require a higher accuracy. For such cases, we recommend to increase ICUT to 11 or higher. The standard value of ITOL is 20. As with ICUT, it is possible to increase ITOL for cases in which a higher accuracy is required. However, in our experience ITOL is much less important than ICUT. Therefore, it is typically not necessary to change ITOL's default value.

The choice of diagonalization routine can be specified by the KDIAG keyword (in $SYSTEM and/or in $GUESS). More information on this can be found in the performance chapter.

Accuracy options specific to each method (SCF, MCSCF, CI, etc.) are discussed in the chapters corresponding to these methods.

Semi-empirical methods

Introduction

Semi-empirical methods are based on Hartree-Fock theory, but strive to reduce the computational cost of Hartree-Fock calculations by simplifying the quantum mechanical problem. This is achieved through a number of ways. First, the number of basis functions is reduced by only treating the valence electrons explicitly, hereby using a minimal Slater-type orbitals basis set. A second important approximation is the Zero Differential Overlap (ZDO) approximation, an approach in which various small two-electron repulsion integrals are neglected.

Through these simplifications, semi-empirical calculations run significantly faster than Hartree-Fock calculations though, obviously, at the cost of accuracy. To compensate for the loss of accuracy, semi-empirical methods use parameters which are based on experimental data or quantum mechanical calculations. At present, various semi-empirical methods exist, which differ in which integrals are neglected and what parameters are used.

One popular program for semi-empirical quantum chemistry calculations is MOPAC, written by J. Stewart. To allow semi-empirical calculations to be carried out with Firefly, Firefly contains parts of the MOPAC 6.0 program. The quantum mechanical nature of semi-empirical theory actually makes it quite compatible with the *ab initio* methodology in Firefly. As a result, very little of the MOPAC code has actually been incorporated. The part that has been incorporated is the code that evaluates:

1) the one- and two-electron integrals,
2) the two-electron part of the Fock matrix,
3) the Cartesian energy derivatives, and
4) the ZDO atomic charges and molecular dipole.

Everything else is Firefly: coordinate input (including point group symmetry), the SCF convergence procedures, the matrix diagonalizer, the geometry searcher, the numerical Hessian driver, and so on. Therefore, the output will look mostly like “regular” Firefly output. Semi-empirical methods will however not work with every option in Firefly. It is currently only possible to use RHF, UHF, and ROHF type wavefunctions in any combination with RUNTYP=ENERGY, GRADIENT, OPTIMIZE, SADPOINT, HESSIAN, and IRC (whereby HESSIAN runs use numerical finite differencing instead of calculating the Hessian analytically). In addition, it is possible to use GVB wavefunctions, though this method currently lacks analytical gradients. It should be noted that the CI and half electron methods present in MOPAC are not supported.

The use of empirical parameters gives semi-empirical theory the potential to be more accurate than Hartree-Fock theory, a potential that is frequently realized by some of the more recently proposed methods. It should be stressed though that semi-empirical methods are not nearly as robust as *ab initio* or DFT methods and can perform extremely poorly with more complex systems. As a tip, one good question to ask before using a semi-empirical method is "How well is my system modeled with an *ab initio* minimal basis

sets, such as STO-3G?". If the answer is "not very well" there is a good chance that a semi-empirical description is poor.

On the flipside, semiempirical calculations are very fast. Actually, one of the motives for the MOPAC implementation within Firefly is to take advantage of this speed. Semiempirical models can rapidly provide reasonable starting geometries for *ab initio* or DFT optimizations. Also, a semiempirical Hessian can be obtained at virtually no computational cost and is very useful as a starting Hessian for a geometry optimization (much better than a guessed Hessian, and often almost as useful as a Hessian calculated at a higher level). Simply use HESS=READ in $STATPT to read a semi-empirically obtained $HESS group in at the start of an *ab initio* or DFT geometry optimization.

Available methods

Using a semi-empirical method in Firefly is extremely simple. All one needs to do is specify GBASIS=MNDO, AM1, PM3, or RM1 in the $BASIS group. This not only picks a particular Slater orbital basis, but it also selects a particular "Hamiltonian", namely a particular parameter set. Note that requesting one of these methods causes all other keywords in $BASIS to be ignored (as they have no function).

Information on the four individual methods is given below.

MNDO

The MNDO, or Modified Neglect of Differential Overlap, method was developed by Dewar and Thiel and published in 1977. MNDO has a number of limitations, one of the most important ones being that it calculates the repulsion between two atom 2 to 3 Ångstroms apart as too high. As a result, it does not perform well for modeling hydrogen bonds and calculating activation energies. Nowadays, it has been surpassed by newer methods with respect to accuracy.

AM1

The AM1, or Austin Model 1, method was developed by Dewar and coworkers, and published in 1985. It improves upon the MNDO method by using Gaussian functions to improve the core-core interaction function and by being reparameterized. As a result, it is overall more accurate than the MNDO method.

PM3

Parametric Model 3 was developed by J. Stewart and published in 1989. It is very similar to the AM1 method, the most important difference being that it uses different, more optimized parameters. It usually outperforms the MNDO and AM1 methods with respect to accuracy, though there are cases where AM1 still performs better.

RM1

Recife Model 1 was developed by J. Stewart and coworkers and published in 2006. It is a re-parameterized version of AM1 with the intent of improving its accuracy for organic molecules and biochemical systems. For these systems, it is generally more accurate than AM1 and PM3.

The above four methods can be used for the following elements. Note that for MNDO, Na and K are so-called "sparkles". This means that a basis is defined for these atoms have a basis set, however, it is not used in any calculations.

For MNDO

H										
Li	Be					B	C	N	O	F
Na'	Mg					Al	Si	P	S	Cl
K'	Ca	...	Cr	...	Zn	*	Ge	*	*	Br
*	*	...	..	...	*	*	Sn	*	*	I
*	*	...	..	...	Hg	*	Pb	*	*	*

(' = sparkle)

For AM1:

H										
Li'	Be'					B	C	N	O	F
Na	Mg					Al	Si	P	S	Cl
K	Ca	...	..	...	Zn	*	Ge	*	*	Br
*	*	...	..	...	*	*	*	*	*	I
*	*	...	..	...	Hg	*	*	*	*	*

(' = MNDO parameters are used)

For PM3

H								
Li	Be			*	C	N	O	F
Na	Mg			Al	Si	P	S	Cl
K	Ca	... .	Zn	Ga	Ge	As	Se	Br
*	*	... .	Cd	In	Sn	Sb	Te	I
*	*	... .	Hg	Tl	Pb	Bi	*	*

For RM1

H								
Li'	Be'			B"	C	N	O	F
Na"	Mg"			Al"	Si"	P	S	Cl
K"	Ca"	... .	Zn"	*	Ge"	*	*	Br
*	*	... .	*	*	*	*	*	I
*	*	... .	Hg"	*	*	*	*	*

(' = MNDO parameters are used)
(" = AM1 parameters are used)

The AM1 and RM1 methods use parameters from other methods for some elements. Use this with caution – mixing parameters from different methods has not been fully tested and is therefore not recommended.

Hartree-Fock

Introduction

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Four SCF wavefunction methods are present in Firefly: restricted Hartree-Fock (RHF), unrestricted Hartree-Fock (UHF), restricted-open Hartree-Fock (ROHF), and generalized valence bond (GVB). This chapter discusses RHF, UHF, and ROHF, as well as direct vs conventional runs and options pertaining to the convergence procedure. GVB is discussed in a separate chapter.

RHF, UHF, and ROHF all have an intrinsic N^4 time dependence, because they are all driven by integrals in the AO basis. Analytic gradients are implemented for all three, and therefore numerical Hessians are also available for each. Analytic Hessian calculations are implemented for RHF, ROHF, but not for UHF. Analytic Hessians are more accurate, and computed much more quickly than numerical Hessians, but require additional disk storage to perform the integral transformation, and also more physical memory.

Restricted Hartree-Fock

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Restricted HF calculations are requested with SCFTYP=RHF in the $CONTRL group.

Unrestricted Hartree-Fock

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Restricted HF calculations are requested with SCFTYP=UHF in the $CONTRL group. The multiplicity of the system can be controlled with the MULT keyword, also of $CONTRL.

Restricted-open Hartree-Fock

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Open shell SCF calculations in Firefly can be performed by both the ROHF code and the GVB code. Note that when the GVB code is executed with no

pairs, the run is NOT a true GVB run, and should be referred to in publications and discussion as a ROHF calculation.

The ROHF module in Firefly can handle any number of open shell electrons, provided these have a high spin coupling. Some commonly occurring cases are:

one open shell, doublet:

```
 $CONTRL SCFTYP=ROHF MULT=2 $END
```

two open shells, triplet:

```
 $CONTRL SCFTYP=ROHF MULT=3 $END
```

m open shells, high spin:

```
 $CONTRL SCFTYP=ROHF MULT=m+1 $END
```

The Fock matrix in the MO basis has the form

	closed	open	virtual
closed	F2	Fb	(Fa+Fb)/2
open	Fb	F1	Fa
virtual	(Fa+Fb)/2	Fa	F0

where Fa and Fb are the usual alpha and beta Fock matrices any UHF program produces. The Fock operators for the doubly, singly, and zero occupied blocks can be written as

```
F2 = Acc*Fa + Bcc*Fb
F1 = Aoo*Fa + Boo*Fb
F0 = Avv*Fa + Bvv*Fb
```

Some choices found in the literature for these canonicalization coefficients are

	Acc	Bcc	Aoo	Boo	Avv	Bvv
Guest and Saunders	1/2	1/2	1/2	1/2	1/2	1/2
Roothaan single matrix	-1/2	3/2	1/2	1/2	3/2	-1/2
Davidson	1/2	1/2	1	0	1	0
Binkley, Pople, Dobosh	1/2	1/2	1	0	0	1
McWeeny and Diercksen	1/3	2/3	1/3	1/3	2/3	1/3
Faegri and Manne	1/2	1/2	1	0	1/2	½

The choice of the diagonal blocks is arbitrary, as ROHF is converged when the off diagonal blocks go to zero. The exact choice for these blocks can however have an effect on the convergence rate. This choice also affects the MO coefficients, and orbital energies, as the different choices produce different canonical orbitals within the three subspaces. All methods, how-

ever, will give identical total wavefunctions, and hence identical properties such as gradients and Hessians.

The default coupling case in Firefly is the Roothaan single matrix set. If one would like to try any other canonicalizations, the Acc, Aoo, Avv and Bcc, Boo, Bvv parameters can be input as the first three elements of ALPHA and BETA in $SCF.

Direct vs conventional

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Direct SCF is implemented for every possible HF type calculation. The direct SCF method may not be used with DEM convergence. Direct SCF may be used during energy, gradient, numerical or analytic Hessian, CI or MP2 energy correction, or localized orbitals computations.

Normally, HF calculations proceed by evaluating a large number of two electron repulsion integrals, and storing these on a disk. This integral file is read back in during each HF iteration to form the appropriate Fock operators. In a direct HF, the integrals are not stored on disk, but are instead reevaluated during each HF iteration. The default for DIRSCF in $SCF is .FALSE.

You can estimate the disk storage requirements for conventional HF using a P or PK file by the following formulae:

```
nint = 1/sigma * 1/8 * N**4
Mbytes = nint * x / 1024**2
```

Here N is the total number of basis functions in your run, which you can learn from an EXETYP=CHECK run. The 1/8 accounts for permutational symmetry within the integrals. Sigma accounts for the point group symmetry, and is difficult to estimate accurately. Sigma cannot be smaller than 1, in no symmetry (C1) calculations. For benzene, sigma would be almost six, since you generate 6 C's and 6 H's by entering only 1 of each in $DATA. For water sigma is not much larger than one, since most of the basis set is on the unique oxygen, and the C2v symmetry applies only to the H atoms. The factor x is 12 bytes per integral for RHF, and 20 bytes per integral for ROHF, UHF, and GVB. Finally, since integrals very close to zero need not be stored on disk, the actual power dependence is not as bad as N**4, and in fact in the limit of very large molecules can be as low as N**2. Thus plugging in sigma=1 should give you an upper bound to the actual disk space needed. If the estimate exceeds your available disk storage, your only recourse is direct HF.

What are the economics of direct HF? Naively, if we assume the run takes 10 iterations to converge, we must spend 10 times more CPU time doing the integrals on each iteration. However, we do not have to waste any CPU time reading blocks of integrals from disk, or in unpacking their indices. We

also do not have to waste any wall clock time waiting for a relatively slow mechanical device such as a disk to give us our data.

There are some less obvious savings too, as first noted by Almlof. First, since the density matrix is known while we are computing integrals, we can use the Schwarz inequality to avoid doing some of the integrals. In a conventional SCF this inequality is used to avoid doing small integrals. In a direct SCF it can be used to avoid doing integrals whose contribution to the Fock matrix is small (density times integral=small). Secondly, we can form the Fock matrix by calculating only its change since the previous iteration. The contributions to the change in the Fock matrix are equal to the change in the density times the integrals. Since the change in the density goes to zero as the run converges, we can use the Schwarz screening to avoid more and more integrals as the calculation progresses. The input option FDIFF in $SCF selects formation of the Fock operator by computing only its change from iteration to iteration. The FDIFF option is not implemented for GVB since there are too many density matrices from the previous iteration to store, but is the default for direct RHF, ROHF, and UHF.

So, in our hypothetical 10 iteration case, we do not spend as much as 10 times more time in integral evaluation. Additionally, the run as a whole will not slow down by whatever factor the integral time is increased. A direct run spends no additional time summing integrals into the Fock operators, and no additional time in the Fock diagonalizations. So, generally speaking, a RHF run with 10-15 iterations will slow down by a factor of 2-4 times when run in direct mode. The energy gradient time is unchanged by direct HF, and this is a large time compared to HF energy, so geometry optimizations will be slowed down even less. This is really the converse of Amdahl's law: if you slow down only one portion of a program by a large amount, the entire program slows down by a much smaller factor.

Convergence options

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Generally speaking, the simpler the function, the better its convergence. In our experience, the majority of RHF, ROHF, and UHF runs will converge readily from GUESS=HUCKEL. GVB runs typically require GUESS=MOREAD, although the Hückel guess usually works for NPAIR=0. RHF convergence is the best, closely followed by ROHF. In the current implementation in Firefly, ROHF is always better convergent than the closely related unrestricted high spin UHF. GVB calculations require much more care, and cases with NPAIR greater than one are particularly difficult.

Unfortunately, not all HF runs converge readily. The best way to improve your convergence is to provide better starting orbitals! In many cases, this means to MOREAD orbitals from a simpler HF case. For example, if you want to do a doublet ROHF, and the HUCKEL guess does not seem to converge, try to do an RHF calculation on the +1 cation. RHF is typically more stable than ROHF, UHF, or GVB, and cations are usually readily convergent. Then

MOREAD the cation's orbitals into the neutral calculation which you wanted to do at first.

GUESS=HUCKEL does not always guess the correct electronic configuration. It may be useful to use PRTMO in $GUESS during a EXETYP=CHECK run to examine the starting orbitals, and then reorder them with NORDER if that seems appropriate.

Of course, by default Firefly uses the convergence procedures which are usually most effective. Still, there are cases which are difficult, so the $SCF group permits you to select several alternative methods for improving convergence. Briefly, these are:

EXTRAP. This extrapolates the three previous Fock matrices, in an attempt to jump ahead a bit faster. This is the most powerful of the old-fashioned accelerators, and normally should be used at the beginning of any SCF run. When an extrapolation occurs, the counter at the left of the SCF printout is set to zero.

DAMP. This damps the oscillations between several successive Fock matrices. It may help when the energy is seen to oscillate wildly. Thinking about which orbitals should be occupied initially may be an even better way to avoid oscillatory behaviour.

SHIFT. This shifts the diagonal elements of the virtual part of the Fock matrix up, in an attempt to uncouple the unoccupied orbitals from the occupied ones. At convergence, this has no effect on the orbitals, just their orbital energies, but will produce different (and hopefully better) orbitals during the iterations.

RSTRCT. This limits mixing of the occupied orbitals with the empty ones, especially the flipping of the HOMO and LUMO to produce undesired electronic configurations or states. This should be used with caution, as it makes it very easy to converge on incorrect electronic configurations, especially if DIIS is also used. If you use this, be sure to check your final orbital energies to see if they are sensible. A lower energy for an unoccupied orbital than for one of the occupied ones is a sure sign of problems.

DIIS. Direct Inversion in the Iterative Subspace is a modern method, due to Pulay, using stored error and Fock matrices from a large number of previous iterations to interpolate an improved Fock matrix. This method was developed to improve the convergence at the final stages of the SCF process, but turns out to be quite powerful at forcing convergence in the initial stages of SCF as well. By giving ETHRSH as 10.0 in $SCF, you can practically guarantee that DIIS will be in effect from the first iteration. The default is set up to do a few iterations with conventional methods (extrapolation) before engaging DIIS. This is because DIIS can sometimes converge to solutions of the SCF equations that do not have the lowest possible energy. For example, the 3-A-2 small angle state of SiLi2 (see M.S.Gordon and M.W.Schmidt, Chem.Phys.Lett., 132, 294-8(1986)) will readily converge with DIIS to a solution with a reasonable S**2, and an energy about 25 milli-Hartree above the correct answer. A SURE SIGN OF TROUBLE WITH DIIS IS WHEN THE ENERGY RISES TO ITS FINAL VALUE. However, if you obtain orbitals at one point on a PES without DIIS, the subsequent use of DIIS with MOREAD will

probably not introduce any problems. Because DIIS is quite powerful, EXTRAP, DAMP, and SHIFT are all turned off once DIIS begins to work. DEM and RSTRCT will still be in use, however.

SOSCF. Approximate second-order (quasi-Newton) SCF orbital optimization. SOSCF will converge about as well as DIIS at the initial geometry, and slightly better at subsequent geometries. There's a bit less work solving the SCF equations, too. The method kicks in after the orbital gradient falls below SOGTOL. Some systems, particularly transition metals with ECP basis sets, may have Huckel orbitals for which the gradient is much larger than SOGTOL. In this case it is probably better to use DIIS instead, with a large ETHRSH, rather than increasing SOGTOL, since you may well be outside the quadratic convergence region. SOSCF does not exhibit true second-order convergence since it uses an approximation to the inverse Hessian. SOSCF will work for MOPAC runs, but is slower in this case. SOSCF will work for UHF, but the convergence is slower than DIIS. SOSCF will work for non-Abelian ROHF cases, but may encounter problems if the open shell is degenerate.

DEM. Direct energy minimization should be your last recourse. It explores the "line" between the current orbitals and those generated by a conventional change in the orbitals, looking for the minimum energy on that line. DEM should always lower the energy on every iteration, but is very time consuming, since each of the points considered on the line search requires evaluation of a Fock operator. DEM will be skipped once the density change falls below DEMCUT, as the other methods should then be able to affect final convergence. While DEM is working, RSTRCT is held to be true, regardless of the input choice for RSTRCT. Because of this, it behooves you to be sure that the initial guess is occupying the desired orbitals. DEM is available only for RHF. The implementation in Firefly resembles that of R. Seeger and J. A. Pople, J. Chem. Phys. 65, 265-271 (1976). Simultaneous use of DEM and DIIS resembles the ADEM-DIOS method of H. Sellers, Chem. Phys. Lett. 180, 461-465 (1991). DEM does not work with direct SCF.

General valence bond

Introduction

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Just as RHF, UHF, and ROHF, GVB wavefunctions have an intrinsic N^4 time dependence. Analytic gradients are implemented, and therefore numerical Hessians are also available for each. Analytic Hessian calculations are implemented only GVB cases with NPAIR=0 or NPAIR=1. Information on the convergence accelerators and conventional *vs* direct runs can be found in the chapter on Hartree-Fock.

Open-shell SCF with GVB

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Genuine GVB-PP runs will be discussed later in this Section. First, we will consider how to do open shell SCF with the GVB part of the program.

It is possible to do other open shell cases with the GVB code, which can handle the following cases:

```
one open shell, doublet:
     $CONTRL SCFTYP=GVB MULT=2 $END
     $SCF    NCO=xx NSETO=1 NO(1)=1 $END
two open shells, triplet:
     $CONTRL SCFTYP=GVB MULT=3 $END
     $SCF    NCO=xx NSETO=2 NO(1)=1,1 $END
two open shells, singlet:
     $CONTRL SCFTYP=GVB MULT=1 $END
     $SCF    NCO=xx NSETO=2 NO(1)=1,1 $END
```

Note that the first two cases duplicate runs which the ROHF module can do better. Note that all of these cases are really ROHF, since the default for NPAIR in $SCF is 0.

Many open shell states with degenerate open shells (for example, in diatomic molecules) can be treated as well.

If you would like to do any cases other than those shown above, you must derive the coupling coefficients ALPHA and BETA, and input them with the occupancies F in the $SCF group.

Mariusz Klobukowski of the University of Alberta has shown how to obtain coupling coefficients for the GVB open shell program for many such open

shell states. These can be derived from the values in Appendix A of the book "A General SCF Theory" by Ramon Carbo and Josep M. Riera, Springer-Verlag (1978). The basic rule is

```
       (1)      F(i) = 1/2 * omega(i)
       (2)  ALPHA(i) =       alpha(i)
       (3)   BETA(i) =     - beta(i),
```

where omega, alpha, and beta are the names used by Ramon in his Tables.

The variable NSETO should give the number of open shells, and NO should give the degeneracy of each open shell. Thus the 5-S state of carbon would have NSETO=2, and NO(1)=1,3.

Some specific examples, for the lowest term in each of the atomic P**N configurations are

```
!   p**1   2-P state
 $CONTRL SCFTYP=GVB  MULT=2   $END
 $SCF    NCO=xx   NSETO=1  NO=3   COUPLE=.TRUE.
      F(1)=  1.0  0.16666666666667
  ALPHA(1)=  2.0  0.33333333333333  0.00000000000000
   BETA(1)= -1.0 -0.16666666666667 -0.00000000000000  $END

!   p**2   3-P state
 $CONTRL SCFTYP=GVB  MULT=3   $END
 $SCF    NCO=xx   NSETO=1  NO=3   COUPLE=.TRUE.
      F(1)=  1.0  0.333333333333333
  ALPHA(1)=  2.0  0.66666666666667  0.16666666666667
   BETA(1)= -1.0 -0.33333333333333 -0.16666666666667  $END

!   p**3   4-S state
 $CONTRL SCFTYP=ROHF  MULT=4   $END

!   p**4   3-P state
 $CONTRL SCFTYP=GVB  MULT=3   $END
 $SCF    NCO=xx   NSETO=1  NO=3   COUPLE=.TRUE.
      F(1)=  1.0  0.66666666666667
  ALPHA(1)=  2.0  1.33333333333333  0.83333333333333
   BETA(1)= -1.0 -0.66666666666667 -0.50000000000000  $END

!   p**5   2-P state
 $CONTRL SCFTYP=GVB  MULT=2   $END
 $SCF    NCO=xx   NSETO=1  NO=3   COUPLE=.TRUE.
      F(1)=  1.0  0.83333333333333
  ALPHA(1)=  2.0  1.66666666666667  1.33333333333333
   BETA(1)= -1.0 -0.83333333333333 -0.66666666666667  $END
```

   Be sure to give all the digits, as these are part of a double precision energy formula.

```
Coupling constants for d**N configurations are

!     d**1   2-D state
 $CONTRL SCFTYP=GVB MULT=2 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.1
         ALPHA(1)= 2.0, 0.20, 0.00
          BETA(1)=-1.0,-0.10, 0.00  $END

!     d**2   average of 3-F and 3-P states
 $CONTRL SCFTYP=GVB MULT=3 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.2
         ALPHA(1)= 2.0, 0.40, 0.05
          BETA(1)=-1.0,-0.20,-0.05  $END

!     d**3   average of 4-F and 4-P states
 $CONTRL SCFTYP=GVB MULT=4 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.3
         ALPHA(1)= 2.0, 0.60, 0.15
          BETA(1)=-1.0,-0.30,-0.15  $END

!     d**4   5-D state
 $CONTRL SCFTYP=GVB MULT=5 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.4
         ALPHA(1)= 2.0, 0.80, 0.30
          BETA(1)=-1.0,-0.40,-0.30 $END

!     d**5   6-S state
 $CONTRL SCFTYP=ROHF MULT=6 $END

!     d**6   5-D state
 $CONTRL SCFTYP=GVB MULT=5 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.6
         ALPHA(1)= 2.0, 1.20, 0.70
          BETA(1)=-1.0,-0.60,-0.50 $END

!     d**7   average of 4-F and 4-P states
 $CONTRL SCFTYP=GVB MULT=4 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.7
         ALPHA(1)= 2.0, 1.40, 0.95
          BETA(1)=-1.0,-0.70,-0.55  $END

!     d**8   average of 3-F and 3-P states
 $CONTRL SCFTYP=GVB MULT=3 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.8
         ALPHA(1)= 2.0, 1.60, 1.25
          BETA(1)=-1.0,-0.80,-0.65  $END

!     d**9   2-D state
 $CONTRL SCFTYP=GVB MULT=2 $END
 $SCF    NCO=xx NSETO=1 NO=5 COUPLE=.TRUE.  F(1)=1.0,0.9
         ALPHA(1)= 2.0, 1.80, 1.60
          BETA(1)=-1.0,-0.90,-0.80 $END
```

The source for these values is R.Poirier, R.Kari, and I.G.Csizmadia's book "Handbook of Gaussian Basis Sets", Elsevier, Amsterdam, 1985.

Note that Firefly can do a proper calculation on the ground terms for the d**2, d**3, d**7, and d**8 configurations only by means of state averaged MCSCF. For d**8, use

```
 $CONTRL SCFTYP=MCSCF MULT=3 $END
 $DRT    GROUP=C1 FORS=.TRUE. NMCC=xx NDOC=3 NALP=2 $END
 $GUGDIA NSTATE=10 $END
 $GUGDM2 WSTATE(1)=1,1,1,1,1,1,1,0,0,0 $END
```

Open shell cases such as s**1,d**n are probably most easily tackled with the state-averaged MCSCF program.

True GVB perfect pairing runs

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

True GVB runs are obtained by choosing NPAIR nonzero. If you wish to have some open shell electrons in addition to the geminal pairs, you may add the pairs to the end of any of the GVB coupling cases shown above. The GVB module assumes that you have reordered your MOs into the order: NCO double occupied orbitals, NSETO sets of open shell orbitals, and NPAIR sets of geminals (with NORDER=1 in the $GUESS group).

Each geminal consists of two orbitals and contains two singlet coupled electrons (perfect pairing). The first MO of a geminal is probably heavily occupied (such as a bonding MO u), and the second is probably weakly occupied (such as an antibonding, correlating orbital v). If you have more than one pair, you must be careful that the initial MOs are ordered u1, v1, u2, v2..., which is -NOT- the same order that RHF starting orbitals will be found in. Use NORDER=1 to get the correct order.

These pair wavefunctions are actually a limited form of MCSCF. GVB runs are much faster than MCSCF runs, because the natural orbital u,v form of the wavefunction permits a Fock operator based optimization. However, convergence of the GVB run is by no means assured. The same care in selecting the correlating orbitals that you would apply to an MCSCF run must also be used for GVB runs. In particular, look at the orbital expansions when choosing the starting orbitals, and check them again after the run converges.

GVB runs will be carried out entirely in orthonormal natural u,v form, with strong orthogonality enforced on the geminals. Orthogonal orbitals will pervade your thinking in both initial orbital selection, and the entire orbital optimization phase (the CICOEF values give the weights of the u,v orbitals in each geminal). However, once the calculation is converged, the program will generate and print the nonorthogonal, generalized valence bond orbitals. These GVB orbitals are an entirely equivalent way of presenting the wavefunction, but are generated only after the fact.

Convergence of true GVB runs is by no means as certain as convergence of RHF, UHF, ROHF, or GVB with NPAIR=0. You can assist convergence by doing a preliminary RHF or ROHF calculation, and use these orbitals for GUESS=MOREAD. Few, if any, GVB runs with NPAIR non-zero will converge without using GUESS=MOREAD. Generation of MVOs during the prelimnary SCF can also be advantageous. In fact, all the advice outlined for MCSCF computations below is germane, for GVB-PP is a type of MCSCF computation.

The total number of electrons in the GVB wavefunction is given by the following formula:

```
NE = 2*NCO + sum 2*F(i)*NO(i) + 2*NPAIR
```

The charge is obtained by subtracting the total number of protons given in $DATA. The multiplicity is implicit in the choice of alpha and beta constants. Note that ICHARG and MULT must be given correctly in $CONTRL anyway, as the number of electrons from this formula is double checked against the ICHARG value.

The special case of TCSCF

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

The wavefunction with NSETO=0 and NPAIR=1 is called GVB-PP(1) by Goddard, two configuration SCF (TCSCF) by Schaefer or Davidson, and CASSCF with two electrons in two orbitals by others. Note that this is just semantics, as these are all identical. This is a very important type of wavefunction, as TCSCF is the minimum acceptable treatment for singlet biradicals. The TCSCF wavefunction can be obtained with SCFTYP=MCSCF, but it is usually much faster to use the Fock based SCFTYP=GVB. Because of its importance, the TCSCF function (if desired, with possible open shells) permits analytic Hessian computation.

A caution about symmetry

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Caution! Some exotic calculations with the GVB program do not permit the use of symmetry. The symmetry algorithm in Firefly was "derived assuming that the electronic charge density transforms according to the completely symmetric representation of the point group", Dupuis/King, JCP, 68, 3998 (1978). This may not be true for certain open shell cases, and in fact during GVB runs, it may not be true for closed shell singlet cases!

First, consider the following correct input for the singlet-delta state of NH:

```
 $CONTRL SCFTYP=GVB NOSYM=1 $END
 $SCF NCO=3 NSETO=2 NO(1)=1,1 $END
```

for the x**1y**1 state, or for the x**2-y**2 state,

```
 $CONTRL SCFTYP=GVB NOSYM=1 $END
 $SCF NCO=3 NPAIR=1 CICOEF(1)=0.707,-0.707 $END
```

Neither gives correct results, unless you enter NOSYM=1.

The electronic term symbol is degenerate, a good tip off that symmetry cannot be used. However, some degenerate states can still use symmetry, because they use coupling constants averaged over all degenerate states within a single term, as is done in EXAM15 and EXAM16. Here the "state averaged SCF" leads to a charge density which is symmetric, and these runs can exploit symmetry.

Secondly, since GVB runs exploit symmetry for each of the "shells", or type of orbitals, some calculations on totally symmetric states may not be able to use symmetry. An example is CO or N2, using a three pair GVB to treat the sigma and pi bonds. Individual configurations such as (sigma)**2,(pi-x)**2,(pi-y*)**2 do not have symmetric charge densities since neither the pi nor pi* level is completely filled. Correct answers for the sigma-plus ground states result only if you input NOSYM=1.

Problems of the type mentioned should not arise if the point group is Abelian (C_1, C_2, C_i, C_s, C_{2v}, C_{2h}, D_2, and D_{2h}), but will be fairly common in linear molecules. Since Firefly cannot detect that the GVB electronic state is not totally symmetric (or averaged to at least have a totally symmetric density), it is left up to you to decide when to input NOSYM=1. If you have any question about the use of symmetry, try it both ways. If you get the same energy, both ways, it remains valid to use symmetry to speed up your run.

And beware! Brain dead computations, such as RHF on singlet O2, which actually is a half-filled degenerate shell, violate the symmetry assumptions, and also violate nature. Use of partially filled degenerate shells always leads to very wild oscillations in the RHF energy, which is how the program tries to tell you to think first, and compute second. Configurations such as pi**2, e**1, or f2u**4 can be treated, but require GVB wavefunctions and F, ALPHA, BETA values from the sources mentioned.

Møller-Plesset correlation corrections

Introduction

One important approximation made in the Hartree-Fock model is that, for each electron, the interaction of this electron with the other electrons in a system is simplified by considering only an average interaction. As a result, the energy of a system as obtained with the Hartree-Fock model is higher than the energy that would be obtained if all electron-electron interactions would individually be taken into account. This difference in energy is commonly referred to as the dynamic electron correlation energy.

One theory for recovering this dynamic correlation energy that is absent from the Hartree-Fock solution is Møller-Plesset perturbation theory. In Firefly, it is possible to perform second-order, third-order, and fourth-order Møller-Plesset energy corrections (commonly abbreviated as MP2, MP3, and MP4), where higher-order methods obtain more of the missing correlation energy than lower-order methods (but also require more computational resources). MP2 is implemented for RHF, UHF, and ROHF wavefunctions with analytic gradients available only with RHF. MP3 and MP4 theories on the other hand are implemented for RHF wavefunctions only and can only be used to obtain energies.

The MP2 code in Firefly has been optimized for performance to a great extent. MP2 performance is discussed in a section of its own, and it is recommended to read through this section before commencing with MP2 calculations.

Second-order Møller-Plesset theory is also implemented for MCSCF wavefunctions in the form of MRMP2 and (X)MCQDPT2 theory. This theory is discussed in the (X)MCQDPT2 chapter.

MP2 calculations

MP2 calculations can be requested by specifying MPLEVL=2 in $CONTROL, in combination with SCFTYP=RHF, UHF, or ROHF. Keywords that pertain to MP2 calculations are found in the $MP2 group. Additionally, keywords in the $MP2GRD group can be used to control the calculation of gradients.

There is little that has to be said with respect to RHF and UHF MP2 calculations. The most important thing is that the description of the system under investigation should already be good at the zeroth-order level (i.e. at the Hartree-Fock level). If this is not the case, it is probably better to use MCSCF theory and recover the correlation energy using (X)MCQDPT2.

One point which may not be commonly appreciated is that the density matrix for the first-order wavefunction for the RHF case, which is generated during gradient runs or if properties are requested in the $MP2 group (MP2PRP=.TRUE.), is of the type known as "response density", which differs

from the more usual "expectation value density". The eigenvalues of the response density matrix (which are the occupation numbers of the MP2 natural orbitals) can therefore be greater than 2 for frozen core orbitals, or even negative values for the highest 'virtual' orbitals. The sum is of course exactly the total number of electrons. We have seen values outside the range 0-2 in several cases where the single configuration RHF wavefunction was not an appropriate description of the system, and thus these occupancies may serve as a guide to the wisdom of using a RHF reference.

The case of ROHF MP2 deserves more explanation. There are a number of open shell perturbation theories described in the literature. It is important to note that these methods give different results for the second-order energy correction, reflecting ambiguities in the selection of the zeroth-order Hamiltonian and in defining the ROHF Fock matrices. Two of these are available in Firefly.

One theory is known as RMP which, it should be pointed out, is entirely equivalent to the ROHF-MBPT2 method. This theory is as UHF-like as possible and can be chosen by selection of OSPT=RMP in $MP2 (though this is actually not necessary as it is the default method). The RMP method diagonalizes the alpha and beta Fock matrices separately so that their occupied-occupied and virtual-virtual blocks are canonicalized. This generates two distinct orbital sets whose double excitation contributions are processed by the usual UHF MP2 program, but an additional energy term from single excitations is required.

RMP's use of different orbitals for different spins adds to the CPU time required for integral transformations, of course. RMP is invariant under all of the orbital transformations for which the ROHF itself is invariant. Unlike UHF MP2, the second-order RMP energy does not suffer from spin contamination, since the reference ROHF wavefunction has no spin contamination. The RMP wavefunction, however, is spin contaminated at 1st and higher order, and therefore the 3rd and higher order RMP energies are spin contaminated

The ZAPT (Z-averaged perturbation theory) formalism is also implemented in Firefly, and can be requested by specifying OSPT=ZAPT in $MP2. Characteristics of this theory is that it is partially noninvariant, is not spin contaminated at any order, and has only a single set of orbitals in the MO transformation. It should be noted that, at present, the new MP2 code can only execute ROHF MP2 calculations of the RMP kind. ZAPT2 calculations should be run with old MP2 code (see the section on MP2 performance for more information on the new MP2 code). In addition, these calculations cannot be run in parallel.

There are a number of other open shell theories with names such as HC, OPT1, OPT2, and IOPT. These are not implemented in Firefly but results equivalent to the results of these methods can be obtained by using GUGA-style MCSCF followed by (X)MCQDPT2. The same is true for GVB-based MP2. For example, one could use a $DRT input such as

```
NMCC=N/2-1 NDOC=0 NAOS=1 NBOS=1 NVAL=0
```

which generates a single CSF if the two open shells have different symmetry. For a one pair GVB function, one can specify

```
NMCC=N/2-1 NDOC=1 NVAL=1
```

which generates a 3 CSF function entirely equivalent to the two configuration TCSCF, also known as GVB-PP(1). And if one would attempt a triplet state with the GUGA MCSCF program

```
NMCC=N/2-1 NDOC=0 NALP=2 NVAL=0
```

one would get a result equivalent to the OPT1 open shell method instead of the RMP result. For details on $DRT input, please see the chapter on MCSCF.

A feature new to Firefly 8.0.0 is the possibility to scale the two MP2 spin components by certain factors. This functionality is controlled through the SCS keyword and is programmed for RHF, UHF, and ROHF (though only RMP, not ZAPT2). Currently, only energies are available. Input should be of the form:

```
 $MP2 SCS(1)=singlet_pairs_multiplier,triplet_pairs_multiplier $END
```

As an example, Grimme's SCS-MP2 scheme can be specified as:

```
 $MP2 SCS(1)=1.2,0.333333333333333 $END
```

while the SOS-MP2 scheme from Head-Gordon and co-workers can be specified as:

```
 $MP2 SCS(1)=1.3,0 $END
```

Note that use of spin scaling forces the use of MP2 METHOD=1 code (see next section), which is faster than Laplace-transform based code even for SOS-MP2.

Performance of the MP2 code

Over the years, the MP2 code in Firefly has been optimized greatly. Therefore, some information with respect to performance should be given.

One of the most important things to note is that, in comparison to the GAMESS(US) code on which Firefly is originally based, Firefly has a new RHF/ROHF/UHF MP2 energy program which is designed to handle large systems (e.g., 500 AOs or more). It is direct, very fast, and requires much less memory compared to other MP2 methods. Use of this new program can be requested by specifying METHOD=1 in $MP2 and is recommended for all medium and large jobs. For small jobs (100 to 150 basis functions), the old method is still faster due to less overhead (the new program always requires the 2-electron AO integrals to be reevaluated four times, regardless of the

size of the job). It is for this reason that the old method has remained the default one.

The memory requirements of the new program scale as approximately N^2. This as opposed to the other MP2 programs currently implemented, which scale as at least N^3 for the segmented transformation and as $A * N^2$ for the alternative integral transformation. Here, A and N are the number of active orbitals and the total number of MOs, respectively. Disk requirements of the new code scale as $const1 * A^2N^2$, whereas they scale as $const2 * A^2(N-A)^2$ for the alternative integral transformation. The new code is very light on I/O: only two passes over its disk file are needed, as opposed to the alternative transformation which performs multiple passes over its (huge) disk file. The segmented transformation does not use temporary disk storage. Disk I/O is used, but to a very limited degree. Therefore, the CPU utilization is usually 90% or even better. The CPU utilization is usually less than 50% for other MP2 transformation methods working in the conventional mode, while, in the direct mode, there is a very serious overhead because of the multiple reevaluation of 2-electron integrals.

Asymptotically, the FLOPs count with the new program is about a half or even better as compared to other MP2 energy transformation methods.

Please note that the new MP2 code requires use of P2P when running in parallel. See the section on P2P for more details.

Then, it is important to discuss some specifics about the MP2 gradient code. Since version 7.1, Firefly has new and efficient semidirect MP2 gradient (and properties) program that is based on fastints code and runs in parallel using the P2P interface. The new MP2 gradient program is used by default when running in parallel. Gradients are available only for RHF wavefunctions.

The memory demands of the new gradient program are quite modest: asymptotically, they are proportional to Nocc * (Nocc + Ncore) * Nvir, where Nocc is the number of active occupied orbitals (i.e., excluding frozen core orbitals), Ncore is the number of frozen core occupied orbitals, and Nvir is the number of virtual orbitals. Parallel scalability is good, with most of the memory demands reducing linearly with the number of used computing nodes.

As for disk I/O, the new MP2 gradient program uses (large) temporary files to store half-transformed 2-electron integrals (in the DASORT file) and the non-separable part of the MP2 2-particle density matrix (DM2, stored in DAFL30 file during gradient runs only). Therefore, disk I/O can be quite intensive. When running in parallel, the contents of these files is evenly distributed across all nodes. If one is running this code in parallel on an SMP or multicore system, it is recommended to assign a separate physical disk to each Firefly process.

Unfortunately, the exact amount of disk space required to store files DASORT and DAFL30 cannot be predicted exactly because of two reasons. First, the Firefly uses sparsity of half-transformed integrals and DM2 to save disk space by storing only non-zero values. Second, half-transformed integrals, DM2 elements as well as their labels are further packed, and it

is impossible to predict the exact packing ratio. However, it is possible to provide upper bounds on the overall maximum sizes of DASORT and DAFL30. Note, these bounds usually seriously overestimate the real size of these files!

Namely, the overall size of all DASORT files is less or equal than:

6 * Nocc * Nvirt * Nao * (Nao + 1) bytes

Similarly, the overall size of all DAFL30 files is less or equal than:

8 * Nocc * Nvirt * Nao * (Nao + 1) bytes

Here, Nocc is the number of active occupied orbitals, Nvirt is the number of virtual orbitals, and Nao is the number of Cartesian atomic orbitals.

When using the new gradient program, it is recommended to use the following input:

```
 $CONTRL INTTYP=HONDO $END
 $P2P P2P=.T. DLB=.T. XDLB=.T. $END
 $SMP CSMTX=.T. $END
 $MP2 METHOD=1 $END
```

Depending on your operating system (e.g., under some Windows systems), the use of the following addition options:

  $SYSTEM SPLITF=.T./.F. $END (large file splitting enabled/disabled)

and/or

  $SYSTEM IOFLGS(30)=1 $END (activates file cache write through mode for unit # 30 which contains DM2 elements)

may have serious positive or negative impact on the overall performance as well.

The following set of parameters of the new MP2 gradient code seems to be optimal in the case you are running large problems in parallel on Linux-based SMP or multicore system which do not have a separate physical disks for each instance of the parallel Firefly processes.

```
 $SYSTEM SPLITF=.F. $END
 $SYSTEM IOFLGS(30)=0 $END
 $MP2GRD DBLBF=.F. FUSED=.F. ASYNC=.T. $END
```

The following set of additional I/O parameters seem to be optimal for large-scale jobs running under Windows Vista and Windows Server 2008 R1:

```
 $MP2 IOFLGS(1)=65536,65536 IOFLGS(3)=65536,65536 $END
```

These settings turn on direct unbuffered disk I/O for the files DASORT and DICTNRY and disable standard buffered disk reads and writes.

If the I/O remains a significant bottleneck, or if the amount of available disk space is not sufficient for carrying out an MP2 calculation, one can enable the direct evaluation of AO integrals. This is done by specifying DIRECT=.T. in $MP2, which will somewhat reduce the amount of disk space used at the cost of additional CPU time.

Finally, it should be noted that the new gradient code requires higher accuracy when there is a partial linear dependence in the AO basis set. We recommend the use of the following settings (tighter values can be used if desired):

```
 $CONTRL ICUT=11 INTTYP=HONDO $END
 $SCF NCONV=7 $END
 $MP2GRD TOL1=1D-12 TOL2=1D-12 $END
```

MP3 and MP4 calculations

MP3 and MP4 type calculations can be requested by specifying MPLEVL=3 and MPLEVL=4 in $CONTROL in combination with SCFTYP=RHF. Related keywords can be found in the groups $MP3 and $MP4. As noted earlier, these calculations are only implemented for RHF wavefunctions – ROHF and UHF MP3/MP4 calculations are currently not possible. For MP4, it is possible to perform three types of calculations, namely MP4(SDQ), MP4(SDTQ), and MP4(T). The default MP4 type is MP4(SDQ).

MP3 and MP4(SDQ) calculations are multithreaded and cannot be executed in parallel mode as the corresponding code was written many years ago and was not allowed to run in parallel at the time. Despite this, the code is still quite efficient and scales well with increasing numbers of threads. The MP4(T) code on the other hand was written more recently, is more advanced, and is written to run in parallel, though it also benefits from multithreading. This makes performing an MP4(SDTQ) calculation in the most optimal way a bit complicated.

To explain this a bit more: it is perfectly possible to perform an MP4 calculation in a single job (which can done by specifying $MP4 SDTQ=.T. $END). However, in terms of computational time this is not the most optimal way. When run in parallel, the MP4(SDQ) part of the calculation cannot run in parallel and will therefore just be duplicated on all instances of Firefly. At the same time, multithreaded execution is also not optimal as the MP4(T) part of the calculation will not benefit as much from multithreading as it does from parallel execution. Also, the Hartree-Fock part of the calculation does not benefit from multithreading.

Because of this, it is advised to split an MP4(SDTQ) calculation up in three separate steps (i.e., jobs). The first step is to perform a parallel Hartree-Fock calculation. The option $CONTRL WIDE=.T. $END can hereby be used to punch the HF orbitals with double accuracy. The second step is then to perform a multithreaded MP4(SDQ) calculation, using the orbitals from the previous step as the initial guess. The third and final step is to perform an MP4(T) calculation (which can be requested using $MP4 TONLY=.T. $END), again using the HF orbitals as the initial guess. As noted, the

MP4(T) code can run parallel and multithreaded at the same time and it is advisable to do so. The optimal number of threads per process depends here on the particular computer architecture that is used. In most cases, it is equal to the number of physical cores sharing the same memory domain on the cc-NUMA system. E.g., assuming that a 2-way four-core Xeon 5500 system is used, the best way is to use 2 processes per box with four working threads each. However, this can be adjusted to minimize I/O or scratch storage, and the code is flexible here.

The usage of Abelian symmetry can greatly decrease the required CPU time for any MP3 or MP4 calculations. Unlike MP2 case, where the speed up due to the use of symmetry is roughly proportional to the first power of the order of the symmetry group (Ng) used, in the case of MP3 and MP4 jobs, the speed up is proportional to Ng^2 on the average.

It finally should be noted that, in practice, there might not be much of a reason for performing MP3 calculations, as the computational cost of a large MP4(SDQ) calculation is typically only 2-3 times greater than that of a similar MP3 job. Also, the memory demands of both types of calculations are very similar. MP4(SDTQ) jobs are always much more demanding.

Density Functional Theory

General information

Density functional theory in Firefly is implemented for restricted, unrestricted, and restricted-open wavefunctions. For each wavefunction, energies and analytical gradients are available. Analytical Hessians are currently not available. Instead, Hessians can be calculated in a semi-numerical fashion. Double hybrid functionals are an exception – for these functionals, only energies are currently available. Various functionals can be used, a list of these is provided in the next section. It should be noted that the DFT code in Firefly is completely different from that of GAMESS (US) with respect to both implementation and input specification.

A DFT calculation can be requested by specifying a functional using the DFTTYP keyword in $CONTRL. The value of SCFTYP determines which type of wavefunction is used. For example, a RO-BLYP calculation can be requested using

```
SCFTYP=ROHF DFTTYP=BLYP
```

Additional DFT-related input is provided with the optional $DFT group. The majority of the keywords in this group can be used to change the accuracy of various aspects of the DFT calculation. The default values are fine for most cases, so providing this input usually isn’t necessary.

Information on the convergence accelerators and conventional vs direct runs can be found in the chapter on Hartree-Fock.

A few things should be mentioned about performance. First of all, the current DFT implementation in Firefly does not support multithreading. It is therefore recommended to run calculations in parallel mode. Secondly, the molecular symmetry specified in $DATA is used only partially during calculation of the DFT contributions to the Fock matrix. To be more precise, the so-called octant symmetry is not used at present.

Resolution of identity / Coulomb fitting techniques, which serve to speed up calculations that use pure DFT functionals, are currently not implemented. As a result, one might expect that other DFT programs that exploit these techniques (for example, TURBOMOLE) will outperform Firefly on pure DFT calculations. This is especially true with large molecules. For such molecules, the time for the pure DFT part of calculations depends approximately linearly on the number of atoms in molecule. The reason for this is that, while for small molecular systems the pure DFT part of the calculations is usually the time-limiting step, for large systems the cost of the standard Coulomb (and possible exchange for hybrid functionals) contributions to the Fock matrix due to two-electron integrals becomes the dominant part of the calculations. The implementation of Coulomb fitting techniques should overcome this shortcoming.

It should also be noted that the RODFT energy/gradient code is routed through the generic UDFT code, therefore, the RODFT performance is identical to that of UDFT. This situation might be changed in the future by adding separate RODFT routines which should provide a speedup of approximately 15 to 20 % for small and medium size molecules.

Double hybrid functionals, which use an MP2-like perturbation term in the correlation part of the functional, can make use of any of the MP2 methods in Firefly. Just as with MP2 calculations, $MP2 METHOD=1 is the preferred method except for small jobs. Please see the chapter on MP2 for more information.

An example input file for a BLYP geometry optimization and subsequent vibrational analysis for a water molecule:

```
 $CONTRL SCFTYP=RHF DFTTYP=BLYP RUNTYP=OPTIMIZE $END
 $SYSTEM TIMLIM=3000 MEMORY=3000000 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $STATPT HSSEND=.T. NPRT=-2 $END
 $FORCE NVIB=2 VIBSIZ=0.005 $END
 $DATA
H2O
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 H           1.0   0.0000000000   0.7565140024   0.1397092302
 $END
```

Finally, some technical details should be mentioned. The DFT implementation in Firefly is grid-based. Angular integration is based on Lebedev quadratures, while radial integration is based on Mura-Knowles quadratures. The atomic partitioning function used is the modification of Stratmann, Scuseria and Frisch (Chem. Phys. Lett. (1996) 257, 213).

Available functionals

Before providing the full list of functionals available in Firefly, it is important to mention that the specification of a functional in the input is unambiguous except for a few specific cases that will discussed below.

Firstly, there are two variations of the VWN functional available as components of the B3LYP and O3LYP functionals, namely VWN formula 1 RPA and VWN formula 5. The reason for the inclusion of both variants is to improve compatibility with other QC programs, which can use either of the two formulae. Formula 1 RPA type B3LYP is used by for example NWCHEM and GAUSSIAN, while formula 5 type B3LYP is the default with GAMESS (US). The use of VWN formula 1 RPA in B3LYP and O3LYP can be specified with DFTTYP=B3LYP1 and DFTTYP=O3LYP1, respectively. Similarly, the use of VWN formula 5 can be specified with DFTTYP=B3LYP5 and DFTTYP=O3LYP5. Note that some QC programs refer to VNW formula 1 RPA as “VWN functional III” and that these are the same functionals.

For B3LYP, it is also possible to specify DFTTYP=B3LYP. The choice of VWN formula will then depend on the value of B3LYP in the $DFT group. Possible values are NWCHEM (formula 1 RPA, the default) and GAMESS (formula 5). DFTTYP=O3LYP will always be interpreted as O3LYP5.

For the O3LYP functionals, the weight of the non-local exchange can be adjusted with the O3LYP keyword in $DFT. The reason for including this option is that recent versions of Gaussian use a weight different from the one used by most other QC programs. Setting O3LYP to GAUSSIAN changes the weight so that the resulting functional will be identical to one used in Gaussian 03 Rev D.01 and above. Specifying O3LYP=DEFAULT (which is, not entirely surprising, the default value) enables compatibility with the implementation used in other QC programs. Note that the reason for the existence of two different implementations is that there are some ambiguities in O3LYP-related papers. Due to these ambiguities, is not possible to say which implementation is “correct”. Instead, these two implementations should be considered as two different functionals.

There is also a keyword pertaining to the Perdew-Zunger 1981 LDA correlation that should be mentioned. In the paper defining this functional, its parameters were given with only four digits. As there are two branches in the fit, two parameters of the fit were selected in such a manner that Exc and Vxc are globally continuous functions, even at the branch point itself. However, the four-digit precision is not enough for this purpose as it causes small deviations from exact continuity that in turn result in errors such as non-precise gradients. Tight geometry optimizations in particular are not possible with this functional. By default, the parameters of the fit are used with increased precision (for P81LDA as well as for all functionals using P81 local correlation), hereby achieving smooth Exc and Vxc. However, this corrected fit does result in a slightly different functional. If desired, the functional can be reverted to its slightly discontinuous form by specifying FIXP81=.F. in the $DFT group.

With respect to double hybrid functionals, it should be mentioned that most of these functionals by default do not freeze any cores. This behavior is forced by Firefly, but can be overridden by specifying a negative value for NCORE. This keyword belongs to the $MP2 group (not the $DFT group), and input should look like this:

```
 $MP2 NCORE=-5 $END
```

Here, the minus sign tells Firefly to override the default behavior and the number is the amount of cores to be frozen.

Note that there are two functionals which by default *do* use frozen cores, namely D-LYP and D-PBEP86 (as these functionals were designed to be used together with the frozen core approximation). This behavior is not forced though so, if desired, it can be changed using the NCORE keyword without having to use the minus sign.

The list of available functionals is as follows:

Exchange functionals (no correlation)

DFTTYP value	Functional	Refs.
LDA exchange		
SLATER	Slater exchange	1
LSDA	A synonym of SLATER	
GGA exchange		
B88	Becke 1988 exchange	2
XPW91	Perdew-Wang 1991 exchange	3
XMPW91	Modified Perdew-Wang 1991 exchange. Modification by Adamo and Barone	11
GILL96	Gill 1996 exchange	4
XPBE96	Perdew-Burke-Ernzerhof 1996 exchange	5
OPTX	Handy-Cohen 2001 OPTX exchange	6
XSOGGA	The exchange part of the SOGGA functional by Zhao and Truhlar	20
XSOGGA11	The exchange part of the SOGGA-11 functional by Peverati, Zhao, and Truhlar	21
Hybrid exchange		
XSOGGAX	The exchange part of the SOGGA-11X hybrid functional by Peverati and Truhlar	31

Pure correlation functionals – these use 100 % exact (i.e., Hartree-Fock) exchange)

DFTTYP value	Functional	Refs.
LDA correlation		
VWN1RPA	VWN formula 1 RPA local correlation. This functional is referred to as “VWN formula 3” by some programs (such as Gaussian, Q-Chem).	7
VWN5	VWN formula 5 local correlation	8
P81LDA	Perdew-Zunger 1981 local correlation	36
PW91LDA	Perdew-Wang 1991 local correlation	9
GGA correlation		
P86	Perdew 1986 nonlocal (1.0) + Perdew-Zunger 1981 local (1.0) correlation	37,36
LYP	Lee-Yang-Parr 1988 correlation	10
CPBE96	Perdew-Burke-Ernzerhof 1996 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	5,9
CPW91	Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	12,9
CSOGGA11	The correlation part of the SOGGA-11 functional by Peverati, Zhao, and Truhlar	21

CSOGGAX	The correlation part of the SOGGA-11X functional by Peverati and Truhlar	31

Exchange-correlation functionals

DFTTYP value	Functional	Refs.
SVWN1RPA	Slater exchange, VWN formula 1 RPA local correlation	1,7
BVWN1RPA	Becke 1988 exchange, VWN formula 1 RPA local correlation	2,7
SVWN5	Slater exchange, VWN formula 5 local correlation	1,8
BVWN5	Becke 1988 exchange, VWN formula 5 local correlation	2,8
SP81LDA	Slater exchange, Perdew-Zunger 1981 local correlation	1,36
BP81LDA	Becke 1988 exchange, Perdew-Zunger 1981 local correlation	2,36
BP86	Becke 1988 exchange, Perdew 1986 nonlocal (1.0) + Perdew-Zunger 1981 local (1.0) correlation	2,37,36
PBEP86	Perdew-Burke-Ernzerhof 1996 exchange, Perdew 1986 nonlocal (1.0) + Perdew-Zunger 1981 local (1.0) correlation	5,37,36
SLYP	Slater exchange, Lee-Yang-Parr 1988 correlation	1,10
BLYP	Becke 1988 exchange, Lee-Yang-Parr 1988 correlation	2,10
GLYP	Gill 1996 exchange, Lee-Yang-Parr 1988 correlation	4,10
MPW91LYP	Modified Perdew-Wang 91 exchange, Lee-Yang-Parr 1988 correlation	11,10
OLYP	OPTX exchange, Lee-Yang-Parr 1988 correlation	6,10
XLYP	Extended exchange functional (a combination of Slater local (1.0), Becke 88 nonlocal (0.722), and Perdew-Wang 91 nonlocal (0.347) exchange), Lee-Yang-Parr 1988 correlation	13
BPW91	Becke 1988 exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	2,12,9
PW91	Perdew-Wang 1991 exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	3,12,9
MPW91	Modified Perdew-Wang 91 exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	11,12,9
PBEPW91	Perdew-Burke-Ernzerhof 1996 exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	5,12,9
PBE96	Perdew-Burke-Ernzerhof 1996 exchange, Perdew-Burke-Ernzerhof 1996 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	5,9
MPW91PBE	Modified Perdew-Wang 91 exchange, Perdew-Burke-Ernzerhof 1996 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	11,5,9

HCTH93	HCTH/93 exchange-correlation functional	14
HCTH120	HCTH/120 exchange-correlation functional	15
HCTH147	HCTH/147 exchange-correlation functional	15
HCTH407	HCTH/407 exchange-correlation functional	16
HCTH407P	HCTH/407+ exchange-correlation functional	17
HCTHP14	HCTH potential-fitted 1/4 functional	18
B97GGA1	Becke 97, reparametrized as pure GGA by Cohen and Handy (2000)	19
B97D	Becke 97-D, another pure GGA reparameterization of B97, by Grimme (2006). To be used with DFT-D. Selection of this functional automatically enables the use of the DFT-D2 scheme (unless the user chooses a different scheme or explicitly disables the use of DFT-D)	30
SOGGA	SOGGA exchange-correlation functional by Zhao and Truhlar	20
SOGGA11	SOGGA-11 exchange-correlation functional by Peverati, Zhao, and Truhlar	21

Hybrid functionals

DFTTYP value	Functional	Refs.
B1P86	Becke 1988 (0.75) + Hartree-Fock (0.25) exchange, Perdew 1986 nonlocal (1.0) + Perdew-Zunger 1981 local (1.0) correlation	2,37,36
B1LYP	Becke 1988 (0.75) + Hartree-Fock (0.25) exchange, Lee-Yang-Parr 1988 correlation	2,10
BHHLYP	Becke 1988 (0.5) + Hartree-Fock (0.5) exchange, Lee-Yang-Parr 1988 correlation	24,10
B1PW91	Becke 1988 (0.75) + Hartree-Fock (0.25) exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	2,12,9
B3P86	Becke 1988 nonlocal (0.72) + Slater local (0.80) + Hartree-Fock (0.20) exchange, Perdew 1986 nonlocal (1.0) + Perdew-Zunger 1981 LDA local (1.0) correlation	22,37,36
B3LYP1	Becke 1988 nonlocal (0.72) + Slater local (0.80) + Hartree-Fock (0.20) exchange, Lee-Yang-Parr 1988 (0.81) + VWN formula 1 RPA local (0.19) correlation. This is B3LYP as implemented in NWCHEM and GAUSSIAN	22,10
B3LYP5	Same as B3LYP1, but with VWN formula 5 instead of VWN formula 1 RPA local correlation. This is B3LYP as implemented in GAMESS (US)	
B3LYP	Either B3LYP1 or B3LYP5, depending on the value of the B3LYP keyword in the $DFT group (default is B3LYP1)	
B3PW91	Becke 1988 nonlocal (0.72) + Slater local (0.80) +	22,12,9

	Hartree-Fock (0.20) exchange, Perdew 1991 nonlocal (0.81) + Perdew-Wang 1991 LDA local (1.0) correlation	
O3LYP1	OPTX nonlocal (0.8133) + Slater local (0.9262) + Hartree-Fock (0.1161) exchange, Lee-Yang-Parr 1988 (0.81) + VWN formula 1 RPA local (0.19) correlation	23
O3LYP5	Same as O3LYP1, but with VWN formula 5 instead of VWN formula 1 RPA local correlation	
O3LYP	A synonym of O3LYP5	
X3LYP	Extended hybrid exchange functional (a combination of Slater local (0.782), Becke 88 nonlocal (0.542), and Perdew-Wang 91 nonlocal (0.167) exchange + Hartree-Fock (0.218)), Lee-Yang-Parr (0.871) + VWN formula 1 RPA local (0.129) correlation	13
PBE1P86	Perdew-Burke-Ernzerhof 1996 (0.75) + Hartree-Fock (0.25) exchange, Perdew 1986 nonlocal (1.0) + Perdew-Zunger 1981 local (1.0) correlation	5,37,36
PBE1PW91	Perdew-Burke-Ernzerhof 1996 (0.75) + Hartree-Fock (0.25) exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 LDA local (1.0) correlation	25,12,9
PBE0	Perdew-Burke-Ernzerhof 1996 (0.75) + Hartree-Fock (0.25) exchange, Perdew-Burke-Ernzerhof 1996 nonlocal (1.0) + Perdew-Wang 1991 LDA local (1.0) correlation. This functional is also known as PBE1PBE	25
MPW1LYP	Modified Perdew-Wang 91 (0.75) + Hartree-Fock (0.25) exchange, Lee-Yang-Parr 1988 correlation	11,10
MPW1PW91	Modified Perdew-Wang 91 (0.75) + Hartree-Fock (0.25) exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 LDA local (1.0) correlation	11
MPW1K	Modified Perdew-Wang 91 (0.572) + Hartree-Fock (0.428) exchange, Perdew 1991 nonlocal (1.0) + Perdew-Wang 1991 LDA local (1.0) correlation. Functional by Truhlar and coworkers.	41
MPW1PBE	Modified Perdew-Wang 91 (0.75) + Hartree-Fock (0.25) exchange, Perdew-Burke-Ernzerhof 1996 nonlocal (1.0) + Perdew-Wang 1991 local (1.0) correlation	11,5,9
MPW3PBE	Modified Perdew-Wang 91 local and nonlocal (0.72) + Slater local (0.08) + Hartree-Fock (0.20) exchange Perdew-Burke-Ernzerhof 1996 nonlocal (0.81) + Perdew-Wang 1991 local (1.0) correlation	11,1,5,9
B970	Becke 97 hybrid exchange-correlation functional	26
B980	Becke 98 hybrid exchange-correlation functional	27
B971	Becke 97-1, a 1998 reparametrized version of B97 by Handy, Tozer, and coworkers	14
B972	Becke 97-2, a 2001 reparametrized version of B97 by Wilson, Bradley, and Tozer	28
B973	Becke 97-3, a 2005 reparametrized version of B97 by Keal and Tozer	29

SOGGAX	SOGGA-11X hybrid exchange-correlation functional by Peverati and Truhlar	31

Double hybrid functionals

DFTTYP value	Functional	Refs.
Self-consistent functionals		
B2PLYP	Becke 1988 (0.47) + Hartree-Fock (0.53) exchange, Lee-Yang-Parr 1988 (0.73) + MP2 (0.27) correlation	32
B2GPPLYP	Becke 1988 (0.35) + Hartree-Fock (0.65) exchange, Lee-Yang-Parr 1988 (0.64) + MP2 (0.36) correlation	33
MPW2PLYP	Modified Perdew-Wang 91 (0.45) + Hartree-Fock (0.55) exchange, Lee-Yang-Parr 1988 (0.75) + MP2 (0.25) correlation	42
DSD-BLYP	Initial version of DSD-BLYP(full), which uses the following terms and parameters: Becke 1988 (0.31) + Hartree-Fock (0.69) exchange, Lee-Yang-Parr 1988 (0.54) + opposite spin MP2 (0.46) + same spin MP2 (0.37) correlation + DFT-D version 2 with an alpha value of 60.0. Selection of this functional automatically sets DFTD=.T.	38
D-BLYP	Newer version DSD-BLYP(fc), which uses the following terms and parameters: Becke 1988 (0.29) + Hartree-Fock (0.71) exchange, Lee-Yang-Parr 1988 (0.55) + opposite spin MP2 (0.46) + same spin MP2 (0.43) correlation + DFT-D version 2 with an alpha value of 20.0. Selection of this functional automatically sets DFTD=.T.	39
D-PBEP86	DSD-PBEP86(fc) functional: Perdew-Burke-Ernzerhof 1996 (0.32) + Hartree-Fock (0.68) exchange, Perdew 1986 nonlocal (0.45) + Perdew-Zunger 1981 LDA local (0.45) + opposite spin MP2 (0.51) + same spin MP2 (0.23) correlation + DFT-D version 2. Selection of this functional automatically sets DFTD=.T. Functional and empirical dispersion parameters recommended for the use of D-PBEP86 with DFT-D version 4 are not implemented but can be given manually.	39
Non-self-consistent functionals		
XYG3	Becke 1988 nonlocal (0.2107) + Slater local (0.1967) + Hartree-Fock (0.8033) exchange, Lee-Yang-Parr 1988 (0.6789) + MP2 (0.3211) correlation. Each term in this functional (incl. the MP2-like term) is evaluated using B3LYP1 orbitals and density.	34
XYGJ-OS	Slater local (0.2269) + Hartree-Fock (0.7731) exchange, Lee-Yang-Parr 1988 (0.2754) + VWN formula 1 RPA local (0.2309) + MP2 (0.4364) correlation. Each term in this functional (incl. the MP2-like term) is	40

	evaluated using B3LYP1 orbitals and density. From the MP2-like term, only the opposite spin part is used.	
XDH-PBE0	Perdew-Burke-Ernzerhof 1996 (0.1665) + Hartree-Fock (0.8335) exchange, Perdew-Burke-Ernzerhof 1996 nonlocal (0.5292) + Perdew-Wang 1991 LDA local (0.5292) + MP2 (0.5428) correlation. Each term in this functional (incl. the MP2-like term) is evaluated using PBE0 orbitals and density. From the MP2-like term, only the opposite spin part is used.	35

References

1. J.C. Slater; Phys. Rev., 81, 385-390 (1951)
2. A.D. Becke; Phys. Rev. A, 38, 3098-3100 (1988)
3. J.P. Perdew, J.A. Chevary, S.H. Vosko, K.A. Jackson, M.R. Pederson, D.J. Singh, C.Fiolhais; Phys. Rev. B, 46, 6671-6687 (1992)
4. P.M.W. Gill; Mol. Phys., 89, 433-445 (1996)
5. J.P. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett., 77, 3865-3868 (1996)
6. N.C. Handy, A.J. Cohen; Mol. Phys., 99, 403-412 (2001)
7. Functional III in S.H. Vosko, L. Wilk, M. Nusair; Can. J. Phys., 58, 1200-1211 (1980)
8. Functional V in S.H. Vosko, L. Wilk, M. Nusair; Can. J. Phys., 58, 1200-1211 (1980)
9. J.P. Perdew, Y. Wang; Phys. Rev. B, 45, 13244-13249 (1992)
10. C. Lee, W. Yang, R.G. Parr; Phys. Rev. B, 37, 785-789 (1988)
11. C. Adamo, V. Barone, J. Chem. Phys., 108, 664-675 (1998)
12. J.P. Perdew, J.A. Chevary, S.H. Vosko, K.A. Jackson, M.R. Pederson, D.J. Singh, C.Fiolhais; Phys. Rev. B, 46, 6671-6687 (1992)
13. X. Xu, W. A. Goddard III, P. Natl. Acad. Sci. USA, 101, 2673-2677 (2004)
14. F.A. Hamprecht, A.J. Cohen, D.J. Tozer, N.C. Handy J. Chem. Phys., 109, 6264-6271 (1998)
15. A.D. Boese, N.L. Doltsinis, N.C. Handy, M. Sprik. J. Chem. Phys., 112, 1670-1678 (2000)
16. A.D. Boese, N.C. Handy, J. Chem. Phys., 114, 5497-5503 (2001)
17. A.D. Boese, A. Chandra, J.M.L. Martin, D. Marx, J. Chem. Phys., 119, 5965-5980 (2003)
18. G. Menconi, P.J. Wilson, D.J. Tozer, J. Chem. Phys., 114, 3958-3967 (2001)
19. A.J. Cohen and N.C. Handy, Chem. Phys. Lett., 316, 160-166 (2000)
20. Y. Zhao, D.G. Truhlar, J. Chem. Phys., 128, 184109 (2008)
21. R. Peverati, Y. Zhao, D.G. Truhlar, J. Phys. Chem. Lett., 2, 1991–1997 (2011)
22. A.D. Becke, J. Chem. Phys., 98, 5648-5652 (1993)
23. W.M. Hoe, A.J. Cohen, N.C. Handy, Chem. Phys. Lett., 341, 319-328 (2001)
24. A.D. Becke J. Chem. Phys., 98, 1372-1377 (1993)
25. C. Adamo, V. Barone, J. Chem. Phys., 110, 6158-6169 (1999)
26. A.D. Becke, J. Chem. Phys., 107, 8554-8560 (1997)
27. H.L. Schmider, A.D. Becke, J. Chem. Phys., 108, 9624-9631 (1998)
28. P.J. Wilson, T.J. Bradley, D.J. Tozer, J. Chem. Phys., 115, 9233-9242 (2001)
29. T.W. Keal, D.J. Tozer, J. Chem. Phys., 123, 121103 (2005)
30. S. Grimme, J. Comp. Chem., 27, 1787-1799 (2006)
31. R. Peverati, D.G. Truhlar, J. Chem. Phys., 135, 191102 (2011)
32. S. Grimme, J. Chem. Phys., 124, 034108 (2006)
33. A. Karton, A. Tarnopolsky, J.-F. Lamère, G.C. Schatz, J.M.L. Martin, J. Phys. Chem. A, 112, 12868–12886 (2008)
34. Y. Zhang. X. Xu, W. A. Goddard III, P. Natl. Acad. Sci. USA, 106, 4963-4968 (2009)

35. I.Y. Zhang, N.Q. Su, E.A.G. Brémond, C. Adamo, X. Xu, J. Chem. Phys. 136, 174103 (2012)
36. J.P. Perdew, A. Zunger, Phys. Rev. B, 23, 5048-5079 (1981)
37. J.P. Perdew, Phys. Rev. B, 33, 8822-8824 (1986)
38. S. Kozuch, D. Gruzman, J.M.L. Martin, J. Phys. Chem. C, 114, 20801-20808 (2010)
39. S. Kozuch, J.M.L. Martin, Phys. Chem. Chem. Phys., 13, 20104-20107 (2011)
40. I.Y. Zhang, X. Xu, Y. Jung, W.A. Goddard III, Proc. Natl. Acad. Sci. U.S.A., 108, 19896-19900 (2011)
41. B.J. Lynch, P.L. Fast, M. Harris and D.G. Truhlar, J. Phys. Chem. A, 104, 4811-4815 (2000)
42. T. Schwabe, S. Grimme, Phys. Chem. Chem. Phys., 8, 4398-4401 (2006)

Making modifications to functionals

Firefly has four important keywords, all belonging to the $DFT group, which can be used to modify some of the (double) hybrid functionals implemented. These are:

- HFX, which provides control over the amount of exact exchange;
- CPT2, which provides control over the amount of MP2-like correlation;
- SCS, which controls scaling of spin components of the MP2-like perturbation;
- PARENT, which selects the parent functional used in non-self-consistent double hybrid functionals (such as XYG3, for which the parent functional is B3LYP1).

Changing the amount of exact exchange used in a functional can, for example, be useful in TDDFT calculations to distinguish between Rydberg and valence states. The amount of exact exchange can be controlled through the HFX option. Formally, HFX should be given as an array (for reasons that will be discussed later), e.g.

```
 HFX(1)=0.75
```

However, it is possible to specify HFX as a single variable:

```
 HFX=0.75
```

The HFX option is available for most hybrid functional and all double hybrid functionals. However, for the B97/B98 type functionals as well as for the SOGGA-X functional this option has not been implemented. These functionals were parameterized for a certain amount of exact exchange, so changing this amount doesn’t make much sense. Note that there is no limit to HFX – it is possible to set its value larger than 1.0 for experimentation purposes.

For (double) hybrid functionals that combine exact exchange with only one type of DFT exchange, such as BHHLYP, PBE0, and MPW1PW91, the way the HFX keyword works is quite straightforward. Setting HFX(1)=0 for example removes all exact exchange from the functional, while HFX(1)=1 makes the exchange part of the functional fully exact. For example, setting HFX(1)=0

for PBE1PW91 makes the functional identical to PBEPW91, while HFX(1)=1 makes the functional identical to PW91. Naturally, any value between 0 and 1 is possible. HFX(1)=0.25 will use the default amount of exact exchange for the PBE1PW91.

For the B3LYP, O3LYP, MPW3PBE, and XYG3 functionals, the HFX interpolation is a bit more complex and is best explained by an example. The exchange part of B3LYP consists of the following parameters (default values in parentheses):

```
 B88X – the amount of Becke 88 local and nonlocal (0.72)
 LDAX – the amount of excess Slater local (0.08)
        (because Becke 88 non-local + Slater local = Becke 88 GGA)
 HFX – the amount of exact exchange (0.2)
```

When the value of HFX is changed, B88X and LDAX change as follows:

```
 B88X_new = B88X * (1 - HFX) / (B88X + LDAX)
 LDAX_new = LDAX * (1 - HFX) / (B88X + LDAX)
```

For X3LYP, the interpolation scheme is different as this functional combines 4 types of exchange. Its parameters in the exchange part are as follows:

```
 B88X – the amount of Becke 88 local and nonlocal (0.542)
 PW91X – the amount of Perdew-Wang 91 local and nonlocal (0.167)
 LDAX – the amount of excess Slater local (0.073)
 HFX – the amount of exact exchange (0.218)
```

When the value of HFX is changed, the other parameters change as follows:

```
 B88X_new = B88X * (1 - HFX) / (1 – HFX_default)
 PW91X_new = PW91X * (1 - HFX) / (1 – HFX_default)
 LDAX_new = LDAX * (1 - HFX) / (1 – HFX_default)
```

where HFX_default is the default value of 0.218.

For non-self-consistent double hybrid functionals, HFX can be used to control the amount of exact exchange for the double hybrid functional as well as its parent function. This can be done by giving an array of two variables. As an example, for xDH-PBE0, the default values can be specified as:

```
 HFX(1)=0.8335,0.25
```

where the first value specifies the amount of exact exchange in the double hybrid functional and the second value specified the amount of exact exchange in the parent functional.

The CPT2 keyword works in the same way as the HFX keyword. As noted earlier, it can be used to control the amount of MP2-like correlation in double hybrid functionals.

Finally, the SCS keyword can be used to scale the spin components of the MP2-like part by certain factors. It functions the same way as the SCS keyword in the $MP2 group. For more information, please see the MP2 section.

Of course, the HFX, CPT2, and SCS can be used to define a few functionals currently not present in Firefly, provided that these are of the same form as a functional already implemented. For example, the B2K-PLYP functional can be specified as follows:

```
 $CONTRL DFTTYP=B2PLYP $END
 $DFT HFX(1)=0.72 CPT2=0.42 $END
```

Here, DFTTYP is set to B2PLYP as this functional is of the same form as B2K-PLYP – only the fractions of exact exchange and MP2-like correlation are different. It would also have been possible to specify DFTTYP=B2GPPLYP as this functional is of the same form.

The DSD-PBEP86 functional implemented in Firefly was designed to be used with DFT-D version 2. However, reference 39 also contains parameters optimized for use with version 4. As a second example, this functional can be specified as follows:

```
 $CONTRL DFTTYP=D-PBEP86 DFTD=.T. $END
 $DFT HFX(1)=0.70 CPT2=0.57
 SCS(1)=0.9298245614035088,0.4385964912280702 $END
 $DFTD VERSN=4 S6=0.418 RS6=0.0 S18=0.25 RS18=5.65 ALP=14.0 $END
```

More information on DFT-D can be found in the next section.

Lastly, the PARENT keyword should be discussed. This keyword is only used for non-self-consistent double hybrid functionals such as XYG3 and selects the parent functional. This parent functional is used for generating the orbitals and density used for the evaluation of each term in the double hybrid functional. Any GGA exchange, GGA correlation, exchange-correlation, and hybrid functional may be specified. LDA functionals and double hybrid functionals cannot be used.

Empirical dispersion correction (DFT-D)

Newly available in Firefly 8.0.0 is the empirical dispersion correction to DFT proposed by Grimme and co-workers, a method commonly abbreviated as DFT-D. Three versions of the correction are available:

- DFT-D version 2, which was proposed in 2006 as part of the B97-D functional and which is an update of the original DFT-D version.
- DFT-D version 3, proposed in 2010.

- DFT-D version 3 with Becke-Johnson damping (from here on referred to as ‘version 4’), proposed in 2011.

The corresponding references are:
- Version 2: S. Grimme, J. Comput. Chem., 27 (2006), 1787-1799
- Version 3: S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, J. Chem. Phys. 132 (2010), 154104
- The BJ-damping used in version 4: S. Grimme, S. Ehrlich, and L. Goerigk, J. Comput. Chem. 32 (2011), 1456-1465

It is important to point out that the available range of versions (together with their default parameters) is defined by the DFT-D extension file in use (either dftd.dll or dftd.ex). Consequently, the empirical dispersion correction functionality is only available if Firefly is capable to find and load this extension file.

The empirical dispersion correction is designed to improve the long-range behavior of DFT methods. However, its use is not limited to DFT only – it can actually be used with any computational method present in Firefly. Both energies and analytical gradients are available with DFT-D, provided of course that analytical gradients are available for the method the correction is used with. Analytical second derivatives are not available, instead, second derivatives should be obtained numerically.

The use of the correction can be enabled by specifying DFTD=.T. in the $CONTRL group. Control over the correction is provided by keywords in the $DFTD group. Here, the VERSN keyword can be used to specify the DFT-D version to be used (default is VERSN=3). Thus, if one would like to use DFT-D version 4, one should specify:

```
$CONTRL DFTD=.T. $END
$DFTD VERSN=4 $END
```

Parameters for many functionals (as well as for Hartree-Fock) are stored internally in Firefly. Their availability is as follows:

Version 2	Version 3	Version 3 with TZ=.T. (*)	Version 4
BP86	Hartree-Fock	BP86	Hartree-Fock
BLYP	Slater (LSDA)	BLYP	BP86
B3LYP1	BP86	B3LYP1	BLYP
B3LYP5	BLYP	B3LYP5	B3LYP1
PBE96	B3LYP1	PBE96	B3LYP5
PBE0	B3LYP5	PBE0	BHHLYP
B97-D	BHHLYP	MPW91LYP	B3PW91
B2PLYP	B3PW91	B97-D	PBE96
B2GP-PLYP	PBE96	B2PLYP	PBE0
DSD-BLYP	PBE0		MPW91LYP
D-BLYP	HCTH/120		HCTH/120
D-PBEP86	OLYP		OLYP
	B97-D		B97-D
	B2PLYP		B2PLYP
	B2GP-PLYP		B2GP-PLYP

(*) The TZ keyword selects the use of a special set of DFT-D version 3 parameters optimized for Ahlrichs' TZVPP basis set. Its default value is .FALSE.

The parameters of the correction model can be changed with the S6, RS6, S18, RS18, and ALP keywords. These should be used if one uses a functional/method for which parameters are not stored internally, or if one would like to use custom parameters. Their functions are:

S6 The s_6 global scaling factor, the main scaling factor in DFT-D version 2. For DFT-D version 3 and 4 it is of lesser importance and is usually set to 1.0 (except with double hybrid functionals).

RS6 For DFT-D version 2, this parameter is used in calculating the dampening factor – its value is 1.1 (for all functionals). For DFT-D version 3, this parameter corresponds to the $s_{r,6}$ scaling factor, which is the main scaling factor in this version. For DFT-D version 4, it corresponds to the α_1 free fit parameter.

S18 The s_8 scaling factor used in DFT-D version 3 and 4. For DFT-D version 2, this parameter has no function.

RS18 For DFT-D version 3, this parameter is used in calculating the dampening factor, its value being 1.0 except with Slater exchange (where its value is 0.697). For DFT-D version 4, this parameter corresponds to the α_2 free fit parameter. For DFT-D version 2, this parameter has no function.

ALP The global scaling parameter of the damping function (which dampens the dispersion correction at short ranges). Its value is usually 20 for DFT-D version 2, and 14 for DFT-D version 3 and 4. For DFT-D version 4, this parameter is only used when ABC=.T. (see below).

The above parameter naming scheme is identical to the one used in the DFT-D3 program by Grimme. However, a few of these parameters have names which do not correspond to the parameters they represent, or have different meanings depending on the DFT-D version used. As we assume this can be confusing for some, aliases have been created for these parameters. Available aliases are:

Keyword	Aliases
RS6	SR6, ALPHA1
S18	S8
RS18	ALPHA2

It is important to note that, when specifying a custom set of parameters, all five of the above parameters should be given a value, even those which are zero or not used by the DFT-D version used!

In addition to the above parameters, there are a few other parameters and keywords that should be mentioned. First of all, it is possible to control the coordination number dependent dispersion (used in DFT-D version 3 and 4) through the K1, K2, and K3 keywords, which correspond to the k_1, k_2, and k_3 parameter, respectively. Their default values are K1=16, K2=4/3, and K3=-4. Typically, there is no need to change them.

Furthermore, it is possible to enable the three-body non-additive contribution to the dispersion correction. This is done by specifying ABC=.T. and pertains to DFT-D versions 3 and 4. By default, the three-body contribution is disabled as Grimme and co-workers found that inclusion of three-body terms had only a marginal impact overall while leading to poorer results in a few specific cases.

It should finally be noted that some parameters, such as C_6 and R_0, cannot be controlled. The values of these parameters are used as specified in the references mentioned at the start of this section.

Time-Dependent Theories

General information

Time-dependent Hartree-Fock (TDHF; also known as RPA, random phase approximation) and time-dependent density functional theory (TDDFT) are theories used to investigate properties of a system in the presence of an external field such an electric or magnetic field. In Firefly, these theories can be used to calculate the properties of excited state such as energies and oscillator strengths. Furthermore, it is possible to calculate various static and/or frequency dependent polarizabilities (with an emphasis on important NLO properties such as second and third harmonic generation). Both types of calculations can be performed by direct type calculations using the 'fastints' computation module (see the "Performance" chapter for details). Only an RHF reference is allowed at present. Analytical gradients are not yet programmed for both TDHF and TDDFT theories.

It is important to emphasize that time-dependent excited state calculations and time-dependent polarizability calculations take place through two different modules and therefore need very different input. Excited state calculations are requested through the CITYP keyword in $CONTRL group. The $TDHF or $TDDFT group can hereby be used to specify additional keywords. Two examples:

```
$CONTRL SCFTYP=RHF CITYP=TDHF RUNTYP=ENERGY $END
$TDHF <additional keywords> $END

$CONTRL SCFTYP=RHF CITYP=TDDFT DFTTYP=PBE0 RUNTYP=ENERGY $END
$TDDFT <additional keywords> $END
```

Polarizability calculations on the other hand are requested by RUNTYP=TDHF in $CONTRL and do not use the CITYP keyword. Two examples:

```
$CONTRL SCFTYP=RHF RUNTYP=TDHF NOSYM=1 $END
$TDHF <additional keywords> $END

$CONTRL SCFTYP=RHF DFTTYP=PBE0 RUNTYP=TDHF NOSYM=1 $END
$TDHF <additional keywords> $END
```

As can be seen, both runs use RUNTYP=TDHF, even though the second one is a TDDFT calculation. Furthermore, in both cases any additional keywords have to be specified through the $TDHF group; the $TDDFT group is not used. The specification of NOSYM=1 is required because the Fock matrices computed during the time-dependent Hartree-Fock CPHF are not symmetric.

It is currently not possible to use all of the available DFT functionals for TDDFT calculations. Supported values of DFTTYP are:

Exchange	Correlation	Exchange-correlation	Hybrid
SLATER	VWN1RPA	SVWN1RPA	B1LYP
B88	VWN5	BVWN1RPA	BHHLYP
XPW91	PW91LDA	SVWN5	B1PW91
GILL96	LYP	BVWN5	B3PW91
XPBE96	CPW91	SLYP	B3LYP1
OPTX	CPBE96	BLYP	B3LYP5
		GLYP	O3LYP1
		OLYP	O3LYP5
		XLYP	X3LYP
		BPW91	PBE1PW91
		PW91	PBE0
		PBEPW91	
		PBE96	

Finally, with respect to the calculation of static and dynamic (hyper) polarizabilities by TDDFT, it should be noted that while alpha values are exact, beta and gamma are only approximate at present as second-order (and higher) exchange-correlation kernels are not properly taken into account. This might be fixed in a future version of Firefly.

Excited state calculations with TDHF (RPA)

As shown in the examples above, TDHF theory for excited state calculations can be requested by the presence of the CITYP=TDHF keyword in the $CONTRL group. Current implementation allows the use of only RHF references, but can pick up both singlet and triplet excited states. Properties are available using "unrelaxed" density. Due to efficiency considerations, TDHF is programmed for SAPS (spin-adapted antisymmetrized product) basis only, so you cannot get both singlet and triplet states at once.

The number of states to be found (excluding the ground state) is controlled by NSTATE keywords. Specification of a state of interest for which properties will be calculated can be done via the ISTATE keyword. Only one state can be chosen. The state-tracking feature of the Firefly's TDHF code may be activated by selecting a negative value for ISTATE in the $TDHF group. Multiplicity (1 or 3) of the singly excited states can, as usual, be requested via the MULT keyword in $CONTRL.

Another important keyword is ISTSYM, which is used to specify the symmetry of the states of interest. Its default value is zero, which disables the use of symmetry during the TD calculation (i.e., states of all symmetries will be considered). Setting its value to the index of the desired irreducible representation (according to Firefly numbering) will produce only states of the desired symmetry and will also exploit the full (including non-abelian) symmetry of molecule, thus significantly reducing the computa-

tion time. Values for ISTSYM can be found in the output under "DIMENSIONS OF THE SYMMETRY SUBSPACES ARE". A tip: when using a UNIX-like OS, one can use *grep -A 3 "DIMENSIONS OF THE SYMMETRY SUBSPACES ARE" outputfile* to find this string quickly.

As an example, below is the output for butadiene, a molecule with C2h symmetry.

```
     DIMENSIONS OF THE SYMMETRY SUBSPACES ARE
 AG  =  28      AU  =   8      BU  =  28      BG  =   8
```

From the order in which the irreps are printed, one can see that:

A_g corresponds to ISTSYM=1,
A_u corresponds to ISTSYM=2,
B_u corresponds to ISTSYM=3,
B_g corresponds to ISTSYM=4,

For a better understanding of the ISTSYM keyword, let us take a closer look this butadiene example. We have seen now that the C2h point group contains four irreducible representations: A_g, B_g, A_u, and B_u. Hereby, only the transitions A_g->A_u, A_g->B_u, B_g->A_u, B_g->B_u are symmetry allowed. As a result, only two symmetry allowed excited states can be formed by single electron excitations: Au and Bu. Instead of solving TDHF for all four possible types of states, we can consider only symmetry allowed states and thus save on computation time. However, forbidden states can play a significant role in non-radiative relaxation processes such as intersystem crossing and internal conversion and should be considered if needed. It is important to note that oscillator strengths printed in the TDHF summary table are calculated using transition dipoles length form only.

There is an option to re-read vectors from a previous run. This can be helpful especially when one would like to obtain properties for more than one state. In this case, there is no need to recalculate the TDHF equations – all information can be read from $TDVEC group. This is achieved by specifying RDTDVC=.T. in the $TDHF group and copying the $TDVEC block from the PUNCH file of the converged TDHF run to the input file of the next run.

In the first example below, a TDHF calculation was performed for six single excited singlets (NSTATE=6); properties for the first excited state will be printed out (ISTATE=1); states of all possible summitries will be considered (ISTSYM=0).

Example 1:

```
 $CONTRL CITYP=TDHF RUNTYP=ENERGY $END
 $SYSTEM TIMLIM=525600 MEMORY=10000000 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $TDHF NSTATE=6 ISTSYM=0 ISTATE=1 $END
 $DATA
1,3-butadiene TDHF/6-31G(d)//RHF/6-31G(d)
CNH 2

 C           6.0   0.5262888328   0.5113710450   0.0000000000
```

```
 C           6.0   1.8213687868   0.2431548398   0.0000000000
 H           1.0   0.1953148875   1.5374180614   0.0000000000
 H           1.0   2.5595257633   1.0243655101   0.0000000000
 H           1.0   2.1906083178  -0.7681578804   0.0000000000
 $END
```

In the second example, a TDHF calculation is performed for four single excited singlets (NSTATE=4); properties for the second excited state will be printed out using the state-tracking feature (ISTATE=-2); only states of Bu symmetry will be considered (ISTSYM=3).

Example 2:

```
 $CONTRL CITYP=TDHF RUNTYP=ENERGY $END
 $SYSTEM TIMLIM=525600 MEMORY=10000000 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $TDHF NSTATE=4 ISTSYM=3 ISTATE=-2 $END
 $DATA
1,3-butadiene TDHF/6-31G(d)//RHF/6-31G(d)
CNH 2

 C           6.0   0.5262888328   0.5113710450   0.0000000000
 C           6.0   1.8213687868   0.2431548398   0.0000000000
 H           1.0   0.1953148875   1.5374180614   0.0000000000
 H           1.0   2.5595257633   1.0243655101   0.0000000000
 H           1.0   2.1906083178  -0.7681578804   0.0000000000
 $END
```

As was mentioned earlier, consideration of the symmetry specific states significantly speed up computation. In the presented examples the speed up was an approximate 2.3 times (!).

Excited state calculations with TDDFT

Excited state calculations with TDDFT are executed the same way as those with TDHF. Virtually everything mentioned in the previous section also applies to TDDFT. The only difference is that TDDFT excited state calculations are requested by specifying CITYP=TDDFT instead of CITYP=TDHF. Additionally, keywords pertaining to the TDDFT calculation belong to the $TDDFT group instead of the $TDHF group (keywords remain the same).

In addition to ‘normal’ TDDFT calculations, it is also possible to use the Tamm-Dancoff approximation to TDDFT (TDDFT/TDA). This can be requested by specifying TDA=.T. in the $TDDFT group.

In the example below a TDDFT calculation is performed in order to obtain four single excited triplet states (NSTATE=4 MULT=3); properties for the first excited state will be printed out (ISTATE=1); the states of all possible symmetries will be considered (ISTSYM=0).

Example 1:

```
 $CONTRL CITYP=TDDFT DFTTYP=PBE0 RUNTYP=ENERGY $END
 $SYSTEM TIMLIM=525600 MEMORY=10000000 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $TDDFT NSTATE=4 ISTSYM=0 ISTATE=1 $END
 $DATA
1,3-butadiene TDDFT/6-31G(d)//RHF/6-31G(d)
CNH 2

 C           6.0   0.5262888328   0.5113710450   0.0000000000
 C           6.0   1.8213687868   0.2431548398   0.0000000000
 H           1.0   0.1953148875   1.5374180614   0.0000000000
 H           1.0   2.5595257633   1.0243655101   0.0000000000
 H           1.0   2.1906083178  -0.7681578804   0.0000000000
 $END
```

The second example illustrates a TDDFT calculation performed for five singly excited singlet states (NSTATE=5); properties for the first excited state will be printed out (ISTATE=1); only states of Au symmetry will be considered (ISTSYM=2).

Example 2:

```
 $CONTRL CITYP=TDDFT DFTTYP=PBE0 RUNTYP=ENERGY $END
 $SYSTEM TIMLIM=525600 MEMORY=10000000 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $TDDFT NSTATE=5 ISTSYM=2 ISTATE=1 $END
 $DATA
1,3-butadiene TDDFT/6-31G(d)//RHF/6-31G(d)
CNH 2

 C           6.0   0.5262888328   0.5113710450   0.0000000000
 C           6.0   1.8213687868   0.2431548398   0.0000000000
 H           1.0   0.1953148875   1.5374180614   0.0000000000
 H           1.0   2.5595257633   1.0243655101   0.0000000000
 H           1.0   2.1906083178  -0.7681578804   0.0000000000
 $END
```

The third and final example illustrates a TDDFT calculation using the Tamm-Dancoff approximation to TDDFT (TDA=.T.). Six singly excited singlet states are requested (NSTATE=6); properties for the first excited state will be printed out (ISTATE=1); no symmetry will be used during the calculation (ISTSYM=0).

Example 3:

```
 $CONTRL CITYP=TDDFT DFTTYP=PBE0 $END
 $SYSTEM TIMLIM=525600 MEMORY=10000000 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $TDDFT NSTATE=6 ISTSYM=0 ISTATE=1 TDA=.T. $END
 $DATA
1,3-butadiene TDDFT/6-31G(d)//RHF/6-31G(d)
```

```
CNH 2

 C           6.0   0.5262888328   0.5113710450   0.0000000000
 C           6.0   1.8213687868   0.2431548398   0.0000000000
 H           1.0   0.1953148875   1.5374180614   0.0000000000
 H           1.0   2.5595257633   1.0243655101   0.0000000000
 H           1.0   2.1906083178  -0.7681578804   0.0000000000
 $END
```

Configuration Interaction methods

Introduction

Configuration interaction (CI) is a theory in which the wavefunction is described as a linear combination of Slater determinants. CI calculations in Firefly can be performed in a few different ways. First, there is a CI-singles (CIS) program which performs single excitations from a RHF reference wavefunction. Analytic gradients are available for this method.

Then, it is also possible to perform higher order CI calculations with the ALDET and GUGA programs. These programs were designed to perform MCSCF calculations such as FORS-MCSCF (also known as CASSCF), and multireference CI, but the GUGA program can also be used for performing single-reference CI calculations. The ALDET program is not capable of performing truncated CI calculations, it can only do full CI. This chapter will only focus on how to perform single-reference CI calculations. Multireference CI is discussed in the MCSCF chapter.

CI singles and higher-order CI calculations will be separately discussed in the next two sections.

CI singles

The CIS method is the simplest way to treat excited states. By Brillouin's Theorem, a single determinant reference such as RHF will have zero matrix elements with singly substituted determinants. The ground state reference therefore has no mixing with the excited states treated with singles only. The CIS method can be thought of as a non-correlated method, rigorously so for the ground state, and effectively so for the various excited states. Some issues making CIS not quite a black box method are:

a) Any states characterized by important doubles are simply missing from the calculation.
b) Excited states commonly possess Rydberg (diffuse) character, so the AO basis used must allow this.
c) Excited states often have different point group symmetry than the ground state, so the starting geometries for these states must reflect their actual symmetry.
d) Excited state surfaces frequently cross, and thus root flipping may very well occur.

The CIS code in the Firefly is based on heavily modified source code of the original GAMESS (US) AO-basis CIS program written by Simon P. Webb. Its current implementation allows the use of RHF reference wavefunctions only, but it can pick up both singlet and triplet excited states. Nuclear gradients are available, as are properties.

A CIS calculation can be requested with the keyword CITYP=CIS in the $CONTRL group. The input group with relevant keywords is the $CIS group. The most important keywords are NSTATE, which determines the amount of excited states to be found, and ISTATE, which determines for which state properties or a gradient should be calculated. Naturally, the value of NSTATE should be at least as high as the value of ISTATE. However, no error message is produced is this requirement is not met. In such a case, ISTATE will be set to be identical to NSTATE. In other words, be sure to check your input as the Firefly will not warn you if you set NSTATE too low. Setting ISTATE=0 will cause the Firefly to default to ISTATE=1.

It is possible to request only states of a specific symmetry using the ISTSYM keyword, provided you specified a point group in $DATA. ISTSYM=0 disables the use of symmetry while a non-zero value of ISTSYM requests states belonging to a certain irreducible representation. For more information on symmetry and the ISTSYM keyword, please see the section on TDHF.

Contrary to the TDHF and TDDFT codes, the CIS code can pick up singlet and triplet states at the same time, however it is not able to do this by default. In order to achieve this, it is necessary to choose a different CI type by using the HAMTYP keyword. The default CI type is HAMTYP=SAPS, which uses a spin-adapted antisymmetrized product basis and which can obtain states of only one multiplicity. The determinant based CI type, set through HAMTYP=DETS, can pick up both multiplicities, however, this CI type does have the downside that it will disable the use of fastints/gencon code during direct CIS runs (which will be detrimental to computation time).

The CIS program has a state-tracking feature which is activated by selecting a negative value of ISTATE. It is intended to be used for the geometry optimization of excited states in the case of root flipping.

An example input file for a CIS calculation on a water molecule:

```
 $CONTRL SCFTYP=RHF CITYP=CIS RUNTYP=ENERGY UNITS=ANGS $END
 $SYSTEM TIMLIM=3000 MWORDS=10 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $CIS NSTATE=3 ISTSYM=0 ISTATE=1 $END
 $DATA
H2O
CNV 2

 O 8.0 0.000000000  0.000000000 0.715595046
 H 1.0 0.000000000 -0.754094334 0.142202477
 $END
```

Note that oscillator strengths printed in the CIS summary table in the output are calculated using transition dipoles length form only. In addition, for degenerate irreps, computed oscillator strengths are for pure quantum states, so for most purposes one needs to multiply them by the factor equal to the degeneracy of the electronic states.

Instructions for the generation of state-averaged CIS natural orbitals can be found here:

http://classic.chem.msu.su/gran/gamess/forum/attach/cisnos_averaged.rar

Higher order CI

Higher order CI calculations can be performed with the ALDET and GUGA programs. These programs have two different approaches for doing CI calculations. ALDET, the Ames Labs DETerminant CI program, uses determinants to form the many electronic basis set. GUGA, the Graphical Unitary Group Approach CI program, on the other hand forms the many electronic basis set using configuration state functions (CSFs). As noted earlier, these two programs were designed for performing MCSCF (and MRCI) calculations. Therefore, many aspects related to these programs, including the difference between determinants and CSFs, are discussed in the chapter on MCSCF. This section will only focus on single-reference CI calculations. Since the ALDET program is only capable of full CI, the remainder of this section will discuss GUGA-style CI.

GUGA CSF-based CI can be requested with the keyword CITYP=GUGA in the $CONTRL group. The GUGA program was originally a set of different programs, so the input to control it is spread over several input groups. The CSFs are specified by a $CIDRT group. Other relevant input groups are $CISORT, $GUGEM, $GUGDIA, and $GUGDM. The $LAGRAN group can possible be relevant as well. Perhaps the most interesting variables outside the $CIDRT group are NSTATE in $GUGDIA to include excited states in the CI computation, and IROOT in $GUGDM to select the state for which properties and/or the gradient are to be calculated.

With CSF-based CI, the CSFs are ordinarily specified by giving a reference CSF together with a maximum degree of electron excitation from that single CSF. The MOs in the reference CSF are filled in the order of FZC first, followed by DOC, AOS, BOS, ALP, VAL, and EXT (the Aufbau principle). AOS, BOS, and ALP are singly occupied MOs (note that the amount of AOS should always be equal to the amount of NBOS). ALP means a high spin alpha coupling, while AOS/BOS are an alpha/beta coupling to an open shell singlet. For a single-reference CI calculation, this input can be kept very simple, an example being:

```
NFZC=3 NDOC=5 NVAL=34
```

which means the reference RHF wavefunction is:

```
FZC FZC FZC DOC DOC DOC DOC DOC VAL VAL ... VAL
```

In this case, NVAL is a large number conveying the total number of virtual orbitals into which electrons are excited. Note that NVAL's spelling was chosen to make the most sense for MCSCF calculations, and so it is a bit of a misnomer here. The excitation level can be set with the IEXCIT keyword, where IEXCIT=2 for example requests a CISD calculation. All excitations smaller than the value of IEXCIT are automatically included in the CI.

Before going on, there is a quirk related to single-reference CI that should be mentioned. Whenever the single-reference wavefunction contains unpaired electrons, such as

NFZC=3 NDOC=4 NALP=2 NVAL=33

some "extra" CSFs will be generated. The reference here can be abbreviated

2222 11 000 000 000 000 000 000 000 000 000 000 000

In the case of IEXCIT=2, the following CSF

2200 22 000 011 000 000 000 000 000 000 000 000 000

will be generated and used in the CI. Most people would prefer to think of this as a quadrupole excitation from the reference, but acting solely on the reasoning that no more than two electrons went into previously vacant NVAL orbitals, the GUGA CSF program decides it is a double. The result is that an open shell CISD calculation with Firefly will not give the same result as would be obtained with other programs, although the result for any such calculation with these "extras" is correctly computed.

Analytical gradients are available for CSF-based CI, however, at present gradient calculations cannot run in parallel and benefit little from multi-threading. Therefore, these calculations do not run very efficiently and might not be feasible with larger systems.

It is possible to request only states of a certain symmetry. This is done with the keywords GROUP and ISTSYM, which both belong to the $CIDRT group. GROUP can be used to specify the point group. The following point groups are supported: C1, C2, CI, CS, C2V, C2H, D2, D2H, C4V, D4, D4H. The desired irrep can be requested with ISTSYM. Note that, contrary to what is the case for CIS calculations, ISTSYM=0 does *not* disable symmetry (setting this will set the symmetry to be defined by the symmetry of the reference CSF). Instead, this is done by specifying GROUP=C1. For more information on the use of symmetry, please see the MCSCF chapter.

An example input file for a CISD energy calculation on a water molecule:

```
 $CONTRL SCFTYP=RHF CITYP=GUGA RUNTYP=ENERGY UNITS=ANGS $END
 $SYSTEM TIMLIM=10000 MWORDS=10 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
 $CIDRT GROUP=C1 ISTSYM=1 NFZC=1 NDOC=4 NVAL=14 IEXCIT=2 $END
 $GUGDIA NSTATE=5 $END
 $DATA
H2O
CNV 2

 O 8.0 0.000000000  0.000000000 0.715595046
 H 1.0 0.000000000 -0.754094334 0.142202477
 $END
```

Additional, performance related keywords are discussed in the MCSCF chapter.

Multi-configurational SCF methods

General information

Multi-configuration self-consistent field (MCSCF) wavefunctions are the most general SCF type, offering a description of chemical processes involving the separation of electrons (i.e. bond breaking, electronically excited states, etc.), which are often not well represented using single configuration SCF methods.

MCSCF wavefunctions, as the name implies, contain more than one configuration, each of which is multiplied by a "configuration interaction (CI) coefficient", determining its weight. In addition, the orbitals which form each of the configurations are optimized, just as in a simpler SCF, to self-consistency.

Typically, the uniqueness of each chemical problem requires that the design of an MCSCF wavefunction to treat it has to be done on a case by case basis. For example, one may be interested in describing the reactivity of a particular functional group instead of elsewhere in the molecule. This means that one has to choose carefully which configurations should be included in the MCSCF procedure in order to achieve a good description of the chemical problem. A popular way of doing this is by dividing the molecular orbitals into an “active” and an “inactive” space, where the active space contains the orbitals that are best described using multiple configurations. If within the active space all possible configurations are considered (similar to a full CI calculation within the active space), the approach is usually referred to as FORS-MCSCF (fully optimized reactive space MCSCF) or CASSCF (complete active space SCF), the latter name being used more often. Because of the importance of CASSCF, a large part of this chapter will focus on this theory. If within the active space only a part of all possible configurations is considered, the approach can be referred to as 'incomplete active space SCF'.

Using CASSCF allows one to calculate the static correlation energy in a chemical problem, but it does not calculate the dynamic correlation energy. A follow-up CI calculation, a method known as multireference CI, is one way to obtain the dynamic correlation, but has the drawbacks of not being size consistent and requiring a lot of computational resources. A more efficient technique implemented in Firefly for finding the dynamic correlation energy is second-order perturbation theory, in the variants known as MCQDPT and XMCQDPT. MRCI (though not a SCF method) is discussed in this chapter, while the MCQDPT and XMCQDPT theories are discussed in a separate chapter.

With the exception of the QUAD converger, the CASSCF program in Firefly is of the type termed "unfolded two step" by Roos. This means the orbital and CI coefficient optimizations are separated. The latter are obtained in a conventional CI diagonalization, while the former are optimized by a separate orbital improvement step.

Simplified, each CASSCF iteration consists of the following steps:
1) transformation of AO integrals to the current MO basis,
2) generation of the Hamiltonian matrix (not needed with direct CI),
3) optimization of the CI coefficients by means of a Davidson diagonalization,
3) generation of the first and second-order density matrix,
4) improvement of the molecular orbitals.

As was discussed in the previous chapter, the CI problem in steps 2 and 3 has two options, namely to use a determinant based CI code (ALDET) or a configuration state function (CSF) based CI code (GUGA). The choice between these is determined by the keyword CISTEP in the $MCSCF group: CISTEP=ALDET will request determinant-based MCSCF while CISTEP=GUGA will request CSF-based MCSCF. The differences between determinants and CSFs will be discussed later on.

The orbital problem in step 4 has four options. These are FOCAS, SOSCF, FULLNR, and QUAD, listed here in order of their computer resource requirements. These options will be discussed in the next section.

Information on the selection of the active space, ways to interpret the resulting MCSCF wavefunction, and the treatment for dynamical correlation not included in the MCSCF wavefunction can be found in the following references:

"The Complete Active Space Self-Consistent Field Method and its Applications in Electronic Structure Calculations" K. P. Lawley and B. O. Roos, Advances in Chemical Physics: Ab Initio Methods in Quantum Chemistry Part 2, Volume 69, 399-445 (1987)

"The Construction and Interpretation of MCSCF wavefunctions" M. W. Schmidt and M. S. Gordon, Ann. Rev. Phys. Chem. 49, 233-266 (1998)

"Multiconfigurational quantum chemistry for ground and excited states" B. O. Roos, Challenges and Advances in Computational Chemistry and Physics Vol. 5: Radiation Induced Molecular Phenomena in Nucleic Acids, 125-156 (2008)

"How to select active space for multiconfigurational quantum chemistry?" V. Veryazov, P. Å. Malmqvist, and B. O. Roos, Int. J. Quantum. Chem. 111, 3329-3338 (2011)

Orbital optimization options

There are presently four orbital improvement options, namely FOCAS, SOSCF, FULLNR, and QUAD. All four MCSCF orbital optimization methods were designed to efficiently run in parallel.

FOCAS is a first-order, complete active space MCSCF optimization procedure. The FOCAS code was written by Michel Dupuis and Antonio Marquez at IBM. It is based on a novel approach due to Meier and Staemmler, using very fast but numerous microiterations to improve the convergence of what is intrinsically a first-order method. Since FOCAS requires only one virtual orbital

index in the integral transformation to compute the orbital gradient (aka the Lagrangian), the total MCSCF job may take less time than a second-order method, even though it may require twice as many iterations to converge. The use of microiterations is crucial to FOCAS' ability to converge. It is important to take a great deal of care choosing the starting orbitals.

SOSCF is a method built upon the FOCAS code, which seeks to combine the speed of FOCAS with second-order convergence properties. It is an approximate Newton-Raphson method which starts at the diagonal guess to the orbital Hessian and then uses the limited memory BFGS (LBFGS) update approach to attain a quasi-linear, i.e. approximately quadratic convergence. Its time requirements per iteration are like FOCAS, with a convergence rate better than FOCAS but not as good as true second-order. LBFGS allows the SOSCF method to be used with much larger basis sets than exact second-order methods. Good convergence by the SOSCF method requires that you prepare starting orbitals carefully, and read in all MOs in $VEC, as the provision of canonicalized virtual orbitals increases the diagonal dominance of the orbital Hessian. It is important to note that the SOSCF converger in Firefly is much improved compared to the original SOSCF converger from GAMESS US. In most cases, SOSCF is better than FULLNR.

FULLNR means a full Newton-Raphson orbital improvement step is taken, using the exact orbital Hessian. FULLNR is a quite powerful convergence method, and normally takes the fewest iterations to converge. Computing the exact orbital Hessian requires two virtual orbital indices to be included in the transformation, making this step quite time consuming, and of course memory for storage of the orbital Hessian must be available. Because both the transformation and orbital improvement steps of FULLNR are time consuming, FULLNR is not the default. The FULLNR MCSCF code in Firefly uses the augmented Hessian matrix approach to solve the Newton-Raphson equations. There are two suboptions for computation of the orbital Hessian. DM2 is the fastest but takes more memory than TEI.

QUAD uses a fully quadratic, or second-order approach and is thus the most powerful MCSCF converger. QUAD runs begin with unfolded FULLNR iterations, until the orbitals approach convergence sufficiently. QUAD then begins the simultaneous optimization of CI coefficients and orbitals, and convergence should be obtained in 3-4 additional MCSCF iterations. The QUAD method requires building the full Hessian, including orbital/orbital, orbital/CI, and CI/CI blocks, which is a rather big matrix. QUAD may be helpful in converging excited electronic states, but note that you may not use state averaging with QUAD. QUAD is a memory hog, and so may be used only for fairly small numbers of configurations.

The default converger is SOSCF because it usually requires the least CPU time, disk space, and memory needs. Depending on the size of your calculation, however, there can be cases where FOCAS is faster so it is recommended you give this converger a try when convergence with SOSCF is generally smooth. If you cannot reach convergence with the SOSCF method, you can try to use the FULLNR method. However, this method should always be used with caution as it is able to reach convergence even with badly chosen active spaces, resulting in orbitals which from a chemical point of view make no sense anymore. When doing MCSCF calculations, it is always wise to regularly visualize your orbitals, but this goes double when using the FULLNR

method. Alternatively, SOSCF convergence can be often be improved by performing several FOCAS iterations to improve the active space followed by an inspection of the natural orbitals, which can then be used in a subsequent SOSCF calculation.

The input to choose the convergence method can be given in the $MCSCF group (e.g. SOSCF=.T.). The keywords MAXIT, ACURCY, and ENGTOL, which control the convergence behavior, also belong to this group. In some circumstances the diagonalizations of the core and virtual orbitals to canonicalize these (after overall MCSCF convergence) may produce spatial orbital symmetry loss, particularly in point groups with degeneracy present. The SD=.T. option can be used to preserve symmetry in this case.

Determinants vs CSFs

Either determinants or configuration state functions (CSFs) may be used to form the many electron basis set. It is necessary to explain these in a bit of detail so that you can understand the advantages of each.

A determinant is a simple object: an antisymmetrized product of spin orbitals with a given Sz quantum number, that is, the number of alpha spins and number of beta spins are a constant. Matrix elements involving determinants are correspondingly simple, but unfortunately determinants are not necessarily eigenfunctions of the S**2 operator.

To expand on this point, consider the four familiar 2e- functions which satisfy the Pauli principle. Here u,v are space orbitals, and a, b are the alpha and beta spin functions. As you know, the singlet and triplets are:

```
S1 = (uv + vu)/sqrt(2) * (ab - ba)/sqrt(2)
T1 = (uv) * aa
T2 = (uv - vu)/sqrt(2) * (ab + ba)/sqrt(2)
T3 = (uv) * bb
```

It is a simple matter to multiply out S1 and T2, and to expand the two determinants which have Sz=0,

```
D1 = |ua vb|          D2 = |va ub|
```

This reveals that

```
S1 = (D1+D2)/sqrt(2)   or   D1 = (S1 + T2)/sqrt(2)
T2 = (D1-D2)/sqrt(2)        D2 = (S1 - T2)/sqrt(2)
```

Thus, one must take a linear combination of determinants in order to have a wavefunction with the desired total spin.

There are two important points to note:

a) A two by two Hamiltonian matrix over D1 and D2 has eigenfunctions with -different- spins, S=0 and S=1.
b) use of all determinants with Sz=0 does allow for the construction of spin adapted states. D1+D2, or D1-D2, are -not- spin contaminated.

By itself, a determinant such as D1 is said to be "spin contaminated", being a fifty-fifty admixture of singlet and triplet (it is curious that calculations with just one such determinant are often called "singlet UHF"). Of course, some determinants are spin adapted all by themselves, for example the spin adapted functions T1 and T3 above are single determinants, as are the closed shells

```
S2 = (uu) * (ab - ba)/sqrt(2).
S3 = (vv) * (ab - ba)/sqrt(2).
```

It is possible to perform a triplet calculation, with no singlet states present, by choosing determinants with Sz=1 such as T1, since then no state with Sz=0 as is required when S=0 exists in the determinant basis set. To summarize, the eigenfunctions of a Hamiltonian formed by determinants with any particular Sz will be spin states with S=Sz, S=Sz+1, S=Sz+2, ... but will not contain any S values smaller than Sz.

CSFs are an antisymmetrized combination of a space orbital product, and a spin adapted linear combination of simple alpha-beta products. Namely, the following CSF

```
C1 = A (uv) * (ab-ba)/sqrt(2)
```

which has a singlet spin function is identical to S1 above if you write out what the antisymmetrizer A does, and the CSFs

```
C2 = A (uv) * aa
C3 = A (uv - vu)/sqrt(2) * ((ab + ba)/sqrt(2))
C4 = A (uv) * bb
```

equal T1-T3. Since the three triplet CSFs have the same energy, Firefly works with the simpler form C2. Singlet and triplet computations using CSFs are done in separate runs, because when spin-orbit coupling is not considered, the Hamiltonian is block diagonal in a CSF basis.

Technical information about the CSFs are that they use Yamanouchi-Kotani spin couplings, and matrix elements are obtained using a GUGA, or graphical unitary group approach.

The determinant implementation in Firefly can perform only full CI computations, meaning its primary use is for MCSCF wavefunctions of the complete active space type. The CSF code is capable of more general CI computations, and so can be used for first- or second-order CI computations. Other comparisons between the determinant and CSF implementations, as they exist in Firefly today, are

	determinants	CSFs
parallel execution	yes	yes
direct CI	yes	yes
exploits space symmetry	yes	yes
state average mixed spins	yes	no
first-order density	yes	yes
state averaged densities	yes	yes

can form CI Lagrangian	yes	yes

The default CISTEP in $MCSCF is ALDET, the Ames Laboratory determinant CI code.

The next two sections describe in detail the input for specification of the configurations, using determinants or CSFs.

Determinant CI code (ALDET)

The determinant CI code is capable only of full CI wavefunctions. Keywords associated with the determinant CI code belong to the $DET and $CIDET groups. The first of these is used for CASSCF calculations while the latter is used for full CI calculations. Despite having a different name, these two groups use almost the same keywords. Input for these groups is relatively simple -- many runs can be done by specifying only the orbital and electron counts: NCORE, NACT, and NELS.

The number of electrons is 2*NCORE+NELS and will be checked against the charge implied by ICHARG. The MULT given in $CONTRL is used to determine the desired Sz value, by extracting S from MULT=2S+1, then by default Sz=S. If you wish to include lower spin multiplicities, which will increase the CPU time of the run but will let you know what the energies of such states are, just input a smaller value for SZ. The states whose orbitals will be MCSCF optimized will be those having the requested MULT value, unless you choose otherwise with the PURES flag. An interesting feature that should be mentioned is that runs with Sz=0 use so-called NA = NB simplification and therefore, on average, twice as fast compared to runs with other values of Sz. This way, triplets can sometimes be computed faster using Sz=0 than the native Sz.

When one would like to obtain many states of a certain multiplicity, difficulties can be encountered when using the ALDET program as NSTATE will need to be set to a very high value. For example, if one would like to obtain 20 singlet states, NSTATE possibly might need be set as high as 60. With this many states, the Davidson diagonalization routine might fail due to loss of orthogonality and spin purity as a consequence of the multiple repeated reorthogonalizations and reconstructions of the expansion basis. In such a case, the ISPIN keyword can be used to filter out a part of the states prior to Davidson diagonalization. ISPIN=0 filters out states with odd S values (triplets, heptaplets, etc.) while ISPIN=1 filters out states with even S values (singlets, quintets, etc.). The NSTGSS keyword should be used to request enough states prior to filtering. An example for obtaining 20 states with an even S value:

```
 NSTATE=20 ISPIN=0 NSTGSS=60
```

The remaining parameters in $DET/$CIDET give extra control over the diagonalization process. Most are not given in normal circumstances, except NSTATE, which you may need to adjust in order to produce enough roots of the desired MULT value. The only important keyword which has not been discussed is the WSTATE array, which gives the weights for each state in form-

ing the first- and second-order density matrix elements, which drive the orbital update methods during MCSCF runs. Note that truly analytic gradients are available only when a pure state is specified, such as WSTATE(1)=0,1,0 which requests gradients of the first excited state to be computed. For state averaged MCSCF, gradients are instead obtained semi-numerically using a state-specific gradient calculation over state-averaged orbitals using a very efficient and accurate dedicated procedure based on the differentiation of the averaged gradient of SA-MCSCF (which is discussed in more detail further on in this chapter). When used for state averaged MCSCF, WSTATE is normalized to a unit sum, thus WSTATE(1)=1,1,1 really means a weight of 0.33333... for each of the states being averaged.

The ALDET code is able to exploit spatial symmetry, which, like the spin and charge, is implicitly determined by the choice of the reference CSF. The keyword GROUP in $DET/$CIDET governs the use of spatial symmetry. The ALDET code works with Abelian point groups, which are D2h and any of its subgroups. Fhe following point groups are supported: C1, C2, CI, CS, C2V, C2H, D2, D2H. For non-Abelian groups, the program automatically assigns the orbitals to an irrep in the highest possible Abelian subgroup. For the other non-Abelian groups, you must at present select an Abelian subgroup of the full point group or no symmetry at all (i.e. GROUP=C1). When symmetry is used, the desired irrep can be chosen with the keyword ISTSYM. Note that when you are computing a Hessian matrix, many of the displaced geometries are asymmetric, hence the program will automatically choose C1 in $DET/$CIDET (however, be sure to use the highest symmetry possible in $DATA!).

CSF CI code (GUGA)

The GUGA CSF package was originally a set of different programs, so the input to control CSF-based MCSCF is spread over several input groups, namely $DRT/$CIDRT, $GUGEM, $GUGDIA, $GUGDM, $GUGDM2, $CISORT, and $GUGDRT. The CSFs are specified by a $DRT group for MCSCF wavefunctions and by a $CIDRT group in the case of CITYP=GUGA. Thus, it is possible to perform an MCSCF calculation defined by a $DRT group (or a $DET group, which is also possible), and follow this up with a CI calculation defined by a $CIDRT group (or even a $CIDET group), in the same run.

Apart from the $DRT group, $GUGDIA, and $GUGDM2 are probably the most important groups for MCSCF runs. The most interesting variables outside the $DRT group are NSTATE in $GUGDIA to include additional states in the CI computation and WSTATE in $GUGDM2 to control which (average) state's energies are optimized. As with determinant-based MCSCF, truly analytic gradients are only available for pure states while state specific gradients for state averaged MCSCF are obtained semi-numerically (which will be discussed further on in this chapter).

The CSFs are specified by giving a reference CSF, together with a maximum degree of electron excitation from that single CSF. The MOs in the reference CSF are filled in the order MCC (MCSCF) / FZC (CI) first, followed by DOC, AOS, BOS, ALP, VAL, and EXT (the Aufbau principle). AOS, BOS, and ALP are singly occupied MOs. ALP means a high spin alpha coupling, while

AOS/BOS are an alpha/beta coupling to an open shell singlet. This requires the value NAOS=NBOS, and their MOs alternate. An example is

```
NMCC=1 NDOC=2 NAOS=2 NBOS=2 NALP=1 NVAL=3
```

which gives the reference CSF

```
MCC,DOC,DOC,AOS,BOS,AOS,BOS,ALP,VAL,VAL,VAL
```

This is a doublet state with five unpaired electrons. VAL orbitals are unoccupied only in the reference CSF, they will become occupied as the other CSFs are generated. This is done by giving an excitation level, either explicitly by the IEXCIT variable, or implicitly by the FORS, FOCI, or SOCI flags. One of these four keywords must be chosen, however, it is possible to use specify IEXCIT in addition to FORS/FOCI/SOCI to limit the excitation level inside the active space (thus creating an incomplete active space). More information on FOCI and SOCI can be found in the section on multireference CI.

Consider another simpler example:

```
NMCC=3 NDOC=3 NVAL=2
```

which gives the reference CSF

```
MCC,MCC,MCC,DOC,DOC,DOC,VAL,VAL
```

having six electrons in five active orbitals. Usually, MCSCF calculations are of the CASSCF/FORS type. These are enabled using the FORS keyword. In the present instance, choosing FORS=.T. gives an excitation level of 4, as the 6 valence electrons have only 4 holes available for excitation. MCSCF runs typically have only a small number of VAL orbitals. It is common to summarize this example as "six electrons in five orbitals" (which is often written as 'CASSCF(6,5)').

Note that, if you choose an excitation level IEXCIT smaller than that needed to generate the complete active space, you must use the FULLNR or SOSCF method (as FOCAS assumes complete active spaces). Be sure to set FORS=.F. in $MCSCF or else very poor convergence will result. Actually, the convergence for incomplete active spaces is likely to be poor anyway.

As was discussed above, the CSFs are automatically spin-symmetry adapted, with S implicit in the reference CSF. The spin quantum number you appear to be requesting in $DRT (basically, S = NALP/2) will be checked against the value of MULT in $CONTRL, and the total number of electrons, 2*NMCC(or NFZC) + 2*NDOC + NAOS + NBOS + NALP will be checked against the input given for ICHARG.

Like the ALDET code, the GUGA code is able to exploit spatial symmetry, which, like the spin and charge, is implicitly determined by the choice of the reference CSF. The keyword GROUP in $DRT/$CIDRT governs the use of spatial symmetry. The CSF program works with Abelian point groups, which are D2h and any of its subgroups. However, $DRT/$CIDRT allows the input of some (but not all) higher point groups -- the following point groups are sup-

ported: C1, C2, CI, CS, C2V, C2H, D2, D2H, C4V, D4, D4H. For non-Abelian groups, the program automatically assigns the orbitals to an irrep in the highest possible Abelian subgroup. For the other non-Abelian groups, you must at present select an Abelian subgroup of the full point group or no symmetry at all (i.e. GROUP=C1). When symmetry is used, the desired irrep can be chosen with the keyword ISTSYM (which overrides the symmetry computed from the reference). Note that when you are computing a Hessian matrix, many of the displaced geometries are asymmetric, hence the program will automatically choose C1 in $DRT/$CIDRT (however, be sure to use the highest symmetry possible in $DATA!).

In cases with high point group symmetry, it may be possible to generate correct state degeneracies only by using no symmetry (GROUP=C1) when generating CSFs. As an example, consider the 2-pi ground state of NO. If you use GROUP=C4V, which will be mapped into its highest Abelian subgroup C2v, the two components of the pi state will be seen as belonging to different irreps, B1 and B2. The only way to ensure that both sets of CSFs are generated is to enforce no symmetry at all, so that CSFs for both components of the pi level are generated. (Alternatively, for this specific example, one can use the C2 group as well as both pi orbitals belong to the same irrep.) This permits state averaging (i.e. WSTATE(1)=0.5,0.5) to preserve cylindrical symmetry. It is however perfectly feasible to use C4v or D4h symmetry in $DRT/$CIDRT when treating sigma states.

Performance

Over the years, many changes have been made to the Firefly code in order to optimize its performance. Improved algorithms have been written in order to achieve maximum performance and various new keywords allow one to choose or tune these algorithms in order to maximize the performance for a specific calculation. Many of these algorithms and their associated keywords were already discussed in the Performance chapter. The focus of this section is on the performance of MCSCF and (MR)CI type calculations.

Using FASTINTS/GENCON code

Throughout this section, we denote "large-scale" MCSCF calculations as systems having large number of basis functions (e.g. 500) and relatively small active spaces (e.g. 12 electrons/12 orbitals). In this case, the overall cost of calculations is mainly dominated by the integral transformation and effective Fock matrix construction steps. To speed up these stages, Firefly includes special fast direct and conventional integral transformation algorithms based on the fastints/gencon code (see the chapter on performance for details). These are available for any CI type, incl. CASSCF-type CI.

Some important keywords, all belonging to the $TRANS group, and their recommended values are:

```
$TRANS MPTRAN=2 DIRTRF=.T. AOINTS=DIST ALTPAR=.T. MODE=gsm $END
```

Here, the MPTRANS keyword instructs Firefly to select a fast alternative algorithm for integral transformation (fastints), using either its direct variant (preferred), or the conventional variant with 2-e integrals distributed over nodes, and selects an alternative (more scalable) parallel

strategy for MSCSF runs. The MODE keyword selects various suboptions. “gsm” is 3-digit decimal number defining the details of the direct parallel transformation code to be used. g can be either 0 or 1, and means either to use (1) or not to use (0) gencon version of the fastints code. s can be either 0 or 1, and means either to use (1) or not to use (0) approximate Schwarz integral screening (note that even if approximate screening is disabled, the exact Schwarz screening will be nevertheless in effect by default). Finally, m can be 0, 1, and 2, and denotes a small (0), medium (1), or large (2) active space. Thus, mode=112 is the most appropriate for most runs. For very small active spaces (typically 4 to 8 orbitals), mode=110 or 111 will perform faster, though one may find different depending on one's system and computer hardware. Note that fastints is only implemented for the FOCAS and SOSCF methods, not for the FULLNR method. Consequently, with FULLNR, MPTRANS cannot be set to 2.

The DIRTRF and AOINTS keywords serve to reduce the amount of I/O operations. ALTPAR duplicates some work but reduces communication and changes disk usage.

Packing the GUGA CI Hamiltonian file

When performing an MCSCF calculation with the GUGA program, an additional keyword of interest is:

```
$GUGEM PACK2=.T. $END
```

This selects (secondary) packing of GUGA CI Hamiltonian file, which works in conjunction with the FASTCI method of compression. This is a lossless method, which generally reduces the size of WORK16 file by a factor of up to 3-6 additionally to the FASTCI squeezing. Thus, the total degree of compression can reach the factor of 10-20 and even more, especially for the non-FORS type CI. Hence, this option is very useful in the case of relatively large CI calculations (i.e. 50000 CSFs or more, up to several millions), saving the disk space and reducing the wall clock time. Currently, the fastest way to perform any large GUGA CI/MCSCF calculation is to use this PACK2 method of packing. The only exception is the case of GUGA CI Hamiltonians of medium size that can entirely reside in the system file cache (which is limited only by the amount of the physical memory currently unused). For such jobs, the default packing settings are still faster.

MCSCF convergers

As noted in an earlier section of this chapter, the SOSCF and FOCAS orbital convergers are generally the fastest methods when performing an MCSCF type calculation. One important exception however is when you are using dozens of orbitals and many CSFs so that the main bottleneck is the diagonaization of the Hamiltonian, not the integral transformation and orbital improvement steps. For such a case, FULLNR can be a faster choice as it will minimize the total number of iterations. In addition, each orbital improvement may contain some microiterations, which consists of an integral transformation over the new MOs, followed immediately by an orbital improvement, reusing the current 2nd order density matrix. This avoids the CI diagonalization step every microiteration, which may be of some use in MCSCF calculations with a large number of configurations. Please note though that, as mentioned above, the FULLNR method currently does not support the use of fastints/gencon algorithm. Finally, since determinant based CI is a direct CI,

it is probably better to use it in this circumstance in order to avoid the very large disk file used to store the CSF Hamiltonian, and its associated I/O bottleneck.

Fast CI integral transformations

There are two keywords that pertain only to CI calculations:

```
$CIINP CASTRF=.T. $END
$TRANST CASTRF=.T. $END
```

The first selects fast MCSCF-like integral transformations for standalone CI runs, the latter fast MCSCF-like integral transformations for standalone CI transition moment/spinorbit runs. Sometimes these options may require much more memory than the standard CI integral transformations though, so they might not be the most economic depending on the size of your job.

Using the new ALDET CASCI/CASSCF diagonalization code

As of version 8.0.0, Firefly contains new code for performing MCSCF/CI diagonalizations with the ALDET program. Compared to the traditional (i.e. “old”) code, when running in parallel, the new code demands much less memory and scales better. To use the new code, one needs to specify a non-zero value for DISTCI (DISTCI stands for Distributed CI) variable:

```
$DET DISTCI=number_of_cores $END
```

or, for CASCI:

```
$CIDET DISTCI=number_of_cores $END
```

In a parallel run, the number of cores specified using the DISTCI variable is ignored, provided it is non-zero. Instead, the actual number of cores will be used. However, the value of the DISTCI variable is important for check runs (which can be run even using a single process) as it allows for the precise calculation of memory demands for a regular job running on the specified number of cores.

In the DISTCI mode, the memory demands are:

```
memory = const1 + const2/number_of_processes
```

where const2 is typically much larger than conts1. For comparison, when using the traditional code, the demands are:

```
memory = const1 + const2
```

Const1 can be estimated with:

```
const1 = MAXV * 2 * (number of determinants in CI)
```

The MAXV parameter defines the batch size over CI roots and can be specified as follows:

```
$(CI)DET MAXV=number $END
```

Allowed values are 1, 2 and 4. By default, MAXV is set automatically as follows:

- for single CI root runs, maxv is set to 1;
- for runs for two or three CI roots, maxv is set to 2;
- for larger number of roots, the default value is 4.

These settings are optimized for best performance.

Const2 can be estimated with:

```
const2 = MXXPAN * 2 * (number of determinants in CI)
```

where MXXPAN can be specified in the $(CI)DET group and defines the sub-space dimension of Davidson CI diagonalization. This keyword is described elsewhere in this manual.

One tip is that the number of CI roots in ALDET CI can be reduced provided one is interested in singlet states only. For example,

```
$DET ISPIN=0 $END
```

will filter out all states for which S is odd. As mentioned earlier, this option may require increasing the NSTGSS variable. More information can be found in the section on determinant-based CI.

CASSCF(2,2) vs GVB-PP(1)

A very common MCSCF wavefunction has 2 electrons in 2 active MOs. This is the simplest possible wavefunction describing a singlet diradical. While this function can be obtained in an MCSCF run (using NACT=2 NELS=2 or NDOC=1 NVAL=1), it can be obtained much faster by use of the GVB code, with one GVB pair. This GVB-PP(1) wavefunction is also known in the literature as two configuration SCF, or TCSCF. The two configurations of this GVB are equivalent to the three configurations used in this MCSCF, as orbital optimization in natural form (configurations 20 and 02) causes the coefficient of the 11 configuration to vanish. Please see the chapter on GVB for details on how to perform a TCSCF GVB calculation.

Performing state-specific MCSCF calculations

Below is an input example for ALDET-style state-specific CASSCF on a butadiene molecule:

```
 $CONTRL SCFTYP=MCSCF RUNTYP=ENERGY UNITS=ANGS $END
 $SYSTEM TIMLIM=10000 MWORDS=20 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=1 $END

 $MCSCF CISTEP=ALDET $END
 $DET GROUP=C2 ISTSYM=1 NCORE=13 NACT=4 NELS=4 NSTATE=1 WSTATE(1)=1,0 $END

 $GUESS GUESS=MOREAD NORB=17 $END
 $DATA
Butadiene
```

```
CN 2

 C           6.0  -1.5193996537   0.3119398887   0.5084798296
 H           1.0  -1.1071381034   0.6368872371   1.4479540422
 H           1.0  -2.5916948192   0.2697833502   0.4350941431
 C           6.0  -0.7391057353  -0.0052615754  -0.5349454028
 H           1.0  -1.2053069850  -0.2818783775  -1.4675271720
 $END
 $VEC
--input orbitals--
 $END
```

The example uses an active space of 4 electrons in 4 orbitals which could, for example, include the molecule's two pi bonding orbitals and the corresponding antibonding orbitals. Which orbitals should be included in the active space is highly dependent on the nature of the chemical problem. The example active space above would allow one to study the electrocyclic conversion of butadiene into cyclobutene, but would not be appropriate when studying, say, C–H bond breaking. In the end, the choice which orbitals to include is up to the researcher. Some strategies for getting chosen orbitals into an active space are discussed in the next section. Note that a CASSCF calculation should always start with the orbitals from a previous calculation, which can be read in with GUESS=MOREAD.

The same calculation, this time performed with the GUGA program:

```
 $CONTRL SCFTYP=MCSCF RUNTYP=ENERGY UNITS=ANGS $END
 $SYSTEM TIMLIM=10000 MWORDS=20 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=1 $END

 $MCSCF CISTEP=GUGA $END
 $DRT GROUP=C2 ISTSYM=1 FORS=.T. NMCC=13 NDOC=2 NVAL=2 $END
 $GUGDIA NSTATE=1 $END
 $GUGDM2 WSTATE(1)=1,0 $END

 $GUESS GUESS=MOREAD NORB=17 $END
 $DATA
Butadiene
CN 2

 C           6.0  -1.5193996537   0.3119398887   0.5084798296
 H           1.0  -1.1071381034   0.6368872371   1.4479540422
 H           1.0  -2.5916948192   0.2697833502   0.4350941431
 C           6.0  -0.7391057353  -0.0052615754  -0.5349454028
 H           1.0  -1.2053069850  -0.2818783775  -1.4675271720
 $END
 $VEC
--input orbitals--
 $END
```

If one is interested in a higher root, the first excited state for example, this state can be requested with WSTATE(1)=0,1. Consequently, NSTATE also has to be set higher in order to find enough states. With GUGA-style CASSCF, NSTATE=2 would be a sufficient value, though it is generally recom-

mended to request a few additional states if this is not too expensive. With ALDET-style CASSCF, NSTATE almost always should be set higher as the states found by the ALDET program are of various multiplicities, not just of the desired multiplicity (though the ISPIN keyword can be used to filter out states with certain multiplicities as described in the section on determinant-based CI).

Performing state-averaged MCSCF calculations

When studying quasi-degenerate or excited states, it is usually better to use state-averaged MCSCF. In Firefly, state-specific gradients for state-averaged orbitals are calculated in a semi-numerical fashion. In more detail, this approach is based on the differentiation of the effective gradient vector computed with state-averaged density matrices over the weight of the state of interest. The differentiation is performed using second-order finite differences and takes three state averaged MCSCF energy evaluations prior to calculating the gradient. As a result, state-specific gradients for SA-MCSCF are approximately two to three times more costly than gradients for MCSCF without state averaging. The approach is described in detail on:

http://classic.chem.msu.su/gran/gamess/ss-gradients.pdf

As the approach make use of differentiation, the computed gradients are more sensitive to numerical errors in the computed solution of SA-MCSCF than the ordinary MCSCF gradients are. Therefore, one needs to increase the overall precision of computations as will be discussed below. The MCSCF procedure itself can be arbitrary, i.e. it can be of any supported type, can use both GUGA and ALDET CI code, and can utilize any of available MCSCF convergers. However, the use of Firefly's unique SOSCF converger is strongly recommended. The relevant input groups are $MCSCF and $MCAVER. The most important keywords in the $MCSCF group are ISTATE, ACURCY, and ENGTOL, while the most important keywords in the $MCAVER group are CONIC, DELTAW, and HPGRAD.

The ISTATE keywords specifies the state number for which SS gradient will be computed. States are numbered starting from one. For example,

```
 $MCSCF ISTATE=2 $END
```

will compute gradients for the first excited state of a given multiplicity and symmetry type as specified by other sections of input file. The only exception is the case of ALDET CI with PURES=.T., for which the numbering will include all states regardless of their multiplicity. Please note that one should not use the ISTATE option (or specify a value of zero) for MCSCF runs without state-averaging or when state-specific gradients are not required! The default value is ISTATE=0, i.e. state-specific gradients are disabled.

The ACURCY and ENGTOL keywords pertain to two convergence criteria for MCSCF. ACURCY specifies the maximum permissible asymmetry in the Lagrangian matrix and is the major convergence criterion in MCSCF. While its default

value of 1.0E-05 is suitable for single-state MCSCF, computation of SS gradients for SA-MCSCF requires tighter convergence. Therefore, its recommended value lies in the range 1.0E-07 to 1.0E-08. ENGTOL specifies the maximum permissible energy change, i.e. the MCSCF is considered converged when the energy change is smaller than this value. The default is value is 1.0E-10. Again, as gradients for SA-MCSCF require tighter convergence, the recommended value of ENGTOL lies in the range 1.0E-12 to 1.0E-13.

As explained earlier, the method for obtaining gradients for state-averaged MCSCF is based on finite differencing. Control over this procedure is provided by the DELTAW and CONIC keywords. The DELTAW keyword sets the step size in the state's weight that is to be used. Its default value is 0.0015, while recommended values lie in the range of 0.0005 to 0.0025. The CONIC keyword can be used to specify one of three approaches for differencing. The default approach is CONIC=0, which selects the use of central (i.e. symmetric) second order finite differences. With this approach, the calculation is performed as follows. First, the calculation of the SA-MCSCF energies and effective gradient is performed with the original weight of the state specified with ISTATE *increased* by DELTAW. Secondly, the calculation of the SA-MCSCF energies and the effective gradient is performed with the original weight of the state specified with ISTATE *decreased* by DELTAW. Thirdly, the calculation of the SA-MCSCF energies is performed using the *unmodified* original weights, and finally, the state-specific expectation value type density matrix is computed for the state specified with ISTATE. This is the most economical way of computation. In addition, it provides the way to obtain the state-specific properties for the state of interest.

CONIC=1 selects an alternative approach that is based on the use of a forward finite differencing scheme. This is more stable than the CONIC=0 approach in the case of nearly quasi-degenerated CI roots and hence it is more suitable for location of conical intersections. The calculation with this approach is organized as follows. First, the calculation of the SA-MCSCF energies and the effective gradient is performed with the original weight of the state specified with ISTATE *increased* by DELTAW. Secondly, the calculations of the SA-MCSCF energies and the effective gradient is performed with the original weight of the state specified by ISTATE *increased* by DELTAW / 2. Finally, the calculation of the SA-MCSCF energies and the effective gradient is performed using *unmodified* original weights, but the state-specific density matrix is not computed, hence no state-specific properties are available.

Finally, CONIC=2 selects another approach that is similar to CONIC=1 but that is even more robust in the vicinities of conical intersections. First, the calculation on the SA-MCSCF energies and the effective gradient is performed with the *original* weight of the state specified with ISTATE. Secondly, the calculation of the SA-MCSCF energies and the effective gradient is performed with the original weight of the state specified by ISTATE *increased* by DELTAW / 2. Finally, the calculation of the SA-MCSCF energies and the effective gradient is performed with the original weight of target state *increased* by DELTAW. As with the CONIC=1 approach no state-specific density matrix is computed, thus the state-specific properties are not available.

Finally, the HPGRAD keyword should be mentioned. If set to .TRUE., it requests extra high precision during the computation of the two-electron contributions to the effective gradient. This may significantly slow down computations and usually does not considerably increase the precision of the computed state-specific gradients. This is why this option is disabled by default.

Typically, the separate stages involved in the MSCSF procedure are: evaluation of the two-electron integrals, integral transformation, CI procedure, computation of one- and two-particle density matrices, and the orbital improvement step. When calculating gradients for SA-MCSCF, one needs to increase the precision of each of these steps. Below is a synopsis of related input groups and keywords, with recommendations on the optimal values for the latter.

1. The precision of two-electron integrals is controlled by INTTYP, ICUT, and ITOL keywords of the $CONTRL group. For the calculation of gradients with SA-MCSCF, the recommended values are INTTYP=HONDO, ICUT=11 (12 or 13 is even better), and ITOL=20.

2. The precision of the integral transformation stage is controlled by the CUTTRF keyword of $TRANS group. For the calculation of gradients with SA-MCSCF, the recommended value of CUTTRF is 1.0D-13 or tighter.

3. The precision of the CI step is controlled by a single keyword for the ALDET CI code and by two keywords for the GUGA CI code. For the ALDET code, this is CVGTOL in the $DET group, its recommended value being 1.0D-7 to 1.0D-9. For GUGA CI, the relevant keywords are CUTOFF of the $GUGEM group and CVGTOL of the $GUGDIA group. The recommended values are CUTOFF=1.0D-20 and CVGTOL=1.0D-7 to 1.0D-9, respectively.

4. For GUGA CI, the precision of the two-particle density matrix computation step is controlled by the CUTOFF keyword of the $GUGDM2 group. The recommended value is CUTOFF=1.0D-15 or a tighter value. For ALDET CI, there is no need for any additional keywords.

5. The precision of the orbital improvement step can be improved using several keywords of $MOORTH and $SYSTEM groups. It is recommended to set:

```
 $SYSTEM KDIAG=0 NOJAC=1 $END
 $MOORTH   NOSTF=.T.   NOZERO=.T.   SYMS=.T.   SYMDEN=.T.   SYMVEC=.T.
SYMVX=.T.  TOLE=0.0D0 TOLZ=0.0D0 $END
```

An example input file is as follows:

```
 $CONTRL SCFTYP=MCSCF RUNTYP=GRADIENT UNITS=ANGS $END
 $SYSTEM TIMLIM=10000 MWORDS=20 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=1 $END

 $MCSCF CISTEP=GUGA ISTATE=2 $END
 $DRT GROUP=C1 ISTSYM=1 FORS=.T. NMCC=13 NDOC=2 NVAL=2 $END
```

```
 $GUGDIA NSTATE=4 $END
 $GUGDM2 WSTATE(1)=1,1 $END

 $GUESS GUESS=MOREAD NORB=17 $END

! Extra accuracy is needed
 $SYSTEM KDIAG=0 NOJAC=1 $END
 $CONTRL INTTYP=HONDO ICUT=11 $END
 $MOORTH NOSTF=.T. NOZERO=.T. SYMS=.T. SYMDEN=.T. SYMVEC=.T. SYMVX=.T.
 TOLE=0.0D0 TOLZ=0.0D0 $END
 $TRANS CUTTRF=1D-13 $END
 $GUGEM CUTOFF=1.0D-20 $END
 $GUGDIA CVGTOL=1.0D-7 $END
 $GUGDM2 CUTOFF=1.0D-15 $END
 $MCSCF ACURCY=1D-7 ENGTOL=1.0D-12 $END

 $DATA
Butadiene – gradient first excited state
CN 2

 C           6.0  -1.5193996537   0.3119398887   0.5084798296
 H           1.0  -1.1071381034   0.6368872371   1.4479540422
 H           1.0  -2.5916948192   0.2697833502   0.4350941431
 C           6.0  -0.7391057353  -0.0052615754  -0.5349454028
 H           1.0  -1.2053069850  -0.2818783775  -1.4675271720
 $END
 $VEC
--input orbitals--
 $END
```

In the above example, GROUP was set to C1 in order to allow state-averaging over two states with different symmetries.

Some additional examples can be found on the Firefly website:

http://classic.chem.msu.su/gran/gamess/ch2o_s0_s1.rar

MCSCF state-tracking

When investigating excited states, it is recommended to make use of state tracking as root switching can occur. This can be requested with NTRACK in the $MCSCF group. NTRACK=6 will, for example, track the 6 lowest states. NTRACK should be at least as high as the highest state included in WSTATE, and can naturally not be higher than the amount of states requested. State tracking is quite cheap, so tracking a few additional states does not have a large effect on performance.

The NTRACK value, if nonzero, activates MCSCF states tracking and remapping feature of Firefly. More precisely, NTRACK defines the number of lowest roots (states) to be tracked and, if necessary, remapped to other states. The present implementation of state tracking in Firefly is rather simple and is based on the analysis of overlap matrix of current states with the

previously computed ones. See the description of $TRACK group below for more details and for additional keywords. The default for NTRACK is NTRACK=0, i.e. do not track states at all. As said, state tracking is cheap and can be of great help e.g. during geometry optimization of excited states as it can detect root flipping and remap states accordingly, so it is generally a good idea to activate it.

Strategies for selecting an MCSCF active space

The first step in selecting an active space is to partition the orbital space into core, active, and external sets, in a manner which is sensible for your chemical problem. This is a bit of an art, and the user is referred to the references quoted at the end of this section. Having decided what MCSCF to perform, you now must consider the more pedantic problem of what orbitals to begin the MCSCF calculation with.

You should always start an MCSCF run with orbitals from some other run, by means of GUESS=MOREAD. Do not expect to be able to use HCORE or HUCKEL! If you are beginning your MCSCF problem, use orbitals from some appropriate converged SCF run. Once you get an MCSCF to converge, you can and should use these MCSCF MOs (which will be Schmidt orthogonalized) at other nearby geometries.

Starting from SCF orbitals can take a little bit of care. Most of the time (but not always) the orbitals you want to correlate will be the highest occupied orbitals in the SCF. Fairly often, however, the correlating orbitals you wish to use will not be the lowest unoccupied virtuals of the SCF. It is then necessary to change the order of the starting orbitals. This is done by specifying NORDER=1 and an accompanying IORDER array in $GUESS. Let’s for example consider a molecule with 15 occupied orbitals. In this example, visualization of your starting orbitals has shown you that the orbitals of interest are occupied orbitals 10, 13, 14, and virtual orbitals 18 and 19 (a six electrons, five orbitals active space). Therefore, your reorder instructions could be:

```
 NORDER=1 IORDER(10)=11,12,15,10,13,14,18,19,16,17
```

In this case, IORDER specifies that, starting from the 10th orbital, the new order of orbitals is 11, 12, 15, etc. Alternatively, one can also specify:

```
 NORDER=1 IORDER(10)=15 IORDER(15)=10 IORDER(16)=18,19,16,17
```

which simply switches orbitals 10 and 15. This may also be 'abbreviated' as:

```
 NORDER=1 IORDER(10)=-15 IORDER(16)=18,19,16,17
```

which functions the same. This might be the preferred syntax when one of the orbitals of interest lies very far from the active space – in such a case, reorder instructions in the fashion of the first example would require one to specify a very long array. Note that the exact order of the

orbitals is irrelevant as long as each orbital lies in its correct partition, e.g. core occupied orbitals lie in the core space, the orbitals of interest lie inside the active space, and external valence orbitals lie in the external valence space.

The occupied and especially the virtual canonical SCF MOs are often spread out over regions of the molecule other than "where the action is". Orbitals which remedy this can be generated by a few additional options at almost no CPU cost.

One way to improve SCF starting orbitals is by a partial localization of the occupied orbitals. Typically MCSCF active orbitals are concentrated in the part of the molecule where bonds are breaking, etc. Canonical SCF MOs are normally more spread out. By choosing LOCAL=BOYS along with SYMLOC=.T. in $LOCAL, you can get orbitals which are localized, but still retain orbital symmetry to help speed the MCSCF along. In groups with an inversion center, a SYMLOC=.T. Boys localization does not change the orbitals, but you can instead use LOCAL=POP. Localization tends to order the orbitals fairly randomly, so be prepared to reorder them appropriately.

Virtual SCF orbitals can generally be improved by the generation of modified virtual orbitals (MVOs). MVOs are obtained by diagonalizing the Fock operator of a very positive ion, within the virtual orbital space only. As implemented in Firefly, MVOs can be obtained at the end of any RHF, ROHF, or GVB run by setting MVOQ in $SCF nonzero, at the cost of a single SCF cycle. A good setting is MVOQ=+6. Generating MVOs does not change any of the occupied SCF orbitals of the original neutral, but gives more valence-like LUMOs.

Pasting the virtuals from a MVOQ run onto the occupied orbitals of a SYMLOC run (both can be done in the same SCF computation) often gives a good set of starting orbitals. If you also take the time to design your active space carefully, select the appropriate starting orbitals from this combined $VEC, and inspect your converged results, you will be able to carry out MCSCF computations correctly.

An additional option for obtaining good starting orbitals is to use the NBO program present in Firefly, which can convert the occupied orbitals into natural bond orbitals (NBOs) and the virtual orbitals in natural localized molecular orbitals (NLMOs). As opposed to MVOs, NLMOs are localized and might therefore be an even better choice as starting orbitals.

Convergence of MCSCF is by no means guaranteed. Poor convergence can invariably be traced back to either a poor initial selection of orbitals, or a poorly chosen number of active orbitals. Good advice is, before you even start: "Look at the orbitals. Then look at the orbitals again". Later, if you have any trouble: "Look at the orbitals some more". Few people are able to see the orbital shapes in the LCAO matrix in a log file, so a visualization program will be of great help.

Even if you don't have any trouble, look at the orbitals to see if they converged to what you expected, and have reasonable occupation numbers. MCSCF is by no means the sort of "black box" that RHF is, so look very carefully at your results.

Multireference configuration interaction

Before discussing how to perform MRCI calculations in Firefly, it is important to note that the MRCI implementation in Firefly is not MRCI in the classical sense of the term. Instead, the FOCI and SOCI methods in Firefly should be seen as variants of MRCI. Let us consider that in a given system with a given active space the orbitals can be grouped as follows:

```
doubly occupied orbitals – active space – valence space
```

In 'classical' MRCI, excitations are allowed from doubly occupied orbitals into the active space, from the active space into the valence space, and from doubly occupied orbitals into the valence space. FOCI and SOCI, however, only allow excitations from the active space into the valence space. These excitations are either singles (FOCI, a variant of MRCIS) or singles and doubles (SOCI, a variant of MRCISD). Below is an example of a FOCI or SOCI wavefunction.

```
NFZC=3 NDOC=3 NVAL=2 NEXT=-1
```

This leads to the reference CSF

```
FZC,FZC,FZC,DOC,DOC,DOC,VAL,VAL,EXT,EXT,...
```

In this example, all CSFs with N electrons (in this case N=6) are distributed in the valence orbitals in all ways (that is, the CASSCF wavefunction) to make the base wavefunction. To this, FOCI adds all CSFs with N-1 electrons in active and 1 electron in external orbitals while SOCI adds all CSFs with N-2 electrons in active orbitals and 2 in external orbitals. SOCI is often prohibitively large, but is also a very accurate wavefunction. FOCI or SOCI is chosen by selecting the appropriate flag (FOCI=.T. or SOCI=.T.). Hereby, the correct excitation level is automatically generated. The value of -1 for NEXT is shorthand and causes all remaining virtual MOs to be included in the external orbital space.

For the butadiene CASSCF example given earlier, SOCI input could look like this:

```
 $CONTRL SCFTYP=MCSCF CITYP=GUGA RUNTYP=ENERGY UNITS=ANGS $END
 $SYSTEM TIMLIM=10000 MWORDS=20 $END
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=1 $END

 $MCSCF CISTEP=ALDET $END
 $DET GROUP=C1 ISTSYM=1 NCORE=13 NACT=4 NELS=4 NSTATE=1 WSTATE(1)=1 $END
 $CIDRT GROUP=C1 ISTSYM=1 NFZC=13 NDOC=2 NVAL=2 NEXT=-1 SOCI=.T. $END
 $GUGDIA NSTATE=1 $END

 $GUESS GUESS=MOREAD NORB=90 $END

 $DATA
Butadiene
CN 2
```

```
 C           6.0  -1.5193996537   0.3119398887   0.5084798296
 H           1.0  -1.1071381034   0.6368872371   1.4479540422
 H           1.0  -2.5916948192   0.2697833502   0.4350941431
 C           6.0  -0.7391057353  -0.0052615754  -0.5349454028
 H           1.0  -1.2053069850  -0.2818783775  -1.4675271720
 $END
 $VEC
--input orbitals--
 $END
```

Note that GROUP in $DET and $CIDRT is set to C1. It is also possible to use the GUGA MCSCF program instead of the ALDET one:

```
 $MCSCF CISTEP=GUGA $END
 $DRT GROUP=C1 ISTSYM=1 NMCC=13 NDOC=2 NVAL=2 FORS=.T. $END
 $CIDRT GROUP=C1 ISTSYM=1 NFZC=13 NDOC=2 NVAL=2 NEXT=-1 SOCI=.T. $END
 $GUGDIA NSTATE=1 $END
```

Because the FOCI and SOCI methods only allow excitations from the active space into the valence space, the active space in the above example is probably too small to lead to results that are significantly more accurate than the CASSCF results. More accurate results can be obtained by increasing the size of the active space during the MRCI step. The active space of the preceding MCSCF step can be also increased, however, this is not required as Firefly does not check or enforce the active space to be same in the MCSCF and MRCI steps. An example:

```
 $MCSCF CISTEP=GUGA $END
 $DRT GROUP=C1 ISTSYM=1 NMCC=13 NDOC=2 NVAL=2 FORS=.T. $END
 $CIDRT GROUP=C1 ISTSYM=1 NFZC=4 NDOC=11 NVAL=2 NEXT=-1 SOCI=.T. $END
 $GUGDIA NSTATE=1 $END
```

Here, the inclusion of more doubly occupied orbitals in the active space during MRCI creates a wavefunction that is much closer to 'classical' MRCISD, though it should be noted that, strictly speaking, this calculation can no longer be considered as canonical MRCI. A downside of this approach though is that the number of CSFs increases quickly as more orbitals are added to the active space. In the preceding example, the number of CSFs is even bigger than would be the case with 'classical' MRCISD.

In order to keep the size of the calculation manageable, the number of CSFs in an MRCI calculation can be decreased by limiting the excitation level. This is done by specifying a value of IEXCIT together with SOCI=.T./FOCI=.T. An example:

```
 $CIDRT GROUP=C1 ISTSYM=1 NFZC=7 NDOC=18 NVAL=6 NEXT=-1 FOCI=.T. IEXCIT=3
$END
```

In this example, specifying IEXCIT=3 will limit the maximum excitation level to 3. This will cause the active space to contain only CISDT excitations

(recall that a complete active space normally contains all possible excitations, i.e. full CI). The entire CSF space will in addition also include FOCI-type CSFs that are singly excited with respect to the NDOC/NVAL active space and that have an overall excitation level that is less than or equal to 3. A word of caution: limiting the number of excitations in the active space can exclude CSFs from the calculation that are important for describing multi-reference cases. So, when a multi-reference description is important for your system, this approach has a chance of giving you incorrect results.

MRCI calculations can also be started from other wavefunctions, such as GVB and RHF wavefunctions. It is even possible to specify SCFTYP=NONE, provided that a set of converged orbitals is read in using GUESS=MOREAD (the full list of orbitals including all virtuals must be given). Below is an example of a GVB-PP (TCSCF) calculation that is followed up with a MRCI calculations). Because gradients not available for MRCI, TRUDGE is used to optimize the geometry.

```
 $CONTRL SCFTYP=GVB CITYP=GUGA RUNTYP=TRUDGE COORD=HINT $END
 $BASIS GBASIS=N31 NGAUSS=6 $END

 $SCF NCO=3 NPAIR=1 $END

 $CIDRT GROUP=C2V SOCI=.TRUE. NFZC=1 NDOC=3 NVAL=1 NEXT=-1 $END
 $GUGDIA NSTATE=5 $END
 $GUGDM IROOT=3 $END

 $TRUDGE OPTMIZ=GEOMETRY NPAR=2 IEX(1)=21,22 P(1)=1.08 $END

 $DATA
Methylene TCSCF+CISD geometry optimization
CNV 2
 C 6. LC 0.00 0.0 0.00 - O K
 H 1. PCC 1.00 53. 0.00 + O K I
 $END
```

(Extended) Multi-configurational quasi-degenerate perturbation theory

Introduction

Perturbation Theory (PT) is one of the most successful yet relatively inexpensive tools of quantum chemistry. In particular, the variant of PT suggested by Møller and Plesset[1] (MPn) turned out to be a very powerful tool for computational ground-state quantum chemistry already at the second order (MP2)[2-3]. However, it is well known that simple single-reference constructs, which the MPn is, work very well for electronic states dominated by a single Slater determinant, but fail severely for multi-configuration wavefunctions. Several successful generalizations of MP or MP-like approaches to the case of multi-configuration reference states (MR-PT) have been suggested during last two decades. Among them are the MR-MP2 approach by Hirao[4-7], CASPT2 approach by Andersson, Malmqvist and Roos[8-11], and NEVPT2 by Angeli *et al*[12-15]. An overview of other approaches to MR-PT can be found in Ref. 16.

Nevertheless, the single-state approaches to MR-PT assume the "diagonalize-then perturb" philosophy and require a zero-order wavefunction that is already well described by CASSCF or a similar procedure. In practice, this is more the exception than the rule; for this reason, the possibility for several reference states to mix within the MR-PT treatment is very important. The natural way to allow this mixing is to use the multi-state (MS) formulations of MR-PT approaches leading to various MS-MR-PT theories. Specifically, the MS formulation of MR-MP2 has long been known as the MCQDPT approach developed by Nakano[17-23]. The MS formulation of CASPT2 known as MS-CASPT2 was originally suggested by Finley *et al*[24] while the MS version of NEVPT2 known as QD-NEVPT2 has been developed by Angeli *et al*[25]. All of them are of the "diagonalize-then perturb-then diagonalize" type. In particular, this means that instead of perturbing the entire CASCI Hamiltonian, only several selected CI roots are perturbed and mixed, *i.e.* these methods employ the so-called partial contraction in the space of CI expansion coefficients. These approaches are more or less directly related to the formal theory of the effective Hamiltonians and Quasi-Degenerate Perturbation Theories (QDPT) that have been well established for a long time[26-37]. Essentially, all three theories define approximations to the exact effective Hamiltonian acting within a model space[33,37], *i.e.* a subspace spanned by the selected CI roots. Energies of perturbed states are then obtained as eigenvalues of effective Hamiltonian while projections of perturbed states onto zero-order states are defined by the corresponding eigenvectors.

Besides direct mixing of target states, the effective Hamiltonians technique allows indirect relaxation *via* interaction with higher-energy CASCI states. When entering effective Hamiltonians, higher-energy states admix with target states, thus allowing relaxation of low-lying states within the CASCI space and improving their overall description. Indeed, CASSCF usually tends to converge to solutions that are too delocalized in the space of CI

coefficients, so additional states in the model space are often needed to eliminate this deficiency at MS-MR-PT level.

The most important distinctions in the formulation of various MS-MR-PT schemes are the selection of a zero-order Hamiltonian and of so called "perturbers", *i.e.* zero-order states that are allowed to interact with zero-order wavefunctions in the PT treatment. In the case of MCQDPT2, "perturbers" are simply all possible individual CSFs or Slater determinants not belonging to the CASCI space and obtained applying single and double excitations individually to every CSF (determinant) entering the reference states. In contrast, both MS-CASPT2 and QD-NEVPT2 are formulated using state-specific and more sophisticated "perturbers" defined as specific linear combinations of some selected classes of CSFs or determinants. Depending on the particular scheme in use, the latter approach is known as an either internal[8,9,38] or external[39,40] contraction. For instance, most of CASPT2 variants can be considered as internally contracted approximations to the MR-MP2, while MS-CASPT2 theories are internally contracted approximations to the MCQDPT2.

Both the MR-MP2 and the CASPT2 approach applies a one-particle Fock-like zero-order Hamiltonian. In contrast, the NEVPT2 scheme adopts a partially bi-electronic zero-order Hamiltonian initially suggested by Dyall[41]. The situation is more complicated for the MS-MR-PT counterparts of these approaches.

Overview of MCQDPT

The MCQDPT (Multi-Configuration Quasi-Degenerate Perturbation Theory) approach was suggested by Nakano[17] as a multi-state generalization of the MR-MP2 theory by Hirao[4]. In this paragraph, we review the basics of MCQDPT. Additional details can be found in the original papers on this theory[17-23].

MCQDPT is a multi-state multi-reference Van-Vleck-type perturbation theory of the partially contracted type that employs isometric (i.e., unitary through any PT order) normalization[42-44]. More precisely, a model space is spanned by several CI vectors (i.e the partial contraction is used) that are typically obtained as a result of the state-averaged CASSCF procedure (the **P** subspace), while the secondary, first-order interacting space is formed by the individual CSFs or determinants and thus is not contracted (the **S** subspace). It is also helpful to define the **O** subspace as a subspace of CASCI vectors that are complementary to **P**, with **P**⊕**O** forming the entire CASCI subspace **R**, and **O**⊕**S** forming the subspace **Q**, as shown in Fig. 1.

MCQDPT applies $\boldsymbol{H}^{0}$ which is constructed using a diagonal model Fock operator $\hat{F}$. More precisely, MCQDPT applies individual blocks of $\boldsymbol{H}^{0}$ that are defined *via* expectation values of $\hat{F}$ using projectors onto individual CI vectors and perturbers. The particular choice of interacting space and zero-order Hamiltonian in MCQDPT theory results in a $\boldsymbol{H}^{0}$ that is fully diagonal in **P**, **O**, and **S** subspaces. The attractive consequence of this feature is a very simple expression for the Resolvent that eliminates any need to solve large systems of linear equations. Instead, the second-order effective Hamiltonian is expressed as the direct sum of different contributions corresponding

to each particular class of diagrams, with energy denominators of a very simple structure. This opens a way to a very efficient implementation of MCQDPT2 as was shown in[45-46] and implemented within Firefly. In particular, the so-called Resolvent-fitting (a.k.a. table-driven) approach is very computationally efficient[45]. The current implementation allows MCQDPT2 calculations of systems with active spaces up to several millions of CSFs and with the overall number of molecular orbitals up to 2000-3000 to be routinely performed on a standalone single-CPU workstation or desktop computer.

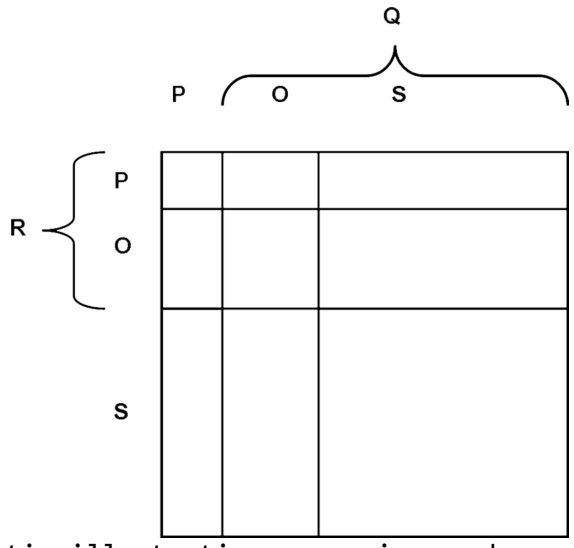


Fig. 1. Schematic illustration on various subspaces.

However, H^0_{mcqdpt} does not have any well-defined algebraic transformation properties with respect to transformations of basis within the model space. Moreover, for any non-trivial model space H^0_{mcqdpt} is actually a many-particle operator, and hence, the perturbation is also artificially made to be a many-particle operator rather than the two-particle one as would naturally expect. These two points apply equally to MS-CASPT2 and QD-NEVPT2 theories. More precisely, MCQDPT results in perturbation being a many-particle operator in the **R** subspace, MS-CASPT2 results in perturbation being a many-particle operator in both **R** and **S** subspaces, while QD-NEVPT2 results in perturbation being a many-particle operator in the **S** subspace. It is worth noting that both original MS-CASPT2 and QD-NEVPT2 apply internal contraction and a multi-partitioning scheme by Zaitsevskii and Malrieu[47] which introduces an additional source of non-invariance into these theories.

For instance, let us assume one fixes the number of CI roots and values of their corresponding weights in computing the averaged one-particle density matrix that is used to define the model Fock operator $\hat{F}$. In this case, the effective second-order MCQDPT2 Hamiltonians of progressively decreasing dimension form the trivial sequence with the $n \times n$ effective operator being exactly equal to the $(n+1) \times (n+1)$ operator with the row and column, corresponding to the $(n+1)^{th}$ extra CI root, being removed. Evidently, there is no any explicit dependence of the effective Hamiltonian on the dimension of

model space in this approach, as the $n \times n$ effective operator is simply enclosed into $(n+1) \times (n+1)$ one.

The XMCQDPT theory

The XMCQDPT (Extended Multi-Configuration Quasi-Degenerate Perturbation Theory) theory[46] is a novel multistate formulation of MR-MP2 employing invariant zero-order Hamiltonian. Being the alternative to MCQDPT2, it is superior to MCQDPT2 and is free of multiple drawbacks of the latter while obeying several important physical and mathematical properties that are desirable for any partially contracted MS-MR-PT-based approach. These properties are discussed below and can be considered as an attempt to bring various formulations of MS-MR-PT into better accordance with the formal theory of Effective Hamiltonians.

In practical applications, various MR-PT and especially MS-MR-PT theories can be plagued by the so-called "intruder state problem"[48,49] which causes the appearance of very small energy denominators in PT series, hereby leading to spurious results of the entire PT calculation. The persistence of this problem varies with the particular theory. For instance, it has been demonstrated that NEVPT2 is inherently less prone to this problem as compared with MR-MP2 and CASPT2[50,51]. However, the problem still persists in the QD version of the NEVPT2 approach on par with other MS-MR-PT schemes. For simplicity, we assume below that intruder states are either irrelevant, or that they were successfully eliminated using, for instance, intruder state avoidance[52] (ISA) or similar techniques.

First, we assume that ***the effective Hamiltonian should be explicitly dependent on the dimension of the model space in any nontrivial order of PT***. Evidently, this dependence should not be just a trivial one caused by the simple extension of the model space. It also should not be related only to the variations of orbital energies, orbitals, or both, caused by the averaging of one-particle density matrices over a varying number of CI roots, and hence, should not be related only to the implicit dependence of the effective Fock operator(s) and zero-order Hamiltonian on the dimension of the model space. Indeed, exact effective Hamiltonians depend explicitly on the model space dimension[46]. Hence, approximations to the exact effective Hamiltonian should approximate this explicit dependence on the model space size as well. It should be realized that, in general, the restriction of an effective Hamiltonian to a subspace of the model space is not a new effective Hamiltonian associated with this smaller subspace.

Second, a very important property is ***the convergence of energies as well as other observable properties with respect to model space extension at any fixed order of PT*** provided that the active space used in underlying MCSCF calculations is well-balanced and accurately designed so that there are no spurious zero-order states (*i.e.* solutions of non-linear MCSCF equations which do not describe real excited states[53-55]). This property is evidently mandatory for the low-lying electronic states that are typically of interest and is desirable for other states as well.

The third property is that ***the effective Hamiltonian should be a function of the subspace spanned by the selected CI vectors, rather than a function***

of any particular choice of basis in this subspace. In particular, this means that the effective Hamiltonian should be the same regardless of whether the initial CI vectors, or any alternative vectors obtained as their arbitrary orthogonal transformation, were used for MS-MR-PT computations. Indeed, as many-body perturbation theory (MBPT) is a purely algebraic approach, it has well-defined algebraic invariance properties. Hence, it is natural to impose the same restriction on the MS-MR-PT. A very important consequence of this property is that ***the correctly formulated partially-contracted theory should be exactly equivalent to the corresponding uncontracted theory provided that the dimension of the model space is equal to the dimension of the CI space*** (*i.e.* the overall number of CSFs or determinants spanning the CI space).

Finally, the most important and physically evident requirement is that ***the computed energies must be uniquely defined, continuous, and smooth functions of the molecular geometry and any other external parameters***, with possible exceptions at the manifolds of their accidental degeneracy such as conical intersections.

In modeling of many-electron molecular systems, the correct dependence of computed energies and other properties on the number of electrons in the correlation treatment plays crucial role. This requirement is related to such features of the underlying theories as exact or approximate size-consistency[56] and separability[57-60]. It is often believed that the exact size-consistency is very desirable for any high-level computational approach designed to describe large systems. However, we do not consider small deviations from the exact size-consistency as a serious deficiency of MS-MR-PT theories, provided that these deviations are more or less constant regardless of the model space dimension and do not cause any significant unphysical effects even for large systems. It seems that the exact or approximate core-separability[60] is the most important requirement, while small deviations from the strict separability are quite acceptable. In particular, the possibility of applying the correctly formulated MS-MR-PT-based methods to the calculation of vertical electronic excitation energies should not be affected by these deviations.

The difference between MCQDPT and XMCQDPT is the choice of the $\boldsymbol{H^0}$ operator. Namely, XMCQDPT employs $\boldsymbol{H^0}$ that is much closer to the one-particle Fock operator $\hat{F}$. More precisely, unlike MCQDPT, the **PP** and **QQ** blocks of $\boldsymbol{H^0}$ is non-diagonal in the basis of CASCI eigenvectors and is formed by the entire projection of $\hat{F}$ onto **P** and **Q** subspaces.

It can be easily shown that any unitary transformation of the basis set in the model space transforms the second-order effective Hamiltonian $\boldsymbol{H^{(2)}_{eff}}$ of XMCQDPT according to the usual law of transformation of an arbitrary linear operator so that XMCQDPT2 (XMCQDPT at the second order of PT) satisfies the invariance property above. Hence, the XMCQDPT2 theory is evidently equivalent to the fully uncontracted approach provided that the dimension of the model space is equal to the dimension of the CI space. The “explicit dependence on the model space extension” property is also satisfied taking into account the non-diagonal nature of the **PP** block of $\boldsymbol{H^0}$ in the basis of CI vectors forming the model space.

As has been mentioned above, MCQDPT, MS-CASPT, and QD-NEVPT2 theories assume the **PP** block of $\boldsymbol{H}^{0}$ that is defined using projectors onto individual CI vectors and do not have well-defined transformation properties with respect to the unitary transformation of the model space. This results in the non-invariant perturbation theories and introduces an artificial many-particle nature into the perturbation. In contrast, XMCQDPT applies $\boldsymbol{H}^{0}$ resulting in the invariant theory with balanced perturbation being an approximation to the true two-particle operator in the **R** subspace and the true two-particle operator in the **S** space. A discussion on the importance of balanced perturbation in PT treatment can be found in Ref. 13 in the context of NEVPT approach.

It is natural to expect that non-invariance should lead to significant erratic behavior of the non-invariant theories in the regions of the molecular geometries where the corresponding CI vectors undergo rapid change or are not uniquely defined at all. More precisely, all non-invariant theories are not defined at the geometries corresponding to conical intersections at the underlying MCSCF level, and behave badly in the vicinities of the avoided crossings.

In multi-reference electronic structure calculations for mixed electronic states, the relative contributions of different reference configurations tend to differ substantially between the reference SA-CASSCF wavefunctions and the final correlated wavefunction. For instance, the position of the avoided crossing point between the neutral and ionic potential energy curves of the $^{1}\Sigma^{+}$ states of the LiF molecule differs by ca. 3 Bohr between SA-CASSCF and MRSDCI calculations. Obviously, in the region between SA-CASSCF and correlated crossing points, the quality of the zero-order wavefunctions obtained using SA-CASSCF procedure degrades. This can be a serious issue for non-invariant MS-MR-PT approaches defined using projectors as this can result in an incorrect description of the interstate mixing, and thus, the PESs. In contrast, this situation is much better for invariant theories such as XMCQDPT2. Indeed, the final correlated states depend only on the subspace spanned by the zero-order wavefunctions (*i.e.* on the model space) rather than on the zero-order wavefunctions themselves. Therefore, if the number of physically important CSFs shared by the zero-order SA-CASSCF states is less or equal to the dimension of model space, one can *a priori* expect a better and more balanced description of the interstate mixing and PESs by invariant MS-MR-PT theories as compared with non-invariant formulations.

MCQDPT2 is neither size-consistent nor core-separable. In contrast, different variants of CASPT2 as well as NEVPT2 are exactly core-separable; moreover, NEVPT2 is exactly size consistent and strictly separable. While analysis of their MS counterparts is more complicated, both theories are at least exactly core-separable.

In the special case of a one-dimensional model space, XMCQDPT2 is completely equivalent to the MR-MP2 approach[4]. Due to the structure of its energy denominators, MR-MP2 is neither exactly size-consistent, nor core-separable. This can be easily verified by performing test computations on simple model systems like the Be_2 molecule at large internuclear distances. However, the deviations from exact size-consistency are typically small.

Keeping these properties of MR-MP2 in mind, it is logical to expect that XMCQDPT2 should in general not be exactly size-consistent or core-separable as well. This is indeed the case. However, we mention as an important fact that the fully uncontracted limit of XMCQDPT is an exactly size-consistent and separable theory through the fourth order of Van-Vleck PT expansion[61]. Assuming that the convergence of energies with an increase of the dimension of the model space is reasonably fast, XMCQDPT2 should be approximately size-consistent and core-separable for low-lying states even for the quite modest dimensions of effective Hamiltonians, with size-inconsistency errors rapidly decreasing upon the extension of the model space.

Let XMCQDPT2' and MCQDPT2' denote the modified (and thus approximate) XMC-QDPT2 and MCQDPT2 calculations that apply the MP2-like expression for double excitations from double occupied inactive orbitals to external MOs. This approximation eliminates the dominant contribution violating exact core-separability and makes both theories almost exactly core-consistent. While MCQDPT2' is typically an improvement over MCQDPT2, XMCQDPT2' is not usually an improvement over XMCQDPT2 as XMCQDPT2' introduces approximations not needed by the parent theory while not significantly improving the core-separability properties of the computed XMCQDPT2 energies.

MCQDPT2 and MS-CASPT2 tend to overestimate off-diagonal elements of effective Hamiltonians[62-70]. This is true, although to a lesser extent, for MCQDPT2' as well. Indeed, one can consider MCQDPT2 as an approximation to XMCQDPT2 that completely neglects the off-diagonal elements of $\boldsymbol{H}^{0}$ in the CASCI space. The quality of this approximation depends on the magnitude of the neglected off-diagonal elements. If the zero-order interaction of the states forming the model space is small, these two theories should be in the close agreement. However, as soon as zero-order interaction increases, MCQDPT2 performance starts to degrade. For instance, one can *a priori* expect large zero-order interactions in situations when two or more CI roots share a common set of leading CSFs, resulting in questionable applicability of MCQDPT2 as well as of all other non-invariant theories formulated using projectors onto the model space. The neglect of off-diagonal terms introduces inaccuracy into both diagonal and off-diagonal elements of effective Hamiltonians.

In the case of MCQDPT2', the leading MP2-like contribution to the correlation energy is treated differently so it results only in the equal shift of all diagonal elements of $\boldsymbol{H}^{(2)}{}_{eff}$. Hence, this reduces the accumulated errors in off-diagonal elements of $\boldsymbol{H}^{(2)}{}_{eff,\ mcqdpt}$. However, this does not eliminate errors entirely.

XMCQDPT *vs* XMS-CASPT2

Soon after the paper on XMCQDPT theory was published, the XMS-CASPT2 theory (*i.e.* the Extended Multi-State Second Order Complete Active Space Perturbation Theory) was suggested by Shiozaki, Győrffy, Celani and Werner (Ref. 71), who based their work on the ideas of the XMCQDPT paper (Ref. 46). The XMS-CASPT2 theory is an improvement over standard MS-CASPT2 and can be considered as an internally-contracted approximation to XMCQDPT2. Typically one may expect similar results from MS-CASPT2 and XMCQDPT2. However, the Resolvent fitting technique cannot be applied to XMS-CASPT2 as this theory

does not allow direct summation of PT series. This makes XMCQDPT2 computationally much more efficient compared with XMS-CASPT2, especially for large molecular systems.

Useful hints and best practices

A. Understanding basic steps of MCQDPT2 and XMCQDPT2 execution flow with Firefly

To activate one of XMCQDPT2 and MCQDPT2, one needs to use $CONTRL MPLEVL=2 together with SCFTYP=MCSCF. The choice between XMCQDPT2 and MCQDPT2 theories is controlled by the presence of the corresponding input groups. *I.e.* if the $MCQDPT group is given in the input, Firefly will perform MCQDPT2 computations. Otherwise, if the $XMCQDPT or $QDPT group is given in the input, XMCQDPT2 computations will be performed. All three groups share a common set of input variables; and, as explained in details in Ref. 46, XMCQDPT2 and MCQDPT2 computations with Firefly use the same common code. In the following we use (X)MCQDPT2 to term either of two. There are three flags and one extra input group that significantly affect execution flow of (X)MCQDPT2 computations.

First, (X)MCQDPT2 can be preceded by MCSCF computations. This is the most typically used scenario. However, there is a possibility to skip the MCSCF part. To do this, one needs to provide a complete $VEC group with converged orbitals, read in orbitals using $GUESS GUESS=MOREAD with the NORB variable equal to the overall number of MOs, and set the INORB variable of the $(X)MCQDPT group to 2 (default is INORB=0) to indicate that the MCSCF stage should not be performed. It is recommended to use orbitals saved with extra high precision, *i.e.* with the $CONTROL WIDE flag set to .TRUE. (default is wide=.FALSE.).

Second, the (X)MCQDPT2 code performs canonicalization of input or previously computed MOs by default. This requires some additional computations, namely the extra CASCI-type integral transformation (performed by the MQTRMC routine), CASCI computations (MQCACI routine), first order state-averaged density matrix construction, Fock matrix construction, and its block-diagonalization (MQFORB routine). Note, the WSTATE array is used in computing weights of roots for state-averaged density matrix calculation at this stage.

However, input orbitals or orbitals generated by the preceding MCSCF procedure can already be in semi-canonical form, so that user can request Firefly to skip canonicalization. This can be done by setting the IFORB variable of the $(X)MCQDPT group to 0 (the default is IFORB=1, *i.e.* not to skip canonicalization).

Third, one can request either the exact (X)MCQDPT2 (the default setting) or the approximate (X)MCQDPT2'. The latter option is activated by setting the IROT variable of the $(X)MCQDPT group to 1 (default is 0). Computationally, (X)MCQDPT2' is a bit cheaper than (X)MCQDPT2. As was discussed above, (X)MCQDPT2' theories also have better properties if the approximate core-separability is of concern. However, one should realize that (X)MCQDPT2'

theories are the approximations to the exact second-order answer given by the (X)MCQDPT2 theories.

Finally, the presence of the $MCQFIT group triggers the use of Resolvent fitting (a.k.a. table-driven) code[45]. Additional options can be specified in $MCQFIT group allowing fine control over Resolvent fitting code.

Regardless of the variables described above, the rest of computations is organized as follows. First, the MP2-type integral transformation is performed by the MQTRF routine. This can be a disk-based or semi-direct transformation, whereby the latter is selected by specifying $TRANS DIRTRF=.TRUE. and is highly recommended. Then, molecular integrals are rearranged and a CASCI procedure in the basis of semi-canonical Fock orbitals is performed (MQCACI). Next, provided the optional CSF selection option is turned on, the number of CSFs entering the PT computations is reduced based on user-specified criteria, and the final CI procedure is performed in this reduced subspace. Note that CSF selection is disabled by default. Finally, the one-particle state-averaged density matrix is recomputed using the AVECOE array, then the Fock matrix and the effective orbital energies are constructed by the routine MQORB2, and some additional setup is performed by the MQLPR1 and MQLPR2 routines (alternatively by MQLPR1, MQLPRB and MQLPRC). At this point, the code is ready for summation of PT series, which is performed for contributions of different types by the routines MQLMB1, MQLMB2, MQLMB3 and MQLMBR/MQLMBR0/MQLMBR1/MQLMBR2 called in a big loop over all CI states forming the model space.

(Footnote: Note that both the WSTATE and AVECOE arrays are used to define the state's weights used to compute state-averaged density matrices. Technically, the averaging used to define semi-canonical orbitals and to define orbital energies of these orbitals can be different. If both WSTATE and AVECOE are given in the input file, the first one defines the averaging used to compute orbitals while the second is used for energies. If only one of them is given in input, then the second one will be set to be identical with the first one. The latter option should be used almost exclusively unless one has really good reasons for using different averaging for different stages.)

At the very end, the MQGETE routine (which is preceded by the MQVSML routine in the case of PT with CSF selection) is called and prints computed energies and zero-order QDPT properties: dipole moments, transition moments, and so on.

B. Planning (X)MCQDPT2 computations

1. Use XMCQDPT2 rather than MCQDPT2

2. When absolutely in need to use MCQDPT2, use MCQDPT2'

3. Use Intruder State Avoidance (ISA) denominator shift. The default shift is zero. To set ISA shift to 0.02 Hartree (the recommended value), use $(X)MCQDPT EDSHIFT=0.02.

4. For reliable results, use an at least triple zeta + polarization functions basis sets such as cc-pVTZ or better.

5. Use the same averaging in SA-CASSCF (defined by the WSTATE array of $DET or $GUGDM2), and in (X)MCQDPT2 (defined by the WSTATE and AVECOE arrays of $(X)MCQDPT) unless there are strong reasons for the opposite. Use all weights equal to unity, once again unless there are strong reasons for the opposite.

6. Add some extra CASCI states to a model space, *i.e.* use a value of NSTATE that is larger than the number of states in SA-CASSCF averaging.

C. Running (X)MCQDPT2 computations

1. Use Firefly's efficient and fast SOSCF or FOCAS convergers for MCSCF stage of job.

2. Use the Resolvent fitting-based code for large (X)MCQDPT2 jobs as it is much faster than the default code. The use of the Resolvent fitting-based code is triggered by the presence of the $MCQFIT group in the input file. In most cases, it is sufficient to specify only $MCQFIT $END without altering any parameters of the $MCQFIT group - the default parameters are selected so that the errors introduced in the computed energies are typically less than 10^{-8} Hartree. There are three adjustable parameters in the $MCQFIT group:

a. DELTAE - double precision, default is zero. If nonzero, this variable is used to define the step of the interpolation grid.

b. NPOINT - integer, default is 400. Together with DELTAE, this variable is used to define the step size and the number of points of the interpolation grid. The actual grid size will be the least of the two, the number specified using the NPOINT variable and the number of points computed based on the value of DELTAE. Increasing NPOINT will result in an increase of the required CPU time while reducing errors in the computed energy.

c. IORDER - integer variable. Available values are 3, 5, and 7. Defines the order of the polynomial to use in interpolation. The default value is 7.

3. Use the efficient direct and semi-direct integral transformation code for large (X)MCQDPT2 jobs *i.e.* specify $TRANS DIRTRF=.TRUE. in the input file. One can control large (MP2-style) integral transformations (MQTRF stage) using the appropriate variables of $MP2 group. In addition, it is best to use Firefly's specific large-scale efficient CASCI/CASSCF integral transformation code during MQTRMC stage by setting the dedicated keyword in the $TRANS group as follows: $TRANS MPTRANS=2 DIRTRF=.TRUE. $END. Other variables that could be useful are $TRANS ALTPAR, MODE, and CUTTRF. These variables are documented elsewhere in the manual.

4. The PT series summation in Firefly's (X)MCQDPT2 code does not run in parallel at present but is efficiently threaded. To speed up computations, use SMP and/or multi-core systems and allow Firefly to use extra computing threads. One can do this by specifying the overall number of threads to use via ether the $SYSTEM NP or $SMP NP variables (these two variables are synonyms). Note, the $SMP HTTNP variable, which can be either 1 or 2 and which

is described elsewhere, affects the actual number of threads when running in an environment that supports hyper-threading.

5. The preceding CASSCF stage and the semidirect MQTRF/ALTTRF integral transformation stage of (X)MCQDPT2 are allowed to run in parallel so if one runs a (X)MCQDPT2 job in parallel, these stages will be executed in parallel. The summation of PT series will however be executed in serial by the master Firefly's process. To speed up this serial stage, use threading as described above. In addition, one needs to set the $SMP SMPPAR=.TRUE. which indicates that created additional working threads should only be used during the execution of serial parts of code *i.e.* the use of a specific type of mixed parallel/threaded model of execution. Note this requires (the low-level) P2P protocol and dynamic load balancing. Thus, the input for a mixed parallel/multithreading (X)MCQDPT2 run (either with our without a preceding CASSCF step in the same run) would look as follows (for a 8 core system without HyperThreading support):

```
 $SYSTEM MKLNP=1 NP=8 $END
 $SMP SMPPAR=.T. $END
 $P2P P2P=.T. DLB=.T. $END
```

6. For the most typically used (X)MCQDPT2 runs involving a preceding CASSCF step in the same run, it is possible skip to first MQCACI and integral transformation by instructing the (X)MCQDPT2 code to reuse information from the CASSCF step. This can be done by specifying the following:

```
 $MCSCF IFORB=.TRUE. $END
 $(X)MCQDPT IFORB(1)=-1,1,1 $END
```

Hereby, it is recommended to also specify $MCSCF SD=.T. These keywords are described in the list of keywords.

7. The numerical gradient based code and all runtypes involving numerical gradients are allowed to execute all stages of (X)MCQDPT2 in parallel using XP and Extended XP models of execution, which are described elsewhere. In addition, running in the Extended XP mode, one can still set $SMP SMPPAR=.TRUE. to allow the mixed parallel/threaded model of execution within each separate group of processes.

8. Use the MASMEM keyword of $SYSTEM group to alter the amount of memory used by the master process (or XP local master processes). The PT summation stage of (X)MCQDPT2 code typically requires more memory than the preceding CASSCF and MQTRF integral transformation stages. With this option, one can allow a master Firefly process to use more memory than other Firefly's instances. *E.g.* the line **$SYSTEM MWORDS=100 MASMEM=200 $END** would allow the master to use 200 Mwords with regular instances of parallel Firefly process still consuming 100 Mwords each. While using MASMEM, it is a good idea to perform exetyp=check run in parallel mode. Alternatively, one can run it twice in serial (once for MWORDS without specifying MASMEM, and once for MWORDS with the desired value of MASMEM).

9. If the (X)MCQDPT2 run runs out of memory, one can consider:

a. decreasing the value of the $(X)MCQDPT MXBASE variable (the dimension of subspace in Davidson diagonalization, which affects the amount of memory used by routine MQCACI);

b. decreasing the number of threads used during the PT summation stage and disabling the use of additional HTT threads (*i.e.* setting $SMP HTTNP=1) (this affects MQLMB1, MQLMB2, MQLMB3, and MQLMBR/R0/R1/R2);

c. limiting the amount of the stack allotted to Firefly (for Linux OSs, ulimit -s 2048 should normally work);

d. allowing Firefly to use additional memory in its heap: $(X)MCQDPT HALLOC=.TRUE.;

e) allowing the selection of most prominent CSFs with $(X)MCQDPT ISELCT(1)=(one of 1, 2, 3, 4, -1, 2, -3, -4) THRWGT="the smallest CSF's weight (i.e. the squared coefficient of CSF in CI expansion) to keep". The default value of THRWGT is 10-6. One can try smaller values e.g. 10-8. By default, ISELCT(1) is equal to 0, i.e. no selection is allowed. The values of 1, 2, 3, and 4 select slightly different schemes of CSFs selection. For the values 2, 3, and 4, it is also possible to specify a value for ISELCT(2) that can be interpreted as the maximum allowed number of CSFs.
For large-scale MCQDPT2 and XMCQDPT2 computations, one of the most important considerations defining the amount of the memory required is the size of the "envelope" of the set of CSFs, *i.e.* the set of all CSFs that can be generated by applying all possible single electron excitation and de-excitation operators acting within the active space to the given set of CSFs. The CSF selection scheme works by first generating a primary set of CSFs using the threshold defined by THRWGT. After this, the primary set is extended in a several different ways provided that the generated envelope will be the same. At this stage, CSFs with small weights can be added if they are free. The size of the envelope is not allowed to exceed the value of ISELCT(2), which gives the maximum allowed size of the envelope. The size of the primary set is reported by the MQSLT2 routine, the size of the envelope by the MQCOND routine.
The least advanced method of selection is method #1, which does not allow one to specify a maximum size for the envelope. The most advanced (and recommended) scheme is method #4, although the difference between different methods is rather subtle. It should be noted that if ISELCT(2) is not given (or set to 0), the primary set of CSFs is selected without limits on the envelope size. It is then extended by several possible ways but again with no restriction on the size of the envelope. This way, it is actually possible to include all CSFs into the extended set of CSFs. Conversely, if the envelop size is kept quite constrained (a small value for ISELCT(2) is given), CSFs with weights higher than THRWGT may be dropped.
The values of -1, -2, -3, and -4 for ISELCT(1) are the counterparts of their positive analogs which in addition perform the rotation of CI roots computed in the incomplete space after CSFs selection. This rotation is within the space spanned by these CI roots and is constructed to maximize the overlap of rotated states with the initial

CASCI states before CSFs selection. This option can be important for MCQDPT2 but is ignored for XMCQDPT2 as results of XMCQDPT2 depend only on the subspace spanned by the CI vectors rather than on the particular basis in this subspace (i.e. CI vectors).

D. Analyzing results

1. Check the resulting second-order Effective Hamiltonian (search the output for "*** EFFECTIVE HAMILTONIAN (0-2) ***") for unusually large off-diagonal elements (say 0.05 Hartree or above). In the case there are some, one may need to reexamine the appropriateness of the underlying (SA-)CASSCF procedure, esp. active orbitals, the entire active space, state-averaging protocol, etc.

2. Check the numbers labeled by "E(MP2)=" (these are the energies of states within the (X)MCQDPT2 theory) for sense. This table is labeled as "*** (X)MC-QDPT2 ENERGIES ***"

3. Check the table containing "EIGENVECTORS OF THE EFFECTIVE HAMILTONIAN" for unexpectedly strong/unphysical mixing of the zero-order CASCI states. Is this exactly what could be expected? If it is not, strong mixing may indicate deficiencies in the underlying CASSCF description of the zero-order states. Is there any considerable admixture of higher-energy roots to the target states? If so, one may need to consider including higher-energy states into SA-CASSCF state-averaging procedure.

4. Check that the target "EIGENVALUES OF THE NON-SYMMETRIC EFFECTIVE HAMILTONIAN" are all real and are all close to the eigenvalues of the symmetric Effective Hamiltonian i.e. numbers labeled by "E(MP2)=" as mentioned above. Large deviations and large imaginary contributions indicate deficiencies in the PT treatment.

5. Check that the "OVERLAP NORM MATRIX OF NON-ORTHOGONAL EIGENVECTORS" does not contain large off-diagonal values, *e.g.* values larger than 0.1 or 0.2. If the off-diagonal overlap is large this typically means that the target states are extremely poorly described by the model space and that they have a (very) large norm of projection on the secondary subspace *i.e.* they are living outside the CASCI space.

6. Check that the energies of states (or at least the transition energies) are stable with respect to the extension of a model space.

7. Analyzing molecular properties printed at the end of (X)MCQDPT2 run, take in mind that the computed values are the zero-order QDPT properties. This means that they include only a part of all effects due to interaction with the secondary space. Namely, they correctly include effects causing rotation and intermixing of zero order states (i.e. CASCI vectors) within the model space but do not include the first order correction to the wavefunction (*i.e.* the part of the perturbed states which belongs to the secondary space). Thus, the printed properties should be considered as approximations to the "true" (X)MCQDPT2 properties.

E. Citing MCQDPT2 and XMCQDPT2

One should cite Firefly as explained in this manual and on the Firefly homepage. In addition, one needs to cite XMCQDPT2 as follows: **A. A.**

Granovsky, J. Chem. Phys. 134, 214113 (2011). For MCQDPT2, the proper reference is: **H. Nakano, J. Chem. Phys. 99, 7983 (1993).**

References

1 C. Møller and M. S. Plesset, Phys. Rev. 46, 618 (1934).
2 J. A. Pople, J. S. Binkley, and R. Seeger, Int. J. Quantum Chem., Quantum Chem. Symp. S10, 1 (1976).
3 M.J.Frisch, M.Head-Gordon, J.A.Pople, Chem. Phys. Lett. 166, 275 (1990).
4 K. Hirao, Chem. Phys. Lett. 190, 374 (1992).
5 K. Hirao, Chem. Phys. Lett. 196, 397 (1992).
6 K. Hirao, Int. J. Quant. Chem. S26, 517 (1992).
7 K. Hirao, Chem. Phys. Lett. 201, 59 (1993).
8 K. Andersson, P-Å. Malmqvist, B. O. Roos, A. J. Sadlej, K. Wolinski, J. Phys. Chem. 94, 5483 (1990).
9 K. Andersson, P.-Å. Malmqvist, and B. O. Roos, J. Chem. Phys. 96, 1218 (1992).
10 K. Andersson, B. O. Roos, Int. J. Quant. Chem. 45, 591 (1993).
11 B. O. Roos, K. Andersson, M. P. Fülscher, P.-Å. Malmqvist, L. Serrano-Andrés, K. Pierloot, and M. Merchán, in Advances in Chemical Physics: New Methods in Computational Quantum Mechanics edited by I. Prigogine, S. A. Rice (Wiley, New York, 1996) Vol. XCIII, p. 219.
12 C. Angeli, R. Cimiraglia, S. Evangelisti, T. Leininger, and J.-P. Malrieu, J. Chem. Phys. 114, 10252 (2001).
13 C. Angeli, R. Cimiraglia, and J.-P. Malrieu, J. Chem. Phys. 117, 9138 (2002).
14 C. Angeli, R. Cimiraglia, and J.-P. Malrieu, Chem. Phys. Lett. 350, 297 (2001).
15 C. Angeli, S. Borini, and R. Cimiraglia, Theor. Chem. Acc. 111, 352 (2004).
16 R. K. Chaudhuri, K. F. Freed, and G. Hose, P. Piecuch, K. Kowalski, and M. Włoch, S. Chattopadhyay and D. Mukherjee, Z. Rolik, Á. Szabados, G. Tóth, and P. R. Surján, J. Chem. Phys. 122, 134105 (2005)
17 H. Nakano, J. Chem. Phys. 99, 7983 (1993).
18 H. Nakano, Chem. Phys. Lett. 207, 372 (1993).
19 H. Nakano, K. Nakayama, K. Hirao, M. Dupuis, J. Chem. Phys. 106, 4912 (1997).
20 T. Hashimoto, H. Nakano, K. Hirao, J. Mol. Struct. (THEOCHEM) 451, 25 (1998).
21 H. Nakano, R. Uchiyama, K. Hirao, J. Comput. Chem. 23, 1166 (2002).
22 M. Miyajima, Y. Watanabe, H. Nakano, J. Chem. Phys. 124, 044101 (2006).
23 R. Ebisuzaki, Y. Watanabe, H. Nakano, Chem. Phys. Lett. 442, 164 (2007).
24 J. Finley, P.-Å. Malmqvist, B. O. Roos, and L. Serrano-Andrés, Chem. Phys. Lett. 288, 299 (1998).
25 C. Angeli, S. Borini, M. Cestari, and R. Cimiraglia, J. Chem. Phys., 121, 4043 (2004)
26 C. Bloch, Nucl. Phys. 6, 329 (1958).
27 B. H. Brandow, Rev. Mod. Phys. 39, 771 (1967).
28 B. H. Brandow, Int. J. Quant. Chem. 15, 207 (1979);
29 I. Lindgren, J. Phys. B 7, 2441 (1974).

30 I. Lindgren, Int. J. Quant. Chem. 14 (S12), 33 (1978).
31 G. Hose and U. Kaldor, J. Phys. B 12, 3827 (1981).
32 J.-P. Daudey and J.-P. Malrieu, in Current Aspects of Quantum Chemistry 1981 (Ed R. Carbó) , p. 35 (Elsevier 1982).
33 Ph. Durand, J.-P. Malrieu, in Adv. Chem. Phys. 67, (Ed. K. P. Lawley), p. 321 (Wiley 1987).
34 K. F. Freed and M. G. Sheppard, J. Chem. Phys. 75, 4507 (1981)
35 K. F. Freed and M. G. Sheppard, J. Chem. Phys. 75, 4525 (1981).
36 V. Hurtubise and K. F. Freed, in Adv. Chem. Phys., 83 (Eds. I. Prigogine and S. A. Rice), p. 465 (Wiley 1993).
37 J.-P. Malrieu, J.-L. Heully, and A. Zaitsevskii, Theor. Chim. Acta 90, 167 (1995).
38 H.-J. Werner and P. J. Knowles, J. Chem. Phys. 89, 5803 (1988).
39 P. E. M. Siegbahn, Chem. Phys. 25, 197 (1977).
40 H. Zhang, J.-P. Malrieu, P. Reinhardt, and J. Ma, J. Chem. Phys. 132, 034108 (2010).
41 K. G. Dyall, J. Chem. Phys. 102, 4909 (1995).
42 B. Kirtman, J. Chem. Phys. 49, 3890 (1968).
43 B. Kirtman, J. Chem. Phys. 75, 798 (1981).
44 I. Shavitt and L. R. Redmon, J. Chem. Phys. 73, 5711 (1980).
45 a) A. A. Granovsky; URL: http://classic.chem.msu.su/gran/gamess/qdpt2.pdf, b) A. A. Granovsky; URL: http://classic.chem.msu.su/gran/gamess/table_qdpt2.pdf
46 A. A. Granovsky, J. Chem. Phys. 134, 214113 (2011).
47 A. Zaitsevskii and J.-P. Malrieu, Chem. Phys. Lett. 223, 597 (1995)
48 T. H. Schucan, H. Weidenmüller, Ann Phys NY, 73, 108 (1972).
49 T. H. Schucan, H. Weidenmüller, Ann Phys NY, 76, 483 (1973).
50 C. Camacho, R. Cimiraglia and H. A. Witek, Phys. Chem. Chem. Phys., 12, 5058 (2010).
51 C. Camacho, H. A. Witek, and R. Cimiraglia J. Chem. Phys. 132, 244306 (2010).
52 H. A. Witek, Y.-K. Choe, J. P. Finley, K. Hirao, J. Comput. Chem. 23, 957-965 (2002).
53 M. Lewin, Arch. Ration. Mech. Anal 171, 83 (2004)
54 É. Cancès, H. Galicher, and M. Lewin, J. Comp. Phys. 212, 73 (2006).
55 M. Lewin, J. Math. Chem. 44, 967 (2008).
56 J. A. Pople, J. S. Binkley, R. Seeger, Int. J. Quantum Chem. Symp. 10, 1 (1976).
57 K. A. Brueckner, Phys. Rev. 100, 36 (1955).
58 A. Hugenholtz, Physica 23, 481 (1957).
59 R. J. Bartlett, Ann. Rev. Phys. Chem. 32, 359 (1981).
60 M. Hanrath, Chem. Phys. 356, 31 (2009).
61 P. Ghosh, S. Chattopadhyay, D. Jana and D. Mukherjee, Int. J. Mol. Sci. 3, 733 (2002)
62 A. V. Nemukhin, A. V. Bochenkova, K. B. Bravaya, and A. A. Granovsky, Proc. SPIE, 6449, 64490N (2007).
63 K. Bravaya, A. Bochenkova, A. Granovsky, and A. Nemukhin, J. Am. Chem. Soc., 129 (43), 13035 (2007)
64 L. Kessel, I. B. Nielsen, A. V. Bochenkova, K. B. Bravaya, and L. H. Andersen, J. Phys. Chem. A, 111 (42), 10537 (2007)
65 K. B. Bravaya, A. V. Bochenkova, A. A. Granovsky, A. P. Savitsky, and A. V. Nemukhin, J. Phys. Chem. A, 112 (37), 8804 (2008).
66 K. Bravaya, A. Bochenkova, A. Granovsky, A. Nemukhin, Russian Journal of Physical Chemistry B, Focus on Physics, 2, 671 (2008).

67 T. Rocha-Rinza, O. Christiansen, J. Rajput, A. Gopalan, D. B. Rahbek, L. H. Andersen, A. V. Bochenkova, A. A. Granovsky, K. B. Bravaya, A. V. Nemukhin, K. L. Christiansen and M. B. Nielsen, J. Phys. Chem. A, 113 (34), 9442 (2009)
68 M. G. Khrenova, A. V. Bochenkova, and A. V. Nemukhin, Proteins: Structure, Function, and Bioinformatics 78, 614 (2010).
69 A. A. Granovsky; URL: http://classic.chem.msu.su/gran/gamess/xmcqdpt.pdf
70 L. Serrano-Andrés, M. Merchán, and R. Lindh, J. Chem. Phys. 122, 104107 (2005).
71 T. Shiozaki, W. Győrffy, P. Celani and H.-J. Werner, J. Chem. Phys. 135, 081106 (2011)

Geometry optimizations

Introduction

Stationary points are places on the potential energy surface with a zero gradient vector (first derivative of the energy with respect to nuclear coordinates). These include minima (whether local or global), better known to chemists as reactants, products, and intermediates; as well as transition states (also known as first order saddle points).

The two types of stationary points have a precise mathematical definition, depending on the curvature of the potential energy surface at these points. If all of the eigenvalues of the Hessian matrix (second derivative of the energy with respect to nuclear coordinates) are positive, the stationary point is a minimum. If there is one, and only one, negative curvature, the stationary point is a first order saddle point (i.e., a transition state). Saddle points with more than one negative curvature (i.e. higher order saddle points) are not uncommon but are less important in chemistry. Because vibrational frequencies are basically the square roots of the curvatures expressed in the mass-weighted coordinates, a minimum has all real frequencies, and a first-order saddle point has one imaginary vibrational "frequency".

Firefly locates minima by geometry optimization, as RUNTYP=OPTIMIZE, and transition states by first order saddle point searches, as RUNTYP=SADPOINT. The input to control both these RUNTYPs is found in the $STATPT group. In addition, the minimum energy difference geometries corresponding conical intersections/interstate crossings/avoided crossings can be identified with RUNTYP=OPTIMIZE. In this chapter, the term "geometry search" is used here to describe features which are common to all these procedures.

As will be noted in the section on optimizations and symmetry later in this chapter, an OPTIMIZE run does not always find a minimum, and a SADPOINT run may not find a transition state, even though the gradient is brought to zero. You can prove you have located a minimum or transition state only by examining the local curvatures of the potential energy surface. This can be done by following the geometry search with a RUNTYP=HESSIAN job, which should be a matter of routine. Note that CI and ISC location runs are exceptions, i.e. RUNTYP=HESSIAN does not have much sense for geometries located in these runs (as they do not locate a minimum or transition state geometry).

Geometry searches do not bring the gradient exactly to zero. Instead they stop when the largest component of the gradient is below the value of OPTTOL (which defaults to a reasonable 0.0001) and when the RMS of components of gradient vector is below the value of OPTTOL/3. Analytic Hessians usually have residual frequencies below 10 cm**-1 with this degree of optimization. The sloppiest value you probably ever want to try is 0.0005.

If a geometry search runs out of time, or exceeds NSTEP, it can be restarted. For RUNTYP=OPTIMIZE or CONIC, restart with the coordinates having the lowest total energy (for SCF methods, do a string search on "FINAL"). For RUNTYP=SADPOINT, restart with the coordinates having the smallest gradient (do a string search on "RMS", which means root mean square). These are not necessarily at the last geometry! The "restart" should actually be a normal run, that is, you should not try to use the restart options in $CONTRL.

Another method to restart a geometry optimization is to add the entire $RSTART group (and only this group) to the unmodified initial input file and to set $STATPT IREST=2 $END to request a restart. The $RSTART group was generated if the

$STATPT IDUMP=1 (to dump a single file with restart info), or

$STATPT IDUMP=2 (to dump also a backup copy of the restart info)

option was provided in the input file. This option is useful for restarting large jobs running on clusters with queue systems controlling the job's execution time. A job restarted using this option will be resumed exactly as if it were not interrupted. Some experimentation with these options is recommended to get the idea on how this form of restart works.

Sometimes when you are fairly close to the minimum, an OPTIMIZE run will take a first step which raises the energy, with subsequent steps improving the energy and perhaps finding the minimum. The erratic first step is caused by the GUESS Hessian. It may help to limit the size of this wrong first step, by reducing its radius, DXMAX. Conversely, if you are far from the minimum, sometimes you can decrease the number of steps by increasing DXMAX.

When using Z-Matrix or natural internals, the program uses an iterative process to convert the internal coordinate change into Cartesian space. In some cases, a small change in the internals will produce a large change in Cartesians, and thus produce a warning message on the output. If these warnings appear only in the beginning, there is probably no problem, but if they persist you can probably devise a better set of coordinates. You may in fact have one of the two problems described in the next paragraph. In some cases (hopefully very few) the iterations to find the Cartesian displacement may not converge, producing a second kind of warning message. The fix for this may very well be a new set of internal coordinates as well, or adjustment of ITBMAT in $STATPT. Note that this should not be an issue with DLCs as Firefly's GDIIS and CONIC codes are able to redefine DLCs.

There are two examples of poorly behaved Z-Matrix or natural internals coordinates which can give serious problems. The first of these is three angles around a central atom, when this atom becomes planar (sum of the angles nears 360). The other is a dihedral where three of the atoms are nearly linear, causing the dihedral to flip between 0 and 180. Avoid these two situations if you want your geometry search to be convergent.

Sometimes it is handy to constrain the geometry search by freezing one or more coordinates, via the IFREEZ array. For example, constrained optimizations may be useful while trying to determine what area of a potential en-

ergy surface contains a first order saddle point. If you try to freeze coordinates with an automatically generated $ZMAT, you need to know that the order of the coordinates defined in $DATA is

```
y
y  x r1
y  x r2  x a3
y  x r4  x a5  x w6
y  x r7  x a8  x w9
```

and so on, where y and x are whatever atoms and molecular connectivity you happen to be using. Note that this automatically generated ZMAT (which is generated for a non-zero value of NZVAR in the absence of the $ZMAT input group) is identical to the user's COORD=ZMAT or COORD=ZMTMPC input.

Optimizations towards a minimum

If the exact molecular Hessian were to be known at each step, geometry searches would be most effectively done by the Newton-Raphson method. However, since this is not the case, so-called quasi-Newton-Raphson methods are typically superior. These methods assume a quadratic potential surface, and require the exact gradient vector and an approximation to the Hessian. It is the approximate nature of the Hessian that makes the method "quasi". The rate of convergence of the geometry search depends on how quadratic the real surface is, and the quality of the Hessian. The latter is something you have control over, and is discussed in the next section.

Firefly contains various implementations of quasi Newton procedures for finding stationary points, namely METHOD=NR, RFO, QA, GDIIS, and the seldom used SCHLEGEL. They differ primarily in how the step size and direction are controlled, and how the Hessian is updated. The CONOPT method is a way of forcing a geometry away from a minimum towards a TS. It is not a quasi-Newton method, and is described at the very end of this section.

The NR method employs a straight Newton-Raphson step. There is no step size control, the algorithm will simply try to locate the nearest stationary point, which may be a minimum, a TS, or any higher order saddle point. NR is not intended for general use, but is used by Firefly in connection with some of the other methods after they have homed in on a stationary point, and by Gradient Extremal runs where location of higher order saddle points is common. NR requires a very good estimate of the geometry in order to converge to the desired stationary point.

The RFO and QA methods are two different versions of the so-called augmented Hessian techniques. They both employ Hessian shift parameter(s) in order to control the step length and direction.

In the RFO method, the shift parameter is determined by approximating the PES with a Rational Function, instead of a second-order Taylor expansion. For a RUNTYP=SADPOINT, the TS direction is treated separately, giving two shift parameters. This is known as a Partitioned Rational Function Optimization (P-RFO). The shift parameter(s) ensure that the augmented Hessian has the correct eigenvalue structure, all positive for a minimum search,

and one negative eigenvalue for a TS search. The (P)-RFO step can have any length, but if it exceeds DXMAX, the step is simply scaled down.

In the QA (Quadratic Approximation) method, the shift parameter is determined by the requirement that the step size should equal DXMAX. There is only one shift parameter for both minima and TS searches. Again the augmented Hessian will have the correct structure. There is another way of describing the same algorithm, namely as a minimization on the "image" potential. The latter is known as TRIM (Trust Radius Image Minimization). The working equation is identical in these two methods. When the QA steplength is close to DXMAX, there is little difference between the QA and RFO methods. However, the RFO step may in some cases exceed DXMAX significantly, and a simple scaling of the step will usually not produce the best direction. For this reason, QA is generally preferred over RFO.

GDIIS (geometry optimization using direct inversion in the iterative subspace) is an extrapolation method analogous to the DIIS method used for the wavefunction optimization. The implementation used in Firefly is unique and uses ideas presented in references 5 and 6. GDIIS is usually superior to RFO and QA, especially on flat regions on the PES. For this reason, it is the default optimizer.

Near a stationary point the straight NR algorithm is the most efficient. RFO, QA and GDIIS may be viewed as methods for guiding the search in the "correct" direction when starting far from the stationary point. Once the stationary point is approached, the RFO, QA, and GDIIS methods switch to NR, automatically, when the NR steplength drops below 0.10 or DXMAX, whichever is the smallest.

Some references can be given if one desires to understand how these methods are designed to work. The first 3 references describe the RFO and TRIM/QA algorithms. Reference 4, written by Frank Jensen of Odense University (who wrote the original FORTRAN code for the geometry searches) compares many of the algorithms implemented in Firefly. References 5 and 6, as said, provide details regarding the implementation of GDIIS in Firefly. Finally, as a general review and textbook on optimization methods we highly recommend reference 7.

1. J. Baker J. Comput. Chem. 7, 385-395 (1986)
2. T. Helgaker Chem. Phys. Lett. 182, 305-310 (1991)
3. P. Culot, G. Dive, V. H. Nguyen, J. M. Ghuysen Theoret. Chim. Acta 82, 189-205 (1992)
4. F. Jensen J. Chem. Phys. 102, 6706-6718 (1995).
5. S. Vogel, T. H. Fischer, J. Hutter, H. P. Luthi Int. J. Quantum Chem. 45, 6, 679-688 (1993)
6. H. Sellers Int. J. Quantum Chem. 45, 1, pages 31-41 (1993)
7. Practical Optimization, Philip E. Gill, Walter Murray, Margaret H. Wright Academic Press, January 28, 1982

The Hessian

Although quasi-Newton methods require only an approximation to the true Hessian, the choice of this matrix has a great effect on convergence of the geometry search.

There is a procedure contained within Firefly for guessing a diagonal, positive definite Hessian matrix, HESS=GUESS. If you are using Cartesian coordinates, the guess Hessian is 1/3 times the unit matrix. The guess is more sophisticated when internal coordinates are defined, as empirical rules will be used to estimate stretching and bending force constants. Other force constants are set to 1/4. The diagonal guess often works well for minima, but cannot possibly find transition states (because it is positive definite). Therefore, GUESS may not be selected for SADPOINT runs.

Two options for providing a more accurate Hessian are HESS=READ and CALC. For the latter, the true Hessian is obtained by direct calculation at the initial geometry, and then the geometry search begins, all in one run. The READ option allows you to feed in the Hessian in a $HESS group, as obtained by a RUNTYP=HESSIAN job. The second procedure is actually preferable, as you get a chance to inspect the calculated frequencies. Then, if the local curvatures look good, you can commit to the geometry search. Be sure to include a $GRAD group (if the exact gradient is available) in the HESS=READ job so that Firefly can take its first step immediately.

Note also that you can compute the Hessian at a lower basis set and/or wavefunction level, and read it into a higher level geometry search. In fact, the $HESS group could be obtained at the semiempirical level or the RHF/3-21G level (for which the Hessian can be computed analytically). For systems not well described by a single-determinant approach, one can try to use GVB-PP(1) (for which analytical Hessians are also available). This trick works because the Hessian is 3Nx3N for N atoms, no matter what atomic basis is used. The gradient from the lower level is of course worthless, as the geometry search must work with the exact gradient of the wavefunction and basis set in current use. Discard the $GRAD group from the lower level calculation!

You often get what you pay for. HESS=GUESS is free, but may lead to significantly more steps in the geometry search. The other two options are more expensive at the beginning, but may pay back by rapid convergence to the stationary point.

The Hessian update schemes usually work very well but may break down under some (rare) circumstances. The symptoms are a nice lowering of the energy or the RMS gradient for many steps, followed by the inefficient steps and warnings. You can try to cure this by putting the best coordinates into $DATA, and resubmitting, to make a fresh determination of the Hessian and internal coordinates.

The default Hessian update for OPTIMIZE runs is that of Broyden–Fletcher–Goldfarb–Shanno (BFGS), which is likely to remain positive definite. The POWELL update is the default for SADPOINT runs, since the Hessian can develop a negative curvature as the search progresses. The POWELL update is also used by the METHOD=NR and CONOPT since the Hessian may have any number of negative eigenvalues in these cases. The MSP update is a mixture of Murtagh-Sargent and Powell, suggested by Josep Bofill, (J.Comput.Chem., 15, 1-

11, 1994). It sometimes works slightly better than Powell, so you may want to try it.

Optimizations towards a transition state

Finding minima is relatively easy. There are large tables of bond lengths and angles, so guessing starting geometries is pretty straightforward. Very nasty cases may require computation of an exact Hessian, but the location of most minima is straightforward.

In contrast, finding transition states is a black art. The diagonal guess Hessian will never work (as there will be no negative curvature(s) in it), so you must provide a computed one. The Hessian should be computed at your best guess as to what the geometry of the transition state (T.S.) should be, and preferably using a level at least close to the one at which you will perform the search. It is safer to do this in two steps as outlined above, rather than through using HESS=CALC. This lets you verify you have guessed a structure with one and only one negative curvature. Guessing a good trial structure is the hardest part of a RUNTYP=SADPOINT.

This point is worth iterating. Even with sophisticated step size control such as is offered by the quasi Newton methods, it is in general very difficult to move correctly from a region with incorrect curvatures towards a first order saddle point. Even procedures such as CONOPT or RUNTYP=GRADEXTR will not replace your own chemical intuition about where transition states may be located.

The RUNTYP=HESSIAN's normal coordinate analysis is rigorously valid only at stationary points on the surface. This means the frequencies from the Hessian at your trial geometry are untrustworthy, in particular the six "zero" frequencies corresponding to translational and rotational (T&R) degrees of freedom will usually be 300-500 cm**-1, and possibly imaginary. The Sayvetz conditions on the printout will help you distinguish the T&R "contaminants" from the real vibrational modes. If you have defined a $ZMAT, the PURIFY option within $STATPT will help zap out these T&R contaminants).

If the Hessian at your assumed geometry does not have one and only one imaginary frequency (taking into account that the "zero" frequencies can sometimes be 300i!), then it could prove difficult to find the transition state (though one can sometimes get away with multiple negative frequencies if the magnitude of the 'desired' frequency is much larger than that of the other negative frequencies). If so, you need to compute a Hessian at a better guess for the initial geometry, or read about mode following below. Often, a relaxed geometry scans (RUNTYP=RSURFACE) can be used to get a good starting geometry for subsequent TS search.

If you need to restart your run, do so with the coordinates which have the smallest RMS gradient. Note that the energy does not necessarily have to decrease in a SADPOINT run, in contrast to an OPTIMIZE run. It is sometimes necessary to do several restarts, involving recalculation of the Hessian, before actually locating the transition state.

Assuming you do find the T.S., it is always a good idea to recompute the Hessian at this structure. As described in the discussion of symmetry, only totally symmetric vibrational modes are probed in a geometry search. Therefore, it is possible to find that at your "T.S." there is a second imaginary frequency, which corresponds to a non-totally symmetric vibration. This means you haven't found the correct T.S., and are back to the drawing board. The proper procedure is to lower the point group symmetry by distorting along the symmetry breaking "extra" imaginary mode, by a reasonable amount. Don't be overly timid in the amount of distortion, or the next run will come back to the invalid structure.

The real trick here is to find a good guess for the transition structure. The closer you are, the better. It is often difficult to guess these structures. One way around this is to compute a linear least motion (LLM) path. This connects the reactant structure to the product structure by linearly varying each coordinate. If you generate about ten structures intermediate to reactants and products, and compute the energy at each point, you will in general find that the energy first goes up, and then down. The maximum energy structure is a "good" guess for the true T.S. structure. Actually, the success of this method depends on how curved the reaction path is.

A particularly good paper on the symmetry which a transition state (and reaction path) can possess is by

P. Pechukas, J. Chem. Phys. 64, 1516-1521 (1976)

Optimizations and symmetry

At the end of any successful geometry search, you will have a set of coordinates where the gradient of the energy is zero. However, your newly discovered stationary point is not necessarily a minimum or a transition state.

This apparent mystery is due to the fact that the gradient vector transforms under the totally symmetric representation of the molecular point group. As a direct consequence, a geometry search is point group conserving. (For a proof of these statements, see J. W. McIver and A. Komornicki, Chem. Phys. Lett., 10, 303-306 (1971)). In simpler terms, the molecule will remain in whatever point group you select in $DATA, even if the true minimum is in some lower point group. Since a geometry search only explores totally symmetric degrees of freedom, the only way to learn about the curvatures for all degrees of freedom is RUNTYP=HESSIAN.

As an example, consider disilene, the silicon analog of ethene. It is natural to assume that this molecule is planar like ethene, and an OPTIMIZE run in D2h symmetry will readily locate a stationary point. However, as a calculation of the Hessian will readily show, this structure is a transition state (one imaginary frequency), and the molecule is really trans-bent (C2h). A careful worker will always characterize a stationary point as either a minimum, a transition state, or some higher order stationary point (which are usually not of any interest) by performing a RUNTYP=HESSIAN.

The point group conserving properties of a geometry search can be annoying, as in the preceding example, or advantageous. For example, assume you wish to locate the transition state for rotation about the double bond in ethene. A little thought will soon reveal that ethene is D2h, the 90 degrees twisted structure is D2d, and structures in between are D2. Since the transition state geometry is actually higher symmetry than the rest of the rotational surface, you can locate it by RUNTYP=OPTIMIZE within D2d symmetry. You can readily find this stationary point with the diagonal guess Hessian. In fact, if you attempt to do a RUNTYP=SADPOINT within D2d symmetry, there will be no totally symmetric modes with negative curvatures, and it is unlikely that the geometry search will be very well behaved.

Although a geometry search cannot lower the symmetry, the gain of symmetry is quite possible. For example, if you initiate a water molecule optimization with a trial structure which has unequal bond lengths, the geometry search will come to a structure that is indeed C2v (to within OPTTOL, anyway). However, Firefly leaves it up to you to realize that a gain of symmetry has occurred.

In general, Mother Nature usually chooses more symmetrical structures over less symmetrical structures. Therefore, you are probably better served to assume the higher symmetry, perform the geometry search, and then check the stationary point's curvatures. The alternative is to start with artificially lower symmetry and see if your system regains higher symmetry. The problem with this approach is that you don't necessarily know which subgroup is appropriate, and you lose the great speedups Firefly can obtain from proper use of symmetry. It is good to note here that "lower symmetry" does not mean simply changing the name of the point group and entering more atoms in $DATA, instead the nuclear coordinates themselves must actually be of lower symmetry. Otherwise, the molecular symmetry will be preserved even in the C1 group and will be destroyed only due to round-off errors accumulation. Sometimes this process is so slow so that the geometry converges faster than the symmetry is destroyed in C1 or any other subgroup.

Mode Following

In certain circumstances, METHOD=RFO, QA, or GDIIS can walk from a region of all positive curvatures (i.e. near a minimum) to a transition state. The criteria for whether this will work is that the mode being followed should be only weakly coupled to other close-lying Hessian modes. Especially, the coupling to lower modes should be almost zero. In practice this means that the mode being followed should be the lowest of a given symmetry, or spatially far away from lower modes (for example, rotation of methyl groups at different ends of the molecule). It is certainly possible to follow also modes which do not obey these criteria, but the resulting walk (and possibly TS location) will be extremely sensitive to small details such as the stepsize.

This sensitivity also explain why TS searches often fail, even when starting in a region where the Hessian has the required one negative eigenvalue. If the TS mode is strongly coupled to other modes, the direction of the mode is incorrect, and the maximization of the energy along that direction is not really what you want (but what you get).

Mode following is really not a substitute for the ability to intuit regions of the PES with a single local negative curvature. When you start near a minimum, it matters a great deal which side of the minima you start from, as the direction of the search depends on the sign of the gradient. We strongly urge that you read before trying to use IFOLOW, namely the papers by Frank Jensen and Jon Baker mentioned above, and see also Figure 3 of C. J. Tsai, K. D. Jordan, J. Phys. Chem. 97, 11227-11237 (1993) which is quite illuminating on the sensitivity of mode following to the initial geometry point.

Note that Firefly retains all degrees of freedom in its Hessian, and thus there is no reason to suppose the lowest mode is totally symmetric. Remember to lower the symmetry in the input deck if you want to follow non-symmetric modes. You can get a printout of the modes in internal coordinate space by a EXETYP=CHECK run, which will help you decide on the value of IFOLOW.

Constrained optimization

CONOPT is a different sort of transition state search procedure. Here, a certain "CONstrained OPTimization" may be considered as another mode following method. The idea is to start from a minimum, and then perform a series of optimizations on hyperspheres of increasingly larger radii. The initial step is taken along one of the Hessian modes, chosen by IFOLOW, and the geometry is optimized subject to the constraint that the distance to the minimum is constant. The convergence criteria for the gradient norm perpendicular to the constraint is taken as 10*OPTTOL, and the corresponding steplength as 100*OPTTOL.

After such a hypersphere optimization has converged, a step is taken along the line connecting the two previous optimized points to get an estimate of the next hypersphere geometry. The stepsize is DXMAX, and the radius of hyperspheres is thus increased by an amount close (but not equal) to DXMAX. Once the pure NR step size falls below DXMAX/2 or 0.10 (whichever is the largest) the algorithm switches to a straight NR iterate to (hopefully) converge on the stationary point.

The current implementation always conducts the search in Cartesian coordinates, but internal coordinates may be printed by the usual specification of NZVAR and ZMAT. At present there is no restart option programmed.

CONOPT is based on the following papers, but the actual implementation is the modified equations presented in Frank Jensen's paper mentioned above.

Y. Abashkin, N. Russo, J. Chem. Phys. 100, 4477-4483 (1994)
Y. Abashkin, N. Russo, M. Toscano, Int. J. Quant. Chem. 52, 695-704(1994)

There is little experience on how this method works in practice, so experiment with it at your own risk!

Locating Conical Intersections (CIs) and Interstate Crossings (ISCs)

Conical Intersections (CIs) and Interstate Crossings (ISCs) are the specific manifolds on the Potential Energy Surfaces (PESs) of two different electronic states where, at the same geometry, the energies of these states are strictly equal each other and thus two states are exactly degenerate. The difference between CIs and ISCs is that ISCs are always formed by the states of different spatial or spin symmetry. As a result these states belong to the different blocks of the Hamiltonian matrix expressed in a block-diagonal form using symmetry-adapted (i.e. adapted according to the different possible irreducible representations) basis set. Let us consider small perturbations of the Hamiltonian matrix caused by the various deformations of the selected molecular geometry that preserve the initial molecular symmetry and hence the symmetry of the Hamiltonian. Evidently, there is no any "interaction" possible between the two states of interest, even exactly at their crossing points, as any "interaction" which could be caused by such perturbations is vanished due to symmetry reasons. As the result, the dimension of the ISC manifold is equal to the dimension of the original PES minus one, as there is in fact only **one** constrain (i.e. $E_i = E_j$) to be satisfied.

By contract, CIs are formed by the states of exactly the same symmetry so that they are allowed to "interact". Mathematically, a second constrain arises: "interactions" of that kind must vanish at the CI manifold, otherwise the states cannot be exactly degenerate. The dimension of CI manifold is thus equal to the dimension of the original PES minus **two**, and the local topology on a two-dimension slice over the crossing manifold is the same as for two crossing cones with a single point in common, hence the "Conical Intersection" name. One could argue there is still only one constrain, namely: $E_i = E_j$ at any CI point. While this is formally true, for CIs the equality $E_i = E_j$ is in fact of the type: $x^+*x + y^+*y = 0$ (where the superscript '+' sign stands for conjugate). The latter, being a single equality, implies both $x = 0$ and $y = 0$!

Usually, there is no need to locate all the geometries belonging to ISC or CI manifolds. Often, one is interested in locating just the Minimum Energy Crossing Point (MECP or MECI). These are specific ISC or CI points having the minimum possible energy. Unlike CIs and ISCs, MECPs and MECIs are not multidimensional manifolds of complex structure but rather are just a single point on PES. MECPs and MECIs are typically exactly what the most people calls ISCs and CIs.

With Firefly versions prior to Firefly v. 8.0.0, it was possible to locate MECIs and MECPs only at the state-averaged MCSCF level of theory, using an efficient procedure developed for the calculation of semi-numerical state-specific gradients for SA-MCSCF (which is documented in the chapter on MCSCF). In addition, only two intersecting states were allowed to be averaged, naturally with equal weights, in the SA-MCSCF procedure. With Firefly v. 8.0.0, the MECIs/MECPs location code was extended to optionally use numerical gradients thus allowing XMCQDPT2, MCQDPT2, and any configuration interaction procedure to be used for the location of MECIs and MECPs in addition to plain SA-MCSCF. Moreover, an arbitrary averaging of states can now be used for the purpose of locating MECIs and MECPs at the SA-MCSCF level, provided the weights of two crossing states are equal. However, the

energies of both intersecting states must be obtained as a result of a single computation (e.g. a single SA-MCSCF or XMCQDPT2 procedure). The primary reason of this is to avoid an imbalanced description of two different states of interest which could arise from the use of different computational schemes for different states.

There are two very different ways how the MECIs/MECPs can be located with Firefly. The first one is based on the use of penalty functions, the second one is based on the use of Lagrange multiplier technique. Any of them can be used with both Cartesian and internal coordinates (including DLCs), and can handle additional geometry constrains like frozen coordinates, etc.

The penalty function based methods programmed in Firefly are based on the minimization of the following expression:

$$F(E_i, E_j) = \text{alpha}*E_i + (1-\text{alpha})*E_j + \text{Penalty}(E_j-E_i) = \min \qquad (1)$$

rather than of the E_i itself as it would be in the case of unconstrained energy optimization. With Firefly, the possible values of alpha are 1.0 and 0.5. This corresponds to the minimization of the energy of the first state of interest or the average energy of two states, respectively. Evidently, at a MECI/MECP point these are exactly the same, but they differ when away from the MECI/MECP geometry. $\text{Penalty}(E_j-E_i) = \text{Penalty}(\Delta E)$ is the penalty function which depends on the energy splitting between two states.

The well designed penalty function must obey some restrictions. It should be zero if ΔE is zero and should be (very) small if ΔE is small enough. At the same time, the penalty function and its gradient should become (very) large when ΔE increases. In addition, it is convenient if $\text{Penalty}(\Delta E)$ is a smooth and differentiable function as all standard geometry optimization engines assume that the PES is a smooth and differentiable function. By forcing $\text{Penalty}(\Delta E)$ to be smooth enough, it is possible to use any standard geometry optimization code for location of MECPs and MECIs. Strictly speaking, the minimum of $F(E_i, E_j)$ does not exactly correspond to a MECI or MECP point and E_i and E_j are not exactly equal at the corresponding geometry. However, for a well-designed penalty function this point is very close to true MECI/MECP.

There are three different penalty functions presently available in Firefly. The choice of the penalty function is controlled by the PENLTY variable of the $MCAVER input group. Each of three penalty functions is parameterized by two parameters which can be altered by the user.

PENLTY=1 calls the so-called Ciminelli penalty function and has the form:

$$\text{Penalty}(\Delta) = B*\ln(1.0 + (\Delta/A)^2) \quad \text{with default values A=0.008 and B=0.2}$$

PENLTY=2 calls a penalty function suggested by Levine, Coe, and Martinez:

$$\text{Penalty}(\Delta) = B*(\Delta^2)/(\Delta+A) \quad \text{with default values A=0.02 and B=3.5}$$

Finally, PENLTY=3 calls a penalty function specific to Firefly:

$$\text{Penalty}(\Delta) = B*((\Delta^2 + A^2)^{1/2}) - B*|A|$$ with default values A=0.01 and B=3.5

The values of 'A' and 'B' can controlled with the A and B keywords in the $MCAVER group.

The method based on the Lagrange multiplier technique as implemented in Firefly is the original self-consistent variant of the Sequential Quadratic Programming approach. Basically, the energy of state i (i.e. E_i) is optimized subject to constrain $E_i = E_j$. The constrain is taken into account by constructing the proper Lagrange function:

$$L(R) = E_i(R) + \lambda(R)*(E_j(R) - E_i(R))$$

where λ is the Lagrange multiplier for constrain $E_i = E_j$ and (R) denotes the dependence of all quantities on the molecular geometry. The Lagrange function is then optimized. Its minimum is located exactly at the MECI/MECP geometry while the computed value of λ contains important information on the type of located MECI/MECP. More precisely, if λ is greater than 1.0, gradients of two degenerated states at their MECI/MECP point are always oppositely directed. Otherwise, they are directed in the same direction. Naturally, one can consider another form of L:

$$L'(R) = (E_i(R) + E_j(R))*0.5 + \lambda'(R)*(E_j(R) - E_i(R))$$

Optimization of both L(R) and L'(R) result in the same MECI/MECP geometry, while the values of two Lagrange multipliers at this geometry differ by 0.5. However, the number of steps required to converge optimization is generally different. The trajectories formed by the sequential points taken by the optimization procedure are different as well.

As to which approach is preferred: the Lagrange multiplier technique usually works better and converges faster. In addition, it provides useful information, i.e. the value of multiplier itself. On the other hand, penalty function based methods are a bit more robust to numerical noise. Our experience shows that, with Firefly, Lagrange multiplier based approaches are the best choice for SA-MCSCF while the penalty function based methods work better for XMCQDPT2.

With Firefly, both penalty function and Lagrange multiplier based code require only state specific energy gradients for both MECI and MECP location. None of the approaches requires computation of the so-called non-adiabatic coupling vector. This vector corresponds to the second direction (the first being the difference of gradients of two states) along which the CI manifold (i.e. the manifold of the exact degeneracy) is destroyed under the small variations of molecular geometry. In other words, this vector describes the second constrain which is specific to CIs but not to ISCs. Taken together with the gradients difference the non-adiabatic coupling vector forms the so-called "branching plane".

Because the non-adiabatic coupling vector is not used nor computed by Firefly's routines, both MECIs and MECPs are handled by the same code and are virtually identical from the point of view of MECI/MECP geometry optimizer. In Firefly, the MECI or MECP location is not implemented as a separate RUNTYP. The standard $CONTRL RUNTYP=OPTIMIZE is used instead together with some additional keywords controlling the process of MECI/MECP optimization. In addition to RUNTYP=OPTIMIZE, the RUNTYP=RSURFACE is also allowed. While the penalty function based approach can be used with any of the available Firefly's geometry optimization engines such as GDIIS, QA, or NR, the Lagrange multiplier-based approach requires the dedicated geometry optimizer which is called "CONIC" and should be explicitly requested as follows: $STATPT METHOD=CONIC. The location of MECIs/MECPs can be performed in Cartesian or internal coordinates, the latter being the recommended coordinate type.

The $MCAVER input group is relevant for both penalty function and Lagrange multiplier based code. In addition, the Lagrange multiplier based approach is also controlled by the dedicated $CONIC input group. If MECI/MECP optimization is based on fully numerical gradients (e.g. the optimization of MECI at XMCQDPT2 level), the ISTATE, JSTATE, and NGRADS keywords of the $NUMGRD input group should be properly set as described in the documentation on the numerical gradient code. Alternatively, if semi-numerical state-specific gradients for SA-MCSCF are used, the ISTATE keyword of the $MCSCF input group must be properly set, as well as other variables that are essential for the computation of state-specific gradients for SA-MCSCF. Finally, when locating CIs/MECPs at SA-MCSCF level of theory, the MCSCF state tracking can be of considerable help. The latter is controlled by the $TRACK input group, which function is described in the MCSCF chapter. More information on the $MCAVER group can be found in the MCSCF chapter as well as in the list of keywords.

The $CONIC group allows control over the detailed behavior of the $STATPT METHOD=CONIC Lagrange multiplier based CIs/ISCs optimizer called CONIC. In addition to the parameters described below, METHOD=CONIC is also controlled by the usual parameters of the $STATPT group. Detailed information on the $CONIC group can be found in the list of keywords.

This section will conclude with three input examples for the optimization of CIs and ISCs. The complete input files as well as the corresponding output files can be found at the following location:

http://classic.chem.msu.su/gran/gamess/Conic_ISC_samples.rar

Input example 1. Optimization of ISC in 1,1-difluoro-1,3-butadiene molecule using Lagrange multiplier based code and semi-numerical gradients for SA-MCSCF.

```
 $CONTRL SCFTYP=MCSCF RUNTYP=OPTIMIZE INTTYP=HONDO
         ICUT=11 D5=.T. NZVAR=1 $END
 $SYSTEM TIMLIM=6000 MWORDS=10 NOJAC=1 KDIAG=0 $END
 $ZMAT DLC=1 AUTO=1 $END
```

```
 $MOORTH NOSTF=1 NOZERO=1 TOLE=0 TOLZ=0 SYMS=1 SYMDEN=1 SYMVEC=1 $END

 $GUESS GUESS=MOREAD NORB=26 $END

 $DRT GROUP=C1 FORS=.T. NMCC=18 NDOC=5 NVAL=3 $END
 $GUGDIA NSTATE=2 ITERMX=200 CVGTOL=1D-7 $END
 $GUGDM2 CUTOFF=1D-12 WSTATE(1)=1,1 $END
 $GUGEM CUTOFF=1D-12 $END
 $MCSCF CISTEP=GUGA MAXIT=350 ISTATE=1 ACURCY=5D-8
        ENGTOL=5.0D-13 NTRACK=2 $END
 $TRANS CUTTRF=1D-12 $END

 $MCAVER JSTATE=2 CONIC=2 $END

 $TRACK TOL=1.4 FREEZE=.T. STICKY=.F. UPDATE=.T. RESET=.F. DELCIV=.F. $END

 $STATPT NSTEP=1000 OPTTOL=1.0D-4 METHOD=CONIC $END

 $DATA
 optimization of ISC in CF2=CH-CH=CH2 by state-averaged CAS
CS
--input coordinates--
 $END
 $VEC
--input orbitals--
 $END
```

Input example 2. Optimization of CI in ethene molecule at XMCQDPT2 level using penalty function based code and numerical gradients.

```
$CONTRL
! Set NZVAR to non-zero to allow automatic DLCs
  NZVAR=1
  SCFTYP=MCSCF MPLEVL=2
! Some standard stuff for XMCQDPT2 numerical derivatives.
  RUNTYP=OPTIMIZE INTTYP=HONDO ICUT=13 NUMDER=1
! We want to use spherical basis set in this particular run
  D5=.T.
 $END

 $SYSTEM
  TIMLIM=6000 MWORDS=40
! To speed up things a bit
  NOJAC=1 KDIAG=0
 $END

! To allow DLCs
 $ZMAT DLC=1 AUTO=1 $END

! This group is required for extreme precision needed for numerical deriva-
tives
 $MOORTH NOSTF=1 NOZERO=1 TOLE=0 TOLZ=0 $END
```

```
 $BASIS GBASIS=N311 NGAUSS=6 NDFUNC=1 POLAR=DUNNING $END

 $GUESS GUESS=MOREAD NORB=9 $END

! We need to increase precision here a bit
 $TRANS ALTPAR=.T. MPTRAN=2 DIRTRF=.T. MODE=100 CUTTRF=1D-13 $END
! We need to increase precision here a bit
 $MCSCF CISTEP=ALDET SOSCF=.T. FOCAS=.F. MAXIT=100 ACURCY=1D-8 $END
 $DET
GROUP=C1 NCORE=7 NACT=2 NELS=2 NSTATE=4 WSTATE(1)=1,1
! We need to increase precision here a bit
ITERMX=1000 CVGTOL=1D-8
 $END

! This input is specific to penalty-based CI search algorithm
! (using penalty function due to work of Todd Martinez)
 $MCAVER TARGET=MIXED PENLTY=2 XGRAD=.F. $END

 $XMCQDPT
NSTATE=2 EDSHFT=0.02 ISTSYM=1 WSTATE(1)=1,1
! We need to increase precision a bit
THRGEN=1D-20 THRERI=1D-20 GENZRO=1D-20 THRCON=1D-8
 $END

! To compute two (NGRADS=2) gradients for ISTATE=1 and JSTATE=2
! using special fitting for CIs (SPLINE=4 CONFIT=1 PRAXES=1) to improve
! precision of numerical gradients near CI point
 $NUMGRD
  ORDER=6 DELTA=0.01 ISTATE=1 JSTATE=2 NGRADS=2 SPLINE=4 CONFIT=1 PRAXES=1
 $END

! Using QA optimizer
 $STATPT NSTEP=400 METHOD=QA OPTTOL=3D-5 FMAXT=1000 $END

 $DATA
 C2H4
C1
--input coordinates--
 $END
 $VEC
--input orbitals--
 $END
```

Input example 3. Optimization of CI in allene molecule using Lagrange function based code and semi-numerical gradients for SA-MCSCF.

```
 $CONTRL SCFTYP=MCSCF RUNTYP=OPTIMIZE INTTYP=HONDO
         ICUT=11 D5=.T. NZVAR=1 $END
 $SYSTEM TIMLIM=6000 MWORDS=10 NOJAC=1 KDIAG=0 $END

 $MOORTH NOSTF=1 NOZERO=1 TOLE=0 TOLZ=0 SYMS=1 SYMDEN=1 SYMVEC=1 $END
```

```
 $GUESS GUESS=MOREAD NORB=13 $END

 $BASIS GBASIS=DH NDFUNC=1 POLAR=DUNNING $END

 $DRT GROUP=CS FORS=.T. NMCC=9 NDOC=2 NVAL=2 ISTSYM=1 $END
 $GUGDIA NSTATE=4 ITERMX=200 CVGTOL=1D-7 MEMMAX=999999 $END
 $GUGEM CUTOFF=1D-12 $END
 $GUGDM2 CUTOFF=1D-12 WSTATE(1)=1,1 $END
 $MCSCF CISTEP=GUGA MAXIT=100 ISTATE=1 ACURCY=1D-7 ENGTOL=1.0D-12 NTRACK=4
$END
 $TRANS CUTTRF=1D-12 $END

 $MCAVER JSTATE=2 CONIC=2 $END

 $CONIC SHIFT0=1D-2 $END

 $TRACK TOL=1.4 FREEZE=.T. STICKY=.F. UPDATE=.T. RESET=.F. DELCIV=.F. $END

 $STATPT NSTEP=1000 OPTTOL=3.0D-5 METHOD=CONIC $END

 $DATA
 C3H4
Cs
--input coordinates--
 $END
 $VEC
--input orbitals--
 $END
```

Using numerical gradients

As of version 8.0.0, Firefly has the ability to calculate gradients numerically. The primary purpose of this feature is to allow computations which normally require analytic energy gradients to be performed using QC methods for which analytic gradients are not yet programmed.

The idea behind the numerical gradients code is very simple; gradients are obtained using various finite difference formulas applied to the energy itself. Thus, the approach is based on multiple reevaluations of energy values at a set of slightly displaced geometries. The numerical gradients code is tightly integrated into Firefly and can be used anywhere where analytical energy gradients are required by default. It operates in both standard parallel, XP, and extended XP modes, allowing parallel computations of gradients for those QC methods for which parallel implementation is not available yet and improving parallel scalability of methods that are already well parallelized. The numerical gradient code handles molecular symmetry in a very efficient way and always uses the minimal required number of single-point energy evaluations. The code supports both static and dynamic load balancing.

The numerical gradient code is enabled by specifying

```
      $CONTRL NUMDER=.T. $END
```

Further control over the numerical gradient code is provided by keywords in the $NUMGRD group. The ORDER keyword determines the amount of energy evaluations that will be performed, which (in the case of C1 symmetry) is

```
X * (3N – 6) + 1
```

where X is the value of ORDER and N is the amount of atoms in $DATA. As already noted, use of molecular symmetry reduces the amount of energy evaluations needed. Possible values for ORDER are 1, 2, 4, and 6.

Then, three other important keywords should be mentioned. The NGRADS keyword can be used to request gradients for multiple states of interest, when using a method that can calculate the energies of several states at once (such as MCSCF). The default is NGRADS=1, meaning that the gradient is only computed for the lowest energy state. NGRADS=3, for example, will calculate the gradients of the three lowest states. Obviously, the amount of states requested with NGRADS should not be larger than the amount of states available.

The keyword ISTATE is used to pick the state to be used by other parts of the program (e.g., the state to be used for a geometry optimization). Its default value is 1, the lowest energy state. ISTATE is intended to be used together with the NGRADS option. For example, when optimizing the geometry of the second root using TDDFT, one should specify NGRADS as greater than or equal to 2, and ISTATE as 2. A second state (to be used for the location of conical intersections or interstate crossings) can be requested with the JSTATE keyword. By default, JSTATE equals 0 meaning that no second state is to be used. Note that, for TD/CIS runs, ISTATE and JSTATE pertain to TD/CIS states, e.g. ISTATE=1 equals the first excited state, not the ground state!

As with any finite differencing, numerical gradients require extra high precision in computed energy values. It is therefore the user's responsibility to increase the precision of all the stages involved into computation of target energies.

Finally, it should be noted that the new RUNTYP=NUMGRAD is a slightly cheaper alternative to RUNTYP=GRADIENT when using numerical gradients. In addition, RUNTYP=NUMGRAD always prints gradients for all NGRADS states while RUNTYP=GRADIENT will only print gradient for the ISTATE state.

Non-gradient total energy minimizations

A non-gradient optimization of exponents or the geometry can be selected by specifying RUNTYP=TRUDGE. The TRUDGE package is a modified version of the same code from Michel Dupuis' HONDO 7.0 system, originally written by H. F. King. The program allows for the optimization of 100 parameters.

Exponent optimization works only for uncontracted primitives, without enforcing any constraints (though symmetrically-equivalent atoms will retain identical exponents).

Geometry optimization works only in HINT internal coordinates. The total energy of all types of wavefunctions can be optimized, although this would make no sense as gradient methods are far more efficient. The main purpose of the program used to be for methods lacking analytical gradients such as open shell MP2 and CI. However, with the availability of numerical gradients in Firefly 8.0.0 the TRUDGE program has much less purpose. It is currently retained for backwards compatibility reasons mainly.

For information on how to run RUNTYP=TRUDGE job, see the list of keywords (specifically, the $TRUDGE and $TRURST groups).

Hessian calculations

General information

The Hessian matrix, *i.e.* the second derivatives of the energy with respect to the input geometry, can be calculated by means of a RUNTYP=HESSIAN job. It can be calculated analytically for RHF and ROHF wavefunctions as well as for GVB wavefunctions with NPAIR=0 or 1. For all other theories, it can only be obtained numerically. Control over the calculation of the Hessian (and data derived from the Hessian) is provided by keywords of the $FORCE group. For analytical calculations, additional control is provided by keywords of the $CPHF group.

Compared to numerical Hessians, analytic Hessians are more accurate and computed much more quickly, but they require additional disk storage as well as more physical memory. The numerical calculation of Hessian requires less storage and memory, but takes more time as it requires at least 3N+1 gradient evaluations (where N is the number of symmetry unique atoms). In addition, the accuracy of numerical Hessians is typically somewhat lower but can be improved by doubling the number of displacements in each Cartesian direction. Such a run requires 6N+1 gradient evaluations and can be requested through the NVIB keyword:

```
 $FORCE NVIB=2 $END
```

Doubling the number of displacements gives a small improvement in accuracy -- frequencies will often differ from NVIB=1 results by 1-100 wavenumbers. However, the normal modes will be more nearly symmetry adapted, and the residual rotational and translational modes will be much closer to zero. A Hessian calculated with NVIB=2 is typically almost as good as an analytically calculate one.

In addition, it is also possible to calculate Hessian for methods for which only energies are available using numerical gradients. As discussed elsewhere, numerical gradients can be requested with:

```
 $CONTRL NUMDER=.T. $END
```

and require tighter accuracy settings than ordinary runs. When calculating Hessians, it is important to set the DELROT keyword of the $NUMGRD group to .FALSE.

A successfully calculated Hessian will be printed both to the output (together with vibrational data, corresponding IR intensities, and a thermochemical analysis) and to the PUNCH file (as a $HESS group). The $HESS group can be used for subsequent runs such as a geometry optimization, an IRC calculation, a Raman calculations, etc. Restart data for a numerical Hessian calculation is written to the IRCDATA file as a $VIB group. If one would like to resume such a calculation, it suffices to copy this group (make sure it is terminated with '$END') to the input and restart the run.

A completed Hessian calculation can be rerun by copying the $HESS group to the input and specifying:

```
 $FORCE RDHESS=.T. $END
```

As is discussed in the chapter on geometry optimizations, a geometry optimization always needs an initial Hessian at the start of the calculation. An initial Hessian can be guessed by:

```
 $STATPT HESS=GUESS $END
```

Unfortunately though, the guessed Hessian is of poor quality which usually causes the geometry converger to require more steps before the equilibrium geometry is reached. Also, for an optimization towards a first-order saddle point, it is important to realize that the guessed Hessian cannot contain negative frequencies, which are required of one would like to have any chance of finding the desired saddle point. As such, it is typically recommended to calculate an initial Hessian prior to the optimization procedure. This can be done at the same level at which the optimization will be performed with:

```
 $STATPT HESS=CALC $END
```

However, it can be almost as effective and much cheaper to calculate the initial Hessian at a lower level of theory. For example, a Hessian calculated at the PM3, RM1, or RHF/3-21G level can already be a good approximation to the exact Hessian and greatly speed up the optimization procedure. For systems which cannot be described well by a single-determinant method, one could try to use GVB with NPAIR=1 (i.e., a GVB-PP(1) or TCSCF calculation) which can provide a starting Hessian readily by means of an analytical calculation. After the calculation of the initial Hessian, the $HESS block should be placed in the input of the geometry optimization run together with:

```
 $STATPT HESS=READ $END
```

When a geometry optimization has converged, the nature of the stationary point found can be investigated by calculating the Hessian at the exact same level of theory that was used for the geometry optimization. This can even be done in the same job by specifying:

```
 $STATPT HSSEND=.T. $END
```

If the Hessian is calculated at a different level, any data derived from the Hessian (such a thermochemistry) is invalid.

Vibrational data

By default, a vibrational analysis is performed at the end of a RUNTYP=HESSIAN or RUNTYP=OPTIMIZE/SADPOINT with HSSEND=.T. run. For each normal mode, the frequency, reduced mass, and IR intensity are printed. IR intensities are automatically calculated as part of the Hessian calculation and are printed also to the PUNCH file (the $DIPDR group).

If a Hessian calculation is rerun by supplying the $HESS group in the input stream, the $DIPDR group should also be supplied, otherwise IR intensities will not be printed.

IR intensities are printed in Debye^2/amu-Å^2. These can be converted into intensities as defined by Wilson, Decius, and Cross's equation 7.9.25 (see *The Theory of Infrared and Raman Vibrational Spectra*), in km/mol by multiplying by 42.255. If you prefer 1/atm-cm^2, use a conversion factor of 171.65 instead. A good reference for deciphering these units is:

A. Komornicki, R. L. Jaffe *J. Chem. Phys.* **1979**, *71*, 2150-2155

As mentioned earlier, a vibrational analysis provides information on the nature of the stationary point found. When the geometry found lies in a minimum on the PES, no imaginary vibrational modes (modes with a negative frequency) may be present. In addition, the six modes that correspond to rotations and translations of the geometry should be close to zero. For a first-order saddle point, only one imaginary mode should be present, and the nature of this mode should correspond to the process to which this saddle point (transition state) belongs. For example, the first-order saddle point on the PES of a bond breaking process will exhibit an imaginary mode in which the bond to be broken will be stretching.

Thermochemistry

A thermochemical analysis is automatically performed at the end of a RUNTYP=HESSIAN or RUNTYP=OPTIMIZE/SADPOINT with HSSEND=.T. run. Thermochemical parameters are printed at the end of the output.

The temperature for which the analysis is performed can be specified with the TEMP keyword in the $FORCE group. The temperature should be given as an array. Up to ten temperatures can be specified at the same time. For example:

```
$FORCE TEMP(1)=298.15, 320.15, 350.15 $END
```

A temperature of 0 K should be specified as 0.001 K. The pressure used for the analysis is 1 atmosphere (101325 Pa) and cannot be modified.

It is possible to specify a scaling factor for the thermochemical contribution to the zero-point energy through the SCLFAC keyword. Scaling factors are typically < 1.0 and have been proposed for various combinations of theories and basis sets. Scaling is only used for the thermochemical analysis and as such does not affect the printed wavenumbers.

It should be noted that Firefly will always attempt to identify rotations and translations automatically, but will print a warning if these modes are not the first six in the list (excluding the single imaginary mode present in the case of a first-order saddle point):

```
  WARNING, MODE 1 HAS BEEN CHOSEN AS A VIBRATION
  WHILE MODE 7 IS ASSUMED TO BE A TRANSLATION/ROTATION.
```

```
  PLEASE VERIFY THE PROGRAM'S DECISION MANUALLY!
  MODES 2 TO 7 ARE TAKEN AS ROTATIONS AND TRANSLATIONS
```

In the case of such a warning, it is recommended to visualize the modes using a third-party visualization tool and check if Firefly has chosen the rotations and translation correctly. An incorrect assignment of vibrations, rotations, and translations will result in an error in the thermochemical data. In many cases, the warning originates from small inaccuracies in the calculated Hessian or from the geometry not yet being fully in a minimum.

Raman activities

After the Hessian has been calculated, it is possible to calculate Raman activities by means of a RUNTYP=RAMAN run. Such a run requires the calculated Hessian to be present in the input (in the form of a $HESS group), together with the geometry for which it was calculated. The calculation requires 18+1 gradient evaluations in total and can be controlled by keywords of the $RAMAN group. The method of calculation is described in:

a) A. Komornicki, J. W. McIver *J. Chem. Phys.* **1979**, *70*, 2014-2016
b) G. B. Bacskay, S. Saebo, P. R. Taylor *Chem. Phys.* **1984**, *90*, 215-224

Dipole derivatives (i.e., IR intensities) are recalculated during a Raman calculation and, as such, the $DIPDR group does not have to be provided in order for IR intensities to be printed.

Calculated Raman data is printed at the end of the output as part of the vibrational data, whereby the Raman activity and depolarization are given for each vibrational mode. In addition, the calculated polarizability is given in the PUNCH file as an $ALPDR group. A completed Raman calculation can be rerun instantly by providing the $DIRDR and $ALPDR groups in the input, which can be of use in the case of isotopic substitution. In addition, the $ALPDR group can also be specified for a RUNTYP=HESSIAN run if one desires to make use of options of the $FORCE group – in this case, Raman activities will be added to the output of this run. Restart data for incomplete RUNTYP=RAMAN runs is written to the IRCDATA file as a $VIB group.

Raman activities are printed $Å^4$/amu. These can be multiplied by 6.0220E-09 in order to obtain units of cm^4/g. For the relationship between Raman activities and intensities, see:

a) P. L. Polavarapu *J. Phys. Chem.* **1990**, *94*, 8106-8112
b) G. Keresztury, S. Holly, J. Varga, G. Besenyei, A. Y. Wang, J. R. Durig *Spectrochim. Acta A* **1993**, *49A*, 2007-2026

As was discussed earlier, the calculation of the Hessian requires one to use the exact level of theory that was also used for the geometry optimization, otherwise the calculated vibrational data will not be valid. However, it is important to note that such a requirement does not exist for the calculation of Raman data. Thus, it is possible to obtain the geometry and Hessian using a level of theory that is more suited for obtaining accurate

geometries and wavenumbers, while a level of theory more suited for the calculation of polarizabilities can be used during the Raman calculation.

It is not possible to calculate other Raman-related properties, such as pre-resonance Raman activities, resonance Raman activities, and Raman optical activities (i.e., ROA spectra).

An example input file for a methane molecule is as follows:

```
 $CONTRL SCFTYP=RHF RUNTYP=RAMAN UNITS=ANGS $END
 $SYSTEM TIMLIM=100000 MWORDS=10 $END

 $BASIS GBASIS=N31 NGAUSS=6 $END

 $DATA
 Methane Raman calculation
Td

 C   6.0   0.0000000   0.0000000   0.0000000
 H   1.0   0.6247991   0.6247991   0.6247991
 $END

 $HESS
--Hessian data--
 $END
```

Potential energy surface scans

Rigid PES scans

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Rigid PES scans are requested by specifying RUNTYP=SURFACE in $CONTRL. Keywords of the $SURF group can be used to set up the scan. In Firefly, rigid PES scans can be used at any level of theory as only energies are required.

The following keywords of $SURF can be used to set up the scan:

IVEC1 = an array of two atoms, defining a coordinate from the first atom given to the second.

IGRP1 = an array specifying a group of atoms, which must include the second atom given in IVEC1. The entire group will be translated (rigidly) along the vector IVEC1, relative to the first atom given in IVEC1.

ORIG1 = starting value of the coordinate, which may be positive or negative. Zero corresponds to the distance given in $DATA.

DISP1 = step size for the coordinate.

NDISP1 = number of steps to take for this coordinate.

There are no reasonable defaults for these keywords, so you should input all of them. ORIG1 and DISP1 should be given in Ångstrom.

In addition, the IVEC2, IGRP2, ORIG2, DISP2, and NDISP2 keywords exist. These function identical to their "1" counterparts, and permit you to make a two dimensional map along two displacement coordinates. If the "2" keywords are not input, the surface map proceeds in only one dimension.

One additional keyword that should be mentioned is NSURF, which specifies for how many states energies will be printed (the default is 1). Note that for TD/CIS runs the amount of states for which energies are printed equals NSURF+1 as ground state energies will be printed as well. When setting NSURF, always make sure you request enough states through use of the keywords appropriate to the method used (e.g., NSTATE in $DET or $GUGDIA for MCSCF, NSTATE in $TDDFT for TDDFT, etc.).

See the next section for input examples.

Relaxed PES scans

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Relaxed PES scans are requested by specifying RUNTYP=RSURFACE in $CONTRL. As for rigid scans, keywords of the $SURF group can be used to set up the scan. Relaxed require the availability of gradients but can also be used with methods for which only energies are available by use of Firefly's numerical gradient feature.

RSURFACE runs should use IFREEZ in $STATPT to manually freeze the same Cartesian/internal coordinates or their combinations which are selected to scan in $surf via VECT1/VECT2. In general, it is recommended to run relaxed scans in delocalized coordinates (DLCs) together with the $ZMAT SCAN=.T. option. The SCAN keyword triggers the use of a specific scan engine which is designed for the use with DLCs.

A few important keywords should be mentioned. VECT1 and VECT2 are double precision arrays containing coefficients of the Cartesian or internal coordinates forming displacements to scan, with the numbering scheme the same as of IFREEZ array in $STATPT. VECT1 (and VECT2) runs should be used in conjunction with $DISP_i$ and $NDISP_i$. Use of $ORIG_i$ is also allowed. Finally, the NORMV keyword is seldom used and means to renorm vectors defined *via* VECT1 or VECT2 to unity.

Examples of input files for rigid and relaxed PES scans can be found at the following locations:

http://classic.chem.msu.su/gran/gamess/pes.html
http://classic.chem.msu.su/gran/gamess/dlcscan.html
http://classic.chem.msu.su/gran/gamess/rsurf_samples.rar
http://classic.chem.msu.su/gran/gamess/firefly_dlc_scans.rar

Reaction coordinate scans

Intrinsic reaction coordinate

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

The Intrinsic Reaction Coordinate (IRC) is defined as the minimum energy path connecting the reactants to products via the transition state. In practice, the IRC is found by first locating the transition state for the reaction. The IRC is then found in halves, going forward and backwards from the transition state, down the steepest descent path in mass weighted Cartesian coordinates. This is accomplished by numerical integration of the IRC equations, by a variety of methods to be described below.

The IRC is becoming an important part of polyatomic dynamics research, as it is hoped that only knowledge of the PES in the vicinity of the IRC is needed for prediction of reaction rates, at least at threshold energies. The IRC has a number of uses for electronic structure purposes as well. These include the proof that a certain transition structure does indeed connect a particular set of reactants and products, as the structure and imaginary frequency normal mode at the transition state do not always unambiguously identify the reactants and products. The study of the electronic and geometric structure along the IRC is also of interest. For example, one can obtain localized orbitals along the path to determine when bonds break or form.

The accuracy to which the IRC is determined is dictated by the use one intends for it. Dynamical calculations require a very accurate determination of the path, as derivative information (second derivatives of the PES at various IRC points, and path curvature) is required later. Thus, a sophisticated integration method (such as AMPC4 or RK4), and small step sizes (STRIDE=0.05, 0.01, or even smaller) may be needed. In addition to this, care should be taken to locate the transition state carefully (perhaps decreasing OPTTOL by a factor of 10), and in the initiation of the IRC run. The latter might require a Hessian matrix obtained by double differencing, certainly the Hessian should be PURIFY'd. Note also that EVIB must be chosen carefully, as described below.

On the other hand, identification of reactants and products allows for much larger step sizes, and cruder integration methods. In this type of IRC one might want to be careful in leaving the transition state (perhaps STRIDE should be reduced to 0.10 or 0.05 for the first few steps away from the transition state), but once a few points have been taken, larger step sizes can be employed. In general, the defaults in the $IRC group are set up for this latter, cruder quality IRC. The STRIDE value for the GS2 method can usually be safely larger than for other methods, no matter what your interest in accuracy is.

The simplest method of determining an IRC is linear gradient following, PACE=LINEAR. This method is also known as Euler's method. If you are employing PACE=LINEAR, you can select "stabilization" of the reaction path by the Ishida, Morokuma, Komornicki method. This type of corrector has no apparent mathematical basis, but works rather well since the bisector usually intersects the reaction path at right angles (for small step sizes). The ELBOW variable allows for a method intermediate to LINEAR and stabilized LINEAR, in that the stabilization will be skipped if the gradients at the original IRC point, and at the result of a linear prediction step form an angle greater than ELBOW. Set ELBOW=180 to always perform the stabilization.

A closely related method is PACE=QUAD, which fits a quadratic polynomial to the gradient at the current and immediately previous IRC point to predict the next point. This pace has the same computational requirement as LINEAR, and is slightly more accurate due to the reuse of the old gradient. However, stabilization is not possible for this pace, thus a stabilized LINEAR path is usually more accurate than QUAD.

Two rather more sophisticated methods for integrating the IRC equations are the fourth order Adams-Moulton predictor-corrector (PACE=AMPC4) and fourth order Runge Kutta (PACE=RK4). AMPC4 takes a step towards the next IRC point (prediction), and based on the gradient found at this point (in the near vincinity of the next IRC point) obtains a modified step to the desired IRC point (correction). AMPC4 uses variable step sizes, based on the input STRIDE. RK4 takes several steps part way toward the next IRC point, and uses the gradient at these points to predict the next IRC point. RK4 is the most accurate integration method implemented in Firefly, and is also the most time consuming.

The Gonzalez-Schlegel 2nd order method finds the next IRC point by a constrained optimization on the surface of a hypersphere, centered at 1/2 STRIDE along the gradient vector leading from the previous IRC point. By construction, the reaction path between two successive IRC points is thus a circle tangent to the two gradient vectors. The algorithm is much more robust for large steps than the other methods, so it has been chosen as the default method. Thus, the default for STRIDE is too large for the other methods. The number of energy and gradients need to find the next point varies with the difficulty of the constrained optimization, but is normally not very many points. Be sure to provide the updated Hessian from the previous run when restarting PACE=GS2.

The number of wavefunction evaluations, and energy gradients needed to jump from one point on the IRC to the next point are summarized in the following table:

PACE	# energies	# gradients	
LINEAR	1	1	
stabilized LINEAR	3	2	
QUAD	1	1	(+ reuse of historical gradient)
AMPC4	2	2	(see note)

RK4	4	4
GS2	2-4	2-4 (equal numbers)

Note that the AMPC4 method sometimes does more than one correction step, with each such correction adding one more energy and gradient to the calculation. You get what you pay for in IRC calculations: the more energies and gradients which are used, the more accurate the path found.

A description of these methods, as well as some others that were found to be not as good is geven by Kim Baldridge and Lisa Pederson, Pi Mu Epsilon Journal, 9, 513-521 (1993).

All methods are initiated by jumping from the first-order saddle point, parallel to the normal mode (CMODE) which has an imaginary frequency. The jump taken is designed to lower the energy by an amount EVIB. The actual distance taken is thus a function of the imaginary frequency, as a smaller FREQ will produce a larger initial jump. You can simply provide a $HESS group instead of CMODE and FREQ, which involves less typing. To find out the actual step taken for a given EVIB, use EXETYP=CHECK. The direction of the jump (towards reactants or products) is governed by FORWRD. Note that if you have decided to use small step sizes, you must employ a smaller EVIB to ensure a small first step. The GS2 method begins by following the normal mode by one half of STRIDE, and then performing a hypersphere minimization about that point, so EVIB is irrelevant to this PACE.

The only method which proves that a properly converged IRC has been obtained is to regenerate the IRC with a smaller step size, and check that the IRC is unchanged. Again, note that the care with which an IRC must be obtained is highly dependent on what use it is intended for.

A small program which converts the IRC results punched to file IRCDATA into a format close to that required by the POLYRATE VTST dynamics program written in Don Truhlar's group is available. For a copy of this IRCED program, contact Mike Schmidt. The POLYRATE program must be obtained from the Truhlar group.

Some key IRC references are:

K.Ishida, K.Morokuma, A.Komornicki J.Chem.Phys. 66, 2153-2156 (1977)
K.Muller Angew.Chem., Int.Ed.Engl.19, 1-13 (1980)
M.W.Schmidt, M.S.Gordon, M.Dupuis J.Am.Chem.Soc. 107, 2585-2589 (1985)
B.C.Garrett, M.J.Redmon, R.Steckler, D.G.Truhlar, K.K.Baldridge, D.Bartol, M.W.Schmidt, M.S.Gordon J.Phys.Chem. 92, 1476-1488(1988)
K.K.Baldridge, M.S.Gordon, R.Steckler, D.G.Truhlar J.Phys.Chem. 93, 5107-5119(1989)
C.Gonzales, H.B.Schlegel J.Chem.Phys. 90, 2154-2161(1989)

The IRC discussion closes with some practical tips:

The $IRC group has a confusing array of variables, but fortunately very little thought need be given to most of them. An IRC run is restarted by moving the coordinates of the next predicted IRC point into $DATA, and inserting the new $IRC group into your input file. You must select the de-

sired value for NPOINT. Thus, only the first job which initiates the IRC requires much thought about $IRC.

The symmetry specified in the $DATA deck should be the symmetry of the reaction path. If a saddle point happens to have higher symmetry, use only the lower symmetry in the $DATA deck when initiating the IRC. The reaction path will have a lower symmetry than the saddle point whenever the normal mode with imaginary frequency is not totally symmetric. Be careful that the order and orientation of the atoms corresponds to that used in the run which generated the Hessian matrix.

If you wish to follow an IRC for a different isotope, use the $MASS group. If you wish to follow the IRC in regular Cartesian coordinates, just enter unit masses for each atom. Note that CMODE and FREQ are a function of the atomic masses, so either regenerate FREQ and CMODE, or more simply, provide the correct $HESS group.

Dynamic reaction coordinate

The dynamical reaction coordinate is a time-dependent classical trajectory method based on quantum chemical potential energy surfaces. In Firefly, DRCs may be calculated by wave mechanics, DFT, and semiempirical methods. Because the vibrational period of a normal mode with frequency 500 wavenumbers is 67 fs, a DRC needs to run for many steps in order to sample a representative portion of phase space.

DRC runs are requested with RUNTYP=DRC in $CONTRL. Keywords that control the DRC run are of the $DRC group. Only a few important keywords will be mentioned here. The number of steps can be set with the NSTEP keywords. The time step size (in fs) is specified with the DELTAT keyword. By default, it is assumed that a started DRC run is a new one and thus starts at 0 seconds. If this is not the case, the starting point can be set nonzero with the TOTIME keyword (in fs).

In general, a DRC can be initiated anywhere, so $DATA might contain coordinates of the equilibrium geometry, or a nearby transition state, or something else. Almost all DRCs break molecular symmetry, so one should build their molecule with C1 symmetry in $DATA, or specify NOSYM=1 in $CONTRL. Also, one must supply an initial kinetic energy, and the direction of the initial velocity, which can be done with the EKIN (the initial kinetic energy) and VEL (an array of velocity components) keywords. The NVEL keyword is a flag to compute the initial kinetic energy from the input VEL using the sum of mass*VEL*VEL/2 (this is usually done only for restarts).

There are five different ways to specify the DRC trajectory:

1. VEL vector with NVEL=.TRUE. This is difficult to specify at your initial point, and so this option is mainly used when restarting your trajectory. The restart information is always in this format.

2. VEL vector and EKIN with NVEL=.FALSE. This will give a desired amount of kinetic energy in the direction of the velocity vector.

3. VIBLVL and VIBENG selected, to give initial kinetic energy to all of the normal modes.

4. NNM and ENM to give quanta to a single normal mode.

5. NNM and EKIN to give arbitrary kinetic energy to a single normal mode.

More information on DRC runs can be found in the list of keywords ($DRC) as well as in the following references:

1. J. J. P. Stewart, L. P. Davis, L. W. Burggraf *J. Comput. Chem.* 8, 1117-1123 (1987)
2. S. A. Maluendes, M. Dupuis *J. Chem. Phys.* 93, 5902-5911 (1990)
3. T. Taketsugu, M. S. Gordon *J. Phys. Chem.* 99, 8462-8471 (1995)
4. T. Taketsugu, M. S. Gordon *J. Phys. Chem.* 99, 14597-604 (1995)
5. T. Taketsugu, M. S. Gordon *J. Chem. Phys.* 103, 10042-10049 (1995)
6. M. S. Gordon, G. Chaban, T. Taketsugu *J. Phys. Chem.* 100, 11512-11525 (1996)
7. T. Takata, T. Taketsugu, K. Hirao, M. S. Gordon *J. Chem. Phys.* 109, 4281-4289 (1998)
8. T. Taketsugu, T. Yanai, K. Hirao, M. S. Gordon *THEOCHEM* 451, 163-177 (1998)

Gradient extremal following

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

This section of the manual, as well as the source code to trace gradient extremals was written by Frank Jensen of Odense University.

A Gradient Extremal (GE) curve consists of points where the gradient norm on a constant energy surface is stationary. This is equivalent to the condition that the gradient is an eigenvector of the Hessian. Such GE curves radiate along all normal modes from a stationary point, and the GE leaving along the lowest normal mode from a minimum is the gentlest ascent curve. This is not the same as the IRC curve connecting a minimum and a TS, but may in some cases be close.

GEs may be divided into three groups: those leading to dissociation, those leading to atoms colliding, and those which connect stationary points. The latter class allows a determination of many (all?) stationary points on a PES by tracing out all the GEs. Following GEs is thus a semi-systematic way of mapping out stationary points.

The disadvantages are:

i) There are many (but finitely many!) GEs for a large molecule.
ii) Following GEs is computationally expensive.
iii) There is no control over what type of stationary point (if any) a GE will lead to.

Normally one is only interested in minima and TSs, but many higher order saddle points will also be found. Furthermore, it appears that it is necessary to follow GEs radiating also from TSs and second (and possibly also higher) order saddle point to find all the TSs.

A rather complete map of the extremals for the H2CO potential surface is available in a paper which explains the points just raised in greater detail:

K.Bondensgaard, F.Jensen, J.Chem.Phys. 104, 8025-8031(1996).

An earlier paper gives some of the properties of GEs:

D.K.Hoffman, R.S.Nord, K.Ruedenberg, Theor. Chim. Acta 69, 265-279(1986).

There are two GE algorithms in Firefly, one due to Sun and Ruedenberg (METHOD=SR), which has been extended to include the capability of locating bifurcation points and turning points, and another due to Jorgensen, Jensen, and Helgaker (METHOD=JJH):

J. Sun, K. Ruedenberg, J.Chem.Phys. 98, 9707-9714(1993)
P. Jorgensen, H. J. Aa. Jensen, T. Helgaker Theor. Chim. Acta 73, 55 (1988).

The Sun and Ruedenberg method consist of a predictor step taken along the tangent to the GE curve, followed by one or more corrector steps to bring the geometry back to the GE. Construction of the GE tangent and the corrector step requires elements of the third derivative of the energy, which is obtained by a numerical differentiation of two Hessians. This puts some limitations on which systems the GE algorithm can be used for. First, the numerical differentiation of the Hessian to produce third derivatives means that the Hessian should be calculated by analytical methods, thus only those types of wavefunctions where this is possible can be used. Second, each predictor/corrector step requires at least two Hessians, but often more. Maybe 20-50 such steps are necessary for tracing a GE from one stationary point to the next. A systematic study of all the GE radiating from a stationary point increases the work by a factor of ~2*(3N-6). One should thus be prepared to invest at least hundreds, and more likely thousands, of Hessian calculations. In other words, small systems, small basis sets, and simple wavefunctions.

The Jorgensen, Jensen, and Helgaker method consists of taking a step in the direction of the chosen Hessian eigenvector, and then a pure NR step in the perpendicular modes. This requires (only) one Hessian calculation for each step. It is not suitable for following GEs where the GE tangent forms a large angle with the gradient, and it is incapable of locating GE bifurcations.

Although experience is limited at present, the JJH method does not appear to be suitable for following GEs in general (at least not in the current implementation). Experiment with it at your own risk!

The flow of the SR algorithm is as follows: A predictor geometry is produced, either by jumping away from a stationary point, or from a step in the tangent direction from the previous point on the GE. At the predictor geometry, we need the gradient, the Hessian, and the third derivative in the gradient direction. Depending on HSDFDB, this can be done in two ways. If .TRUE. the gradient is calculated, and two Hessians are calculated at SNUMH distance to each side in the gradient direction. The Hessian at the geometry is formed as the average of the two displaced Hessians. This corresponds to a doublesided differentiation, and is the numerical most stable method for getting the partial third derivative matrix. If HSDFDB = .FALSE., the gradient and Hessian are calculated at the current geometry, and one additional Hessian is calculated at SNUMH distance in the gradient direction. This corresponds to a single-sided differentiation. In both cases, two full Hessian calculations are necessary, but HSDFDB = .TRUE. require one additional wavefunction and gradient calculation. This is usually a fairly small price compared to two Hessians, and the numerically better double-sided differentiation has therefore been made the default.

Once the gradient, Hessian, and third derivative is available, the corrector step and the new GE tangent are constructed. If the corrector step is below a threshold, a new predictor step is taken along the tangent vector. If the corrector step is larger than the threshold, the correction step is taken, and a new micro iteration is performed. DELCOR thus determines how closely the GE will be followed, and DPRED determine how closely the GE path will be sampled.

The construction of the GE tangent and corrector step involve solution of a set of linear equations, which in matrix notation can be written as Ax=B. The A-matrix is also the second derivative of the gradient norm on the constant energy surface.

After each corrector step, various things are printed to monitor the behavior: The projection of the gradient along the Hessian eigenvalues (the gradient is parallel to an eigenvector on the GE), the projection of the GE tangent along the Hessian eigenvectors, and the overlap of the Hessian eigenvectors with the mode being followed from the previous (optimzed) geometry. The sign of these overlaps are not significant, they just refer to an arbitrary phase of the Hessian eigenvectors.

After the micro iterations has converged, the Hessian eigenvector curvatures are also displayed, this is an indication of the coupling between the normal modes. The number of negative eigenvalues in the A-matrix is denoted the GE index. If it changes, one of the eigenvalues must have passed through zero. Such points may either be GE bifurcations (where two GEs cross) or may just be "turning points", normally when the GE switches from going uphill in energy to downhill, or vice versa. The distinction is made based on the B-element corresponding to the A-matrix eigenvalue = 0. If the B-element = 0, it is a bifurcation, otherwise it is a turning point.

If the GE index changes, a linear interpolation is performed between the last two points to locate the point where the A-matrix is singular, and the corresponding B-element is determined. The linear interpolation points will in general be off the GE, and thus the evaluation of whether the B-element is 0 is not always easy. The program additionally evaluates the two limiting vectors which are solutions to the linear sets of equations, these are also used for testing whether the singular point is a bifurcation point or turning point.

Very close to a GE bifurcation, the corrector step become numerically unstable, but this is rarely a problem in practice. It is a priori expected that GE bifurcation will occur only in symmetric systems, and the crossing GE will break the symmetry. Equivalently, a crossing GE may be encountered when a symmetry element is formed, however such crossings are much harder to detect since the GE index does not change, as one of the A-matrix eigenvalues merely touches zero. The program prints an message if the absolute value of an A-matrix eigenvalue reaches a minimum near zero, as such points may indicate the passage of a bifurcation where a higher symmetry GE crosses. Run a movie of the geometries to see if a more symmetric structure is passed during the run.

An estimate of the possible crossing GE direction is made at all points where the A-matrix is singular, and two perturbed geometries in the + and - direction are written out. These may be used as predictor geometries for following a crossing GE. If the singular geometry is a turning point, the + and - geometries are just predictor geometries on the GE being followed.
In any case, a new predictor step can be taken to trace a different GE from the newly discovered singular point, using the direction determined by interpolation from the two end point tangents (the GE tangent cannot be uniquely determined at a bifurcation point). It is not possible to determine what the sign of IFOLOW should be when starting off along a crossing GE at a bifurcation, one will have to try a step to see if it returns to the bifurcation point or not.

In order to determine whether the GE index change it is necessary to keep track of the order of the A-matrix eigenvalues. The overlap between successive eigenvectors are shown as "Alpha mode overlaps".

Things to watch out for:

1) The numerical differentiation to get third derivatives requires more accuracy than usual. The SCF convergence should be at least 100 times smaller than SNUMH, and preferably better. With the default SNUMH of 10**(-4) the SCF convergence should be at least 10**(-6). Since the last few SCF cycles are inexpensive, it is a good idea to tighten the SCF convergence as much as possible, to maybe 10**(-8) or better. You may also want to increase the integral accuracy by reducing the cutoffs (ITOL and ICUT) and possibly also try more accurate integrals (INTTYP=HONDO). The CUTOFF in $TRNSFM may also be reduced to produce more accurate Hessians. Don't attempt to use a value for SNUMH below 10**(-6), as you simply can't get enough accuracy. Since experience is limited at present, it is recommended that some tests runs are made to learn the sensitivity of these factors for your system.

2) GEs can be followed in both directions, uphill or downhill. When stating from a stationary point, the direction is implicitly given as away from the stationary point. When starting from a non-stationary point, the "+" and "-" directions (as chosen by the sign of IFOLOW) refers to the gradient direction. The "+" direction is along the gradient (energy increases) and "-" is opposite to the gradient (energy decreases).

3) A switch from one GE to another may be seen when two GE come close together. This is especially troublesome near bifurcation points where two GEs actually cross. In such cases a switch to a GE with -higher- symmetry may occur without any indication that this has happened, except possibly that a very large GE curvature suddenly shows up. Avoid running the calculation with less symmetry than the system actually has, as this increases the likelihood that such switches occurring. Fix: alter DPRED to avoid having the predictor step close to the crossing GE.

4) "Off track" error message: The Hessian eigenvector which is parallel to the gradient is not the same as the one with the largest overlap to the previous Hessian mode. This usually indicate that a GE switch has occured (note that a switch may occur without this error message), or a wrong value for IFOLOW when starting from a non-stationary point. Fix: check IFOLOW, if it is correct then reduce DPRED, and possibly also DELCOR.

5) Low overlaps of A-matrix eigenvectors. Small overlaps may give wrong assignment, and wrong conclusions about GE index change. Fix: reduce DPRED.

6) The interpolation for locating a point where one of the A-matrix eigenvalues is zero fail to converge. Fix: reduce DPRED (and possibly also DELCOR) to get a shorter (and better) interpolation line.

7) The GE index changes by more than 1. A GE switch may have occurred, or more than one GE index change is located between the last and current point. Fix: reduce DPRED to sample the GE path more closely.

8) If SNRMAX is too large the algorithm may try to locate stationary points which are not actually on the GE being followed. Since GEs often pass quite near a stationary point, SNRMAX should only be increased above the default 0.10 after some consideration.

Solvation models

Introduction

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

In a very thorough 1994 review of continuum solvation models, Tomasi and Persico divide the possible approaches to the treatment of solvent effects into four categories:

a) virial equations of state, correlation functions
b) Monte Carlo or molecular dynamics simulations
c) continuum treatments
d) molecular treatments

The Effective Fragment Potential method, documented in the following section of this chapter, falls into the latter category, as each EFP solvent molecule is modeled as a distinct object. This section describes the two continuum models which are implemented in the standard version of Firefly.

Continuum models typically form a cavity of some sort containing the solute molecule, while the solvent outside the cavity is thought of as a continuous medium and is categorized by a limited amount of physical data, such as the dielectric constant. The electric field of the charged particles comprising the solute interact with this background medium, producing a polarization in it, which in turn feeds back upon the solute's wavefunction.

Self-consistent reaction field

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

A simple continuum model is the Onsager cavity model, often called the Self-Consistent Reaction Field, or SCRF model. This represents the charge distribution of the solute in terms of a multipole expansion. SCRF usually uses an idealized cavity (spherical or ellipsoidal) to allow an analytic solution to the interaction energy between the solute multipole and the multipole which this induces in the continuum. This method is implemented in Firefly in the simplest possible fashion:

i) a spherical cavity is used
ii) the molecular electrostatic potential of the solute is represented as a dipole only, except a monopole is also included for an ionic solute.

The input for this implementation of the Kirkwood-Onsager model is provided in $SCRF.

Some references on the SCRF method are

1. J.G.Kirkwood J.Chem.Phys. 2, 351 (1934)
2. L.Onsager J.Am.Chem.Soc. 58, 1486 (1936)
3. O.Tapia, O.Goscinski Mol.Phys. 29, 1653 (1975)
4. M.M.Karelson, A.R.Katritzky, M.C.Zerner Int.J.Quantum Chem., Symp. 20, 521-527 (1986)
5. K.V.Mikkelsen, H.Agren, H.J.Aa.Jensen, T.Helgaker J.Chem.Phys. 89, 3086-3095 (1988)
6. M.W.Wong, M.J.Frisch, K.B.Wiberg J.Am.Chem.Soc. 113, 4776-4782 (1991)
7. M.Szafran, M.M.Karelson, A.R.Katritzky, J.Koput, M.C.Zerner J.Comput.Chem. 14, 371-377 (1993)
8. M.Karelson, T.Tamm, M.C.Zerner J.Phys.Chem. 97, 11901-11907 (1993)

The method is very sensitive to the choice of the solute RADIUS, but not very sensitive to the particular DIELEC of polar solvents. The plots in reference 7 illustrate these points very nicely. The SCRF implementation in Firefly is Zerner's Method A, described in the same reference. The total solute energy includes the Born term, if the solute is an ion. Another limitation is that a solute's electrostatic potential is not likely to be fit well as a dipole moment only, for example see Table VI of reference 5 which illustrates the importance of higher multipoles. Finally, the restriction to a spherical cavity may not be very representative of the solute's true shape. However, in the special case of a roundish molecule, and a large dipole which is geometry sensitive, the SCRF model may include sufficient physics to be meaningful:

M.W.Schmidt, T.L.Windus, M.S.Gordon J.Am.Chem.Soc. 117, 7480-7486(1995).

Polarizable continuum model

The following text might currently not be complete and/or contain outdated information, but will be improved in the future.

A much more sophisticated continuum method, named the Polarizable Continuum Model, is also available. The PCM method places a solute in a cavity formed by a union of spheres centered on each atom. PCM also includes a more exact treatment of the electrostatic interaction with the surrounding medium, as the electrostatic potential of the solute generates an 'apparent surface charge' on the cavity's surface. The computational procedure divides this surface into small tesserae, on which the charge (and contributions to the gradient) are evaluated. Typically the spheres defining the cavity are taken to be 1.2 times the van der Waals radii. A technical difficulty caused by the penetration of the solute charge density outside this cavity is dealt with by a renormalization. Procedures are provided not only for the computation of the electrostatic interaction of the solute with the apparent surface charges, but also for the cavitation energy, and dispersion and repulsion contributions to the solvation free energy.

Two variants of PCM have been implemented in Firefly: dielectric PCM (DPCM) in which the solvent is treated in terms of dielectrics, and conductor-like PCM (CPCM) in which the solvent is treated as a conductor-like continuum. CPCM can be seen as an implementation of the COSMO solvation model in the PCM framework. Another popular PCM variant, IEFPCM, is currently not available in Firefly.

The main input group is $PCM. The PCMTYP keyword can be used to specify the type PCM, possible values being DPCM and CPCM. For CPCM, the value of 'x' in the equation

$$f(\varepsilon) = (\varepsilon - 1) / (\varepsilon + x)$$

can be set with the XCPCM keyword. The default is XCPCM=0.0, which makes the model identical to CPCM as implemented most frequently. Setting XCPCM=0.5 makes the solvent model identical to COSMO.

Another important keyword is IXPCM. Setting IXPCM=2 activates Firefly-specific, fully variational DPCM and CPCM models. These models have not yet been published.

The $PCMCAV group can be used to provide auxiliary cavity information. If any of the optional energy computations are requested in $PCM, the additional input groups $NEWCAV, $DISBS, or $DISREP may be required.

Solvation of course affects the non-linear optical properties of molecules. The PCM implementation extends RUNTYP=TDHF to include solvent effects. Both static and frequency dependent hyperpolarizabilities can be found. Besides the standard PCM electrostatic contribution, the IREP and IDP keywords can be used to determine the effects of repulsion and dispersion on the polarizabilities.

Due to its sophistication, users of the (D)PCM model are strongly encouraged to read the primary literature.

General papers on the PCM method:
1) S.Miertus, E.Scrocco, J.Tomasi Chem.Phys. 55, 117-129(1981)
2) J.Tomasi, M.Persico Chem.Rev. 94, 2027-2094(1994)
3) R.Cammi, J.Tomasi J.Comput.Chem. 16, 1449-1458(1995)

The GEPOL method for cavity construction:
4) J.L.Pascual-Ahuir, E.Silla, J.Tomasi, R.Bonaccorsi J.Comput.Chem. 8, 778-787(1987) Charge renormalization (see also ref. 3):
5) B.Mennucci, J.Tomasi J.Chem.Phys. 106, 5151-5158(1997)

Derivatives with respect to nuclear coordinates (energy gradient and Hessian):
6) R.Cammi, J.Tomasi J.Chem.Phys. 100, 7495-7502(1994)
7) R.Cammi, J.Tomasi J.Chem.Phys. 101, 3888-3897(1995)
8) M.Cossi, B.Mennucci, R.Cammi J.Comput.Chem. 17, 57-73(1996)

Derivatives with respect to applied electric fields (polarizabilities and hyperpolarizabilities):
9) R.Cammi, J.Tomasi Int.J.Quantum Chem. Symp. 29, 465-474(1995)

10) R.Cammi, M.Cossi, J.Tomasi J.Chem.Phys. 104, 4611-4620(1996) 11) R.Cammi, M.Cossi, B.Mennucci, J.Tomasi J.Chem.Phys. 105, 10556-10564(1996) 12) B. Mennucci, C. Amovilli, J. Tomasi J.Chem.Phys. submitted.

Cavitation energy:
13) R.A.Pierotti Chem.Rev. 76, 717-726(1976)
14) J.Langlet, P.Claverie, J.Caillet, A.Pullman J.Phys.Chem. 92, 1617-1631(1988)

Dispersion and repulsion energies:
15) F.Floris, J.Tomasi J.Comput.Chem. 10, 616-627(1989)
16) C.Amovilli, B.Mennucci J.Phys.Chem. B101, 1051-1057(1997)

At the present time, the PCM model in Firefly has the following limitations:

a) SCFTYP=RHF, ROHF, UHF, and MCSCF, only.
b) point group symmetry is switched off internally during PCM.
c) electric field integrals at normals to the surface elements are stored on disk, even during DIRSCF runs. The file size may be considerable.
d) nuclear derivatives are limited to gradients, although theory for Hessians is given in Ref. 7.

Effective fragment potentials

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

The basic idea behind the effective fragment potential (EFP) method is to replace the chemically inert part of a system by EFPs, while performing a regular ab initio calculation on the chemically active part. Here "inert" means that no covalent bond breaking process occurs. This "spectator region" consists of one or more "fragments", which interact with the ab initio "active region" through non-bonded interactions, and so of course these EFP interactions affect the ab initio wavefunction. A simple example of an active region might be a solute molecule, with a surrounding spectator region of solvent molecules represented by fragments. Each discrete solvent molecule is represented by a single fragment potential, in marked contrast to continuum models for solvation.

The quantum mechanical part of the system is entered in the $DATA group, along with an appropriate basis. The EFPs defining the fragments are input by means of a $EFRAG group, one or more $FRAGNAME groups describing each fragment's EFP, and a $FRGRPL group. These groups define non-bonded interactions between the ab initio system and the fragments, and between the fragments. The former interactions enter via one-electron operators in the ab initio Hamiltonian, while the latter interactions are treated by analytic functions. The only electrons explicitly treated (e.g. with basis functions used to expand occupied orbitals) are those in the active region, so there are no new two electron terms. Thus the use of EFPs leads to significant time savings compared to full ab initio calculations on the same system.

At ISU, the EFPs are currently used to model RHF/DZP water molecules in order to study aqueous solvation effects, for example references 1,2,3. Our co-workers at NIST have also used EFPs to model parts of enzymes, see reference 4.

Terms in an EFP

The non-bonded interactions currently implemented are:

1) Coulomb interaction. The charge distribution of the fragments is represented by an arbitrary number of charges, dipoles, quadrupoles, and octupoles, which interact with the ab initio hamiltonian as well as with multipoles on other fragments. It is possible to input a screening term that accounts for the charge penetration. Typically the multipole expansion points are located on atomic nuclei and at bond midpoints.

2) Dipole polarizability. An arbitrary number of dipole polarizability tensors can be used to calculate the induced dipole on a fragment due to the electric field of the ab initio system as well as all the other fragments. These induced dipoles interact with the ab initio system as well as the other EFPs, in turn changing their electric fields. All induced dipoles are therefore iterated to self-consistency. Typically the polarizability tensors are located at the centroid of charge of each localized orbital of a fragment.

3) Repulsive potential. Two different forms for the repulsive potentials are used: one for ab initio-EFP repulsion and one for EFP-EFP repulsion. The form of the potentials is empirical, and consists of distributed gaussian or exponential functions, respectively. The primary contribution to the repulsion is the quantum mechanical exchange repulsion, but the fitting technique used to develop this term also includes the effects of charge transfer. Typically these fitted potentials are located on atomic nuclei within the fragment.

Constructing an EFP using Firefly

RUNTYP=MOROKUMA assists in the decomposition of intermolecular interaction energies into electrostatic, polarization, charge transfer, and exchange repulsion contributions. This is very useful in developing EFPs since potential problems can be attributed to a particular term by comparison to these energy components for a particular system.

A molecular multipole expansion can be obtained using $ELMOM. A distributed multipole expansion can be obtained by either a Mulliken-like partitioning of the density (using $STONE) or by using localized molecular orbitals ($LOCAL: DIPDCM and QADDCM). The molecular dipole polarizability tensor can be obtained during a Hessian run ($CPHF), and a distributed LMO polarizability expression is also available ($LOCAL: POLDCM).

The repulsive potential is derived by fitting the difference between ab initio computed intermolecular interaction energies, and the form used for Coulomb and polarizability interactions. This difference is obtained at a large number of different interaction geometries, and is then fitted. Thus,

the repulsive term is implicitly a function of the choices made in representing the Coulomb and polarizability terms. Note that Firefly currently does not provide a way to obtain these repulsive potential, or the charge interpenetration screening parameters.

Since you cannot develop all terms necessary to define a new EFP's $FRAGNAME group using Firefly, in practice you will be limited to using the internally stored H2OEF2 potential mentioned below.

Current Limitations

1. The energy and energy gradient are programmed, which permits RUNTYP=ENERGY, GRADIENT, and numerical HESSIAN. The necessary modifications to use the EFP gradients while moving on the potential surface are programmed for RUNTYP=OPTIMIZE, SADPOINT, and IRC (see reference 3), but the other gradient based potential surface explorations such as DRC are not yet available. Finally, RUNTYP=PROP is also permissible.

2. The ab initio system must be treated with RHF, ROHF, UHF, or the open shell SCF wavefunctions permitted by the GVB code. The correlated methods in Firefly (MP2 and CI) should not be used, since the available H2OEF2 potential was derived at the RHF level, and therefore does not contain dispersion terms. A correlated computation on the ab initio system without these terms in the EFP will probably lead to unphysical results. MCSCF and GVB-PP represent a gray area, but Mo Krauss has obtained some results for a solute described by an MCSCF wavefunction in which the EFP results agree well with fully ab initio computations (in structures and interaction energies).

3. EFPs can move relative to the ab initio system and relative to each other, but the internal structure of an EFP is frozen.

4. The boundary between the ab initio system and the EFPs must not be placed across a chemical bond.

5. Calculations must be done in C1 symmetry at present. Enter NOSYM=1 in $CONTRL to accomplish this.

6. Reorientation of the fragments and ab initio system is not well coordinated. If you are giving Cartesian coordinates for the fragments (COORD=CART in $EFRAG), be sure to use $CONTRL's COORD=UNIQUE option so that the ab initio molecule is not reoriented.

7. If you need IR intensities, you have to use NVIB=2. The potential surface is usually very soft for EFP motions, and double differenced Hessians should usually be obtained.

Practical hints for using EFPs

At the present time, we have only one EFP suitable for general use. This EFP models water, and its numerical parameters are internally stored, using the fragment name H2OEF2. These numerical parameters are improved values over the H2OEF1 set which were presented and used in reference 2, and they also include the improved EFP-EFP repulsive term defined in reference 3.

The H2OEF2 water EFP was derived from RHF/DH(d,p) computations on the water dimer system. When you use it, therefore, the ab initio part of your system should be treated at the SCF level, using a basis set of the same quality (ideally DH(d,p), but probably other DZP sets such as 6-31G(d,p) will give good results as well). Use of better basis sets than DZP with this water EFP has not been tested.

As noted, effective fragments have frozen internal geometries, and therefore only translate and rotate with respect to the ab initio region. An EFP's frozen coordinates are positioned to the desired location(s) in $EFRAG as follows:

a) the corresponding points are found in $FRAGNAME.
b) Point -1- in $EFRAG and its FRAGNAME equivalent are made to coincide.
c) The vector connecting -1- and -2- is aligned with the corresponding vector connecting FRAGNAME points.
d) The plane defined by -1-, -2-, and -3- is made to coincide with the corresponding FRAGNAME plane.

Therefore the 3 points in $EFRAG define only the relative position of the EFP, and not its internal structure. So, if the "internal structure" given by points in $EFRAG differs from the true values in $FRAGNAME, then the order in which the points are given in $EFRAG can affect the positioning of the fragment. It may be easier to input water EFPs if you use the Z-matrix style to define them, because then you can ensure you use the actual frozen geometry in your $EFRAG. Note that the H2OEF2 EFP uses the frozen geometry r(OH)=0.9438636, a(HOH)=106.70327, and the names of its 3 fragment points are ZO1, ZH2, ZH3.

The translations and rotations of EFPs with respect to the ab initio system and one another are automatically quite soft degrees of freedom. After all, the EFP model is meant to handle weak interactions! Therefore the satisfactory location of structures on these flat surfaces will require use of a tight convergence on the gradient: OPTTOL=0.00001 in the $STATPT group.

EFP references

The first of these is more descriptive, and the second has a very detailed derivation of the method. The latest EFP developments are discussed in the 3rd paper.

1. "Effective fragment method for modeling intermolecular hydrogen bonding effects on quantum mechanical calculations"
J.H.Jensen, P.N.Day, M.S.Gordon, H.Basch, D.Cohen, D.R.Garmer, M.Krauss, W.J.Stevens in "Modeling the Hydrogen Bond" (D.A. Smith, ed.) ACS Symposium Series 569, 1994, pp 139-151.

2. "An effective fragment method for modeling solvent effects in quantum mechanical calculations".
P.N.Day, J.H.Jensen, M.S.Gordon, S.P.Webb, W.J.Stevens, M.Krauss, D.Garmer, H.Basch, D.Cohen J.Chem.Phys. 105, 1968-1986(1996).

3. "The effective fragment model for solvation: internal rotation in formamide"

W.Chen, M.S.Gordon, J.Chem.Phys., 105, 11081-90(1996)

4. "Transphosphorylation catalyzed by ribonuclease A: Computational study using ab initio EFPs"
B.D.Wladkowski, M. Krauss, W.J.Stevens, J.Am.Chem.Soc. 117, 10537-10545(1995).

5. "A study of aqueous glutamic acid using the effective fragment potential model"
P.N.Day, R.Pachter J.Chem.Phys. 107, 2990-9(1997)

6. "Solvation and the excited states of formamide"
M.Krauss, S.P.Webb J.Chem.Phys. 107, 5771-5(1997)

7. "Study of small water clusters using the effective fragment potential method"
G.N.Merrill, M.S.Gordon J.Phys.Chem.A 102, 2650-7(1998)

8. "Menshutkin Reaction"
S.P.Webb, M.S.Gordon J.Phys.Chem.A in press

9. "Solvation of Sodium Chloride: EFP study of NaCl(H2O)n"
C.P.Petersen, M.S.Gordon J.Phys.Chem.A 103, 4162-6(1999)

Calculating single geometry properties

Introduction

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Firefly is capable of calculation various properties. Some of these are already calculated at the end of a simple SCF job while others require a separate job in order to be obtained. This chapter attempts to provide an overview of properties available that were not yet discussed in the previous chapters.

It’s important to mention the existence of the RUNTYP=PROP capability. This can be used to quickly obtain some properties if one already has a converged set of orbitals (a $VEC group has to be supplied and read in at the start of this RUNTYP).

Electrostatic moments, electrostatic potential, electron density, electrostatic field, and electric field gradient

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Electrostatic moments are cheap and are printed at the end of most energy runs. The way in which they are calculated and printed can be controlled with keywords of the $ELMOM group. By default, only monopole and dipole moments are printed, however, quadrupole and octopole moments can also be obtained.

Calculation of the electrostatic potential can be controlled by keywords of the $ELPOT group. This property is the electrostatic potential V(a) felt by a test positive charge, due to the molecular charge density. A nucleus at the evaluation point is ignored. If this property is evaluated at the nuclei, it obeys the equation sum on nuclei(a)

Z(a)*V(a) = 2*V(nn) + V(ne)

The electronic portion of this property is called the diamagnetic shielding.

The calculation of the electron density can be controlled by keywords of the $ELDENS group.

Electrostatic field and electric field gradient calculations can be controlled by keywords of the $ELFLDG group.

Cube files

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The cube capability provides the ability to punch a 3 dimensional grid of electrostatic potential or electron density information suitable for 3D isosufacing or 3D volumetric rendering.

The cube output data are in the standard 'cube' format used by Gaussian and are suitable for use as input to a wide variety of available visualization programs (*e.g.* gOpenMol, ReView 3, Molden, etc.).

Cube input to Firely is performed using the $CUBE namelist in combination with the properties namelists $ELPOT or/ and $ELDENS. Specific $CUBE namelist variables and their definitions are as follow:

VARIABLE	VALUES	DEFINITION
CUBE	.T. or .F.	A logical flag that enables (.T.) or disables (.F.) the generation of cube data. Default is .F.
MESH	 COARSE MEDIUM FINE ULTA	Specifies desired density of points along each side of the cube. 40 points 80 points (The default) 100 points 200 points Using these mesh values as an initial guess, The cube generation program attempts to provide a common size of increment along each axis to minimize distortion in the visualization. In the process, the actual number of points along each axis may differ slightly from that listed here.
NXCUBE	> 0	The user input density of points along each x side of the cube. Default is 0. This is an integer variable.
NYCUBE	> 0	The user input density of points along each y side of the cube. Default is 0. This is an integer variable.
NZCUBE	> 0	The user input density of points along each z side of the cube. Default is 0. This is an integer variable. Note: if NXCUBE, NYCUBE, and NZCUBE are all equal to 0, all cube input will be ignored except for CUBE and MESH. All the cube parameters will be generated internally using the value of MESH as described previously.
X0CUBE		The x-coordinate of the cube origin. The default is 0.0 This is a floating point variable. Units are Bohr (au).
Y0CUBE		The y-coordinate of the cube origin. The default

		is 0.0 This is a floating point variable. Units are Bohr (au).
Z0CUBE		The z-coordinate of the cube origin. The default is 0.0 This is a floating point variable. Units are Bohr (au).
XINCUB	> 0.0	The increment (or interval size) along the x-axis of the cube. The default is 0.0 This is a floating point variable. Units are Bohr (au).
YINCUB	> 0.0	The increment (or interval size) along the y-axis of the cube. The default is 0.0 This is a floating point variable. Units are Bohr (au).
ZINCUB	> 0.0	The increment (or interval size) along the z-axis of the cube. The default is 0.0 This is a floating point variable. Units are Bohr (au).

If $ELPOT group is provided with IEPOT = 1 and the $CUBE group is provided with CUBE=.T., then a 3D "cube" format info on electrostatic potential data will be punched. The structure of the data follows the GAUSSIAN format:

Title
record for density source information
NAtoms, X0CUBE, Y0CUBE, Z0CUBE
NXCUBE, XINCUB, 0., 0.
NYCUBE, 0., YINCUB, 0.
NZUBE, 0., 0., ZINCUB
IA1, Chg1, X1, Y1, Z1 Atomic number, charge, and coordinates of the first atom
...
IAn, Chgn, Xn, Yn, Zn Atomic number, charge, and coordinates of the last atom
*(NXCUBE*NYCUBE) records*, each of length NZCUBE values of the potential at each point in the grid. Note that a separate write is used for each record.

If one wishes to read the values of the potential back into an array dimensioned X(NZCUBE,NYCUBE,NXCUBE) code like the following Fortran loop may be used:

```
      Do 10 I1 = 1, nxcube
      Do 10 I2 = 1, nycube
         Read(n,'(6E13.5)') (X(I3,I2,I1),I3=1,nzcube)
10    Continue
```

where ‘n’ is the unit number corresponding to the cube file.

Similarly, if the $ELDENS group is provided with IEDEN=1, MORB=0 and the $CUBE group is provided with CUBE=.T., then a 3D "cube" of total density data will be punched. The structure of the data follows the GAUSSIAN format as discussed previously.

Some additional keywords of the $ELDENS should be mentioned.

SPIND=.F.(default)/.T. - calculate spin density if available

DIFFD=.F.(default)/.T. - calculate delta density

SKIPHF=.T.(default)/.F. - skip (HF - initial guess) delta density calculation

DERIVS(1)=0,0,0(default) - any combination of up to three numbers 0, 1, and 2. Defines the level of derivatives required. 0 means density, 1 means the density gradient (or its norm), 2 means density Laplacian.

Also, the 'WHERE' variable of the $ELPOT and $ELDENS groups can be assigned the value CUBE (*i.e.* WHERE=CUBE). This is the default in the presence of the $CUBE group. Thus, it is not necessary to use $CUBE CUBE=.T. $END, it is enough simply to use $CUBE $END.

GRDFLD=.F.(default)/.T. - modifies density gradient calculations so that the 3-component vector gradient field is written (by default, the norm of the gradient is written).

ICORBS - integer array - indices of alpha MOs (positive) and (if available) beta MOs/NOs (negative) for which to calculate the values of orbitals (not the density!). Most of other $CUBE-related options are disabled in the presence of ICORBS array. In the PUNCH file, the inner loop of the printout is over different MOs, then over z-coordinate, etc.

UHFNOS=.F.(default)/.T. modifies ICORBS array entries so that negative entries are treated as requests to calculate UHF NOs, not UHF beta orbitals.

Below are a few input examples:

```
 $CONTRL SCFTYP=RHF RUNTYP=optimize CITYP=GUGA $END
 $SYSTEM TIMLIM=3000 MEMORY=30000000 $END
 $CIDRT  NFZC=1 NDOC=4 NVAL=57 NEXT=0 IEXCIT=2 GROUP=C2V $END
 $BASIS  GBASIS=TZV NDFUNC=3 NPFUNC=3 DIFFSP=.T. DIFFS=.T. $END
 $CUBE   MESH=coarse $END
 $ELDENS IEDEN=1  spind=.f. diffd=.t. skiphf=.f. derivs(1)=0,1,2  $END
 $STATPT OPTTOL=1D-5 $END
 $DATA
H2O
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 H           1.0   0.0000000000   0.7565140024   0.1397092302
 $END
```

Example 2:

```
 $CONTRL SCFTYP=RHF RUNTYP=optimize CITYP=GUGA $END
 $SYSTEM TIMLIM=3000 MEMORY=30000000 $END
 $CIDRT  NFZC=1 NDOC=4 NVAL=57 NEXT=0 IEXCIT=2 GROUP=C2V $END
 $BASIS  GBASIS=TZV NDFUNC=3 NPFUNC=3 DIFFSP=.T. DIFFS=.T. $END
 $CUBE   MESH=ultra $END
```

```
 $ELDENS IEDEN=1  spind=.f. diffd=.t. skiphf=.t. derivs(1)=0,2  $END
 $STATPT OPTTOL=1D-5 $END
 $DATA
H2O
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 H           1.0   0.0000000000   0.7565140024   0.1397092302
 $END
```

Example 3:

```
 $CONTRL SCFTYP=UHF RUNTYP=energy icharg=+1 mult=2 $END
 $SYSTEM TIMLIM=3000 MEMORY=30000000 $END
 $BASIS  GBASIS=TZV NDFUNC=3 NPFUNC=3 DIFFSP=.T. DIFFS=.T. $END
 $CUBE   MESH=coarse $END
! Request to calculate first 9 alpha and 7 beta orbitals.
! Minus sign means beta orbitals if available, or natural orbitals,
! if available. Note that program will rearrange MOs to simplify pro-
cessing.
! The order of MOs here is not important.
! Look into punch cube printout to find the actual order of orbitals.
 $ELDENS IEDEN=1  ICORBS(1)=1,-1,2,-2,3,-3,4,-4,5,-5,6,-6,7,-7,8,9 $END
 $DATA
H2O
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 H           1.0   0.0000000000   0.7565140024   0.1397092302
 $END
```

Example 4:

```
 $CONTRL SCFTYP=ROHF RUNTYP=energy icharg=+1 mult=2 $END
 $SYSTEM TIMLIM=3000 MEMORY=30000000 $END
 $BASIS  GBASIS=TZV NDFUNC=3 NPFUNC=3 DIFFSP=.T. DIFFS=.T. $END
 $CUBE   MESH=coarse $END
 $ELDENS IEDEN=1  MORB=4 $END
 $DATA
H2O
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 H           1.0   0.0000000000   0.7565140024   0.1397092302
 $END
```

Example 5:

```
 $CONTRL SCFTYP=UHF RUNTYP=energy icharg=+1 mult=2 $END
 $SYSTEM TIMLIM=3000 MEMORY=30000000 $END
 $BASIS  GBASIS=TZV NDFUNC=3 NPFUNC=3 DIFFSP=.T. DIFFS=.T. $END
 $CUBE   MESH=coarse $END
 $ELDENS IEDEN=1 spind=.t. grdfld=.t. derivs(1)=0,1,2 $END
```

```
 $DATA
H2O
CNV 2

 O           8.0   0.0000000000   0.0000000000   0.7205815395
 H           1.0   0.0000000000   0.7565140024   0.1397092302
 $END
```

Radiative transition moment

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The radiative transition moment can be calculated by a RUNTYP=TRANSITN run. Keywords controlling this run are of the $TRANST group (which also pertains to spin-orbit coupling calculations).

Polarizabilities

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The use of RUNTYP=TDHF for the calculation of polarizabilities was already discussed. In addition, it should be mentioned that RUNTYP=FFIELD can be used to apply finite electric fields in order to extract polarizabilities. Such calculations can be controlled by keywords of the $FFCALC group. The FFIELD method is general, and so works for all ab initio SCFTYPs. There are some restrictions however: analytic Hessians are not available, but numerical hessians are. Because an external field causes a molecule with a dipole to experience a torque, geometry optimizations must be done in Cartesian coordinates only. Internal coordinates eliminate the rotational degrees of freedom, which are no longer free.

Notes: a Hessian calculation will have two rotational modes with non-zero "frequency", caused by the torque. A gas phase molecule will rotate so that the dipole moment is anti-parallel to the applied field. To carry out this rotation during geometry optimization will take many steps, and you can help save much time by inputting a field opposite the molecular dipole. There is also a stationary point at higher energy with the dipole parallel to the field, which will have two imaginary frequencies in the hessian. Careful, these will appear as the first two modes in a hessian run, but will not have the i for imaginary included on the printout since they are rotational modes.

Transition moments and spin-orbit coupling

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Firefly can compute transition moments and oscillator strengths for the radiative transition between two CI wavefunctions. The moments are computed using both the "length (dipole) form" and "velocity form". Firefly can also compute the one electron portion of the "microscopic Breit-Pauli spin orbit operator". The two electron terms can be approximately accounted for by means of effective nuclear charges.

The orbitals for the CI can be one common set of orbitals used by all CI states. If one set of orbitals is used, the transition moment or spin-orbit coupling can be found for any type of CI wavefunction Firefly can compute.

Alternatively, two sets of orbitals (obtained from separate MCSCF orbital optimizations) can be used. Two separate CIs will be carried out (in 1 run). The two MO sets must share a common set of frozen core orbitals, and the CI -must- be of the complete active space type. These restrictions are needed to leave the CI wavefunctions invariant under the necessary transformation to corresponding orbitals. The non-orthogonal procedure implemented in Firefly is a GUGA driven equivalent to the method of Lengsfield, et al. Note that the FOCI and SOCI methods described by these workers are not available in Firefly.

If you would like to use separate orbitals for the states, use the FCORE option in $MCSCF in a SCFTYP=MCSCF optimization of the orbitals. Typically you would optimize the ground state completely, and use these MCSCF orbitals in an optimization of the excited state, under the constraint of FCORE=.TRUE. The core orbitals of such a MCSCF optimization should be declared MCC, rather than FZC, even though they are frozen.

In the case of transition moments either one or two CI calculations are performed, necessarily on states of the same multiplicity. Thus, only a $CIDRT1 is read. A spinorbit coupling run almost always does two or more CI calculations, as the states to be coupled are usually of different multiplicities. So, spin-orbit runs might read only $CIDRT1, but normally read several, $CIDRT1, $CIDRT2, ... The first CI calculation, defined by $CIDRT1, must be for the state of lower spin multiplicity, with $CIDRT2, $CIDRT3, ... being successively higher multiplicities.

You will probably have to lower the symmetry in $CIDRT1 and $CIDRT2 to C1. You may use full spatial symmetry in the CIDRT groups only if the two states happen to have the same total spatial symmetry.

The spin-orbit operator contains a one electron term arising from the Pauli's reduction of the hydrogenic Dirac equation to one-component form, and a two electron term added by Breit. At present, the code for treating the full Breit-Pauli operator is limited to consideration of only singlet states with one triplet state. The active space for METHOD=BREIT is limited to 10 active orbitals on 32 bit machines and about 16 on 64 bit machines.

As an approximation, the nuclear charge appearing in the one electron term can be regarded as an empirical scale factor, compensating for the omission of the two electron operator. This is METHOD=ZEFF, and is general both to any number of active orbitals or spin muptiplicities. The values of ZEFF may be very different from the true nuclear charge if ECP basis sets are in use, see the two references mentioned below.

For Pauli-Breit runs you can several options control the form factors calculation. For large active spaces you may want to precalculate the form factors and save them to disk by using the ACTION option. In case if you do not provide enough storage for the form factors sorting then some extra disk space will be used; the extra disk space can be eliminated if you set SAVDSK=.TRUE. (the amount of savings depends on the active space and memory provided, it some cases it can decrease the disk space up to one order of magnitude). The form factors are in binary format, and so can be transfered between computers only if they have compatible binary files. There is a built-in check for consistency of a restart file DAFL30 with the current run parameters.

The transition moment and spin orbit coupling driver is a rather restricted path through Firefly, in that

1) Give SCFTYP=NONE. $GUESS is not read, as the program expects to MOREAD the orbitals $VEC1 group, and perhaps $VEC2,... groups. It is not possible to reorder MOs.

2) $CIINP is not read. The CI is hardwired to consist of CIDRT generation, integral transformation/sorting, Hamiltonian generation, and diagonalization. This means $CIDRT1 (and maybe $CIDRT2,...), $TRANS, $CISORT, $GUGEM, and $GUGDIA input is read, and acted upon.

3) The density matrices are not generated, and so no properties (other than the transition moment or the spin-orbit coupling) are computed.

4) There is no restart capability provided.

5) CIDRT input is given in $CIDRT1 and maybe $CIDRT2.

6) IROOTS will determine the number of CI states in each CI for which the properties are calculated. Use NSTATE to specify the number of CI states for the CI Hamiltonian diagonalisation. Sometimes the CI convergence is assisted by requesting more roots to be found in the diagonalization than you want to include in the property calculation.

7) The spin-orbit integrals permit the basis to be METHOD=Zeff: s,p,d,f but not l,g (l means sp) METHOD=Breit: s,p,d,l but not f,g

Reference for separate orbital optimization:

1. B.H.Lengsfield, III, J.A.Jafri, D.H.Phillips, C.W.Bauschlicher, Jr. J.Chem.Phys. 74,6849-6856(1981) References for transition moments:
2. F.Weinhold, J.Chem.Phys. 54,1874-1881(1970)
3. C.W.Bauschlicher, S.R.Langhoff Theoret.Chim.Acta 79:93-103(1991)
4. "Intramediate Quantum Mechanics, 3rd Ed." Hans A. Bethe, Roman Jackiw Benjamin/Cummings Publishing, Menlo Park, CA (1986), chapters 10 and 11.
5. S.Koseki, M.S.Gordon J.Mol.Spectrosc. 123, 392-404(1987)

References for Zeff spin-orbit coupling, and ZEFTYP values:

6. S.Koseki, M.W.Schmidt, M.S.Gordon J.Phys.Chem. 96, 10768-10772 (1992)
7. S.Koseki, M.S.Gordon, M.W.Schmidt, N.Matsunaga J.Phys.Chem. 99, 12764-12772 (1995)
8. N.Matsunaga, S.Koseki, M.S.Gordon J.Chem.Phys. 104, 7988-7996 (1996)
9. S.Koseki, M.W.Schmidt, M.S.Gordon J.Phys.Chem.A 102, 10430-10435 (1998)

References for full Breit-Pauli spin-orbit coupling:

10. T.R.Furlani, H.F.King J.Chem.Phys. 82, 5577-5583 (1985) 11. H.F.King, T.R.Furlani J.Comput.Chem. 9, 771-778 (1988)

Special thanks to Bob Cave and Dave Feller for their assistance in performing check spin-orbit coupling runs with the MELDF programs. Special thanks to Tom Furlani for contributing his 2e- spin-orbit code and answering many questions about its interface.

Here is an example. Note that you must know what you are doing with term symbols, J quantum numbers, point group symmetry, and so on in order to make skillful use of this part of the program.

```
! Compute the splitting of the famous sodium D line.
!
! The two SCF energies below give an excitation energy
! of 16,044 cm-1 to the 2-P term. The computed spin-orbit
! levels are at RELATIVE E=-10.269 and 5.135 cm-1, which
! means the 2-P level interval is 15.404 cm-1.
!
! Charlotte Moore's Atomic Energy Levels, volume 1, gives
! the experimental 2-P interval as 17.1963, the levels are
! at 2-S-1/2=0.0, 2-P-1/2=16,956.183, 2-P-3/2=16,973.379
!
```

1. generate ground state 2-S orbitals by conventional ROHF.
the energy of the ground state is -161.8413919816

```
--- $contrl scftyp=rohf mult=2 $end
--- $system kdiag=3 memory=300000 $end
--- $guess guess=huckel $end
--- $basis gbasis=n31 ngauss=6 $end
```

2. generate excited state 2-P orbitals, using a state-averaged SCF wavefunction to ensure radial degeneracy of the 3p shell is preserved. The open shell SCF energy is -161.7682895801. The computation is both spin and space restricted open shell SCF on the 2-P Russell-Saunders term. Starting orbitals are reordered orbitals from step 1.

```
--- $contrl scftyp=gvb mult=2 $end
--- $system kdiag=3 memory=300000 $end
--- $guess guess=moread norb=13 norder=1 iorder(6)=7,8,9,6 $end
--- $basis gbasis=n31 ngauss=6 $end
--- $scf nco=5 nseto=1 no(1)=3 rstrct=.true. couple=.true.
```

```
--- f(1)= 1.0 0.16666666666667
--- alpha(1)= 2.0 0.33333333333333 0.0
--- beta(1)= -1.0 -0.16666666666667 0.0 $end
```

3. compute spin orbit coupling in the 2-P term. The use of C1 symmetry in $CIDRT ensures that all three spatial CSFs are kept in the CI function. In the preliminary CI, the spin function is just the alpha spin doublet, and all three roots should be degenerate, and furthermore equal to the GVB energy at step 2. The spin-orbit coupling code uses both doublet spin functions with each of the three spatial wavefunctions, so the spin-orbit Hamiltonian is a 6x6 matrix. The two lowest roots of the full 6x6 spin-orbit Hamiltonian are the doubly degenerate 2-P-1/2 level, while the other four roots are the degenerate 2-P-3/2 level.

```
 $contrl scftyp=none cityp=guga runtyp=spinorbt mult=2 $end
 $system memory=500000 $end
 $gugdia nstate=3 $end
 $transt numvec=1 numci=1 nfzc=5 nocc=8 iroots=3 zeff=10.04 $end
 $cidrt1 group=c1 fors=.true. nfzc=5 nalp=1 nval=2 $end

 $data
Na atom...2-P excited state...6-31G basis, typed w/o L shells.
Dnh 2

Na 11.0
...basis information...

 $end

--- GVB ORBITALS --- GENERATED AT  7:46:08 CST 30-MAY-1996
Na atom...2-P excited state
E(GVB)=     -161.7682895801, 5 ITERS
 $VEC1
...orbitals from step 2 go here...
 $END
```

Stone's distributed multipole analysis

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Stone’s distributed multipole analysis is triggered by the presence of the $STONE group in the input. Keywords of this group provide control over the analysis.

Additional capabilities

Orbital localization

The following text was inherited from an old version of the Firefly manual. It might currently not be complete and/or contain outdated information, but will be improved in the future.

Three different orbital localization methods are implemented in Firefly. The energy and dipole based methods normally produce similar results, but see M.W.Schmidt, S.Yabushita, M.S.Gordon in J.Chem.Phys., 1984, 88, 382-389 for an interesting exception. You can find references to the three methods at the beginning of this chapter.

The method due to Edmiston and Ruedenberg works by maximizing the sum of the orbitals' two electron self-repulsion integrals. Most people who think about the different localization criteria end up concluding that this one seems superior. The method requires the two electron integrals, transformed into the molecular orbital basis. Because only the integrals involving the orbitals to be localized are needed, the integral transformation is actually not very time consuming.

The Boys method maximizes the sum of the distances between the orbital centroids, that is the difference in the orbital dipole moments.

The population method due to Pipek and Mezey maximizes a certain sum of gross atomic Mulliken populations. This procedure will not mix sigma and pi bonds, so you will not get localized banana bonds. Hence it is rather easy to find cases where this method give different results than the Ruedenberg or Boys approach.

Firefly will localize orbitals for any kind of RHF, UHF, ROHF, or MCSCF wavefunctions. The localizations will automatically restrict any rotation that would cause the energy of the wavefunction to be changed (the total wavefunction is left invariant). As discussed below, localizations for GVB or CI functions are not permitted.

The default is to freeze core orbitals. The localized valence orbitals are scarcely changed if the core orbitals are included, and it is usually convenient to leave them out. Therefore, the default localizations are: RHF functions localize all doubly occupied valence orbitals. UHF functions localize all valence alpha, and then all valence beta orbitals. ROHF functions localize all valence doubly occupied orbitals, and all singly occupied orbitals, but do not mix these two orbital spaces. MCSCF functions localize all valence MCC type orbitals, and localize all active orbitals, but do not mix these two orbital spaces. To recover the invariant MCSCF function, you must be using a FORS=.TRUE. wavefunction, and you must set GROUP=C1 in $DRT, since the localized orbitals possess no symmetry.

In general, GVB functions are invariant only to localizations of the NCO doubly occupied orbitals. Any pairs must be written in natural form, so pair orbitals cannot be localized. The open shells may be degenerate, so in general these should not be mixed. If for some reason you feel you must localize the doubly occupied space, do a RUNTYP=PROP job. Feed in the GVB orbitals, but tell the program it is SCFTYP=RHF, and enter a negative ICHARG so that Firefly thinks all orbitals occupied in the GVB are occupied in this fictitous RHF. Use NINA or NOUTA to localize the desired doubly occupied orbitals. Orbital localization is not permitted for CI, because we cannot imagine why you would want to do that anyway.

Boys localization of the core orbitals in molecules having elements from the third or higher row almost never succeeds. Boys localization including the core for second row atoms will often work, since there is only one inner shell on these. The Ruedenberg method should work for any element, although including core orbitals in the integral transformation is more expensive.

The easiest way to do localization is in the run which generates the wavefunction, by selecting LOCAL=xxx in the $CONTRL group. However, localization may be conveniently done at any time after determination of the wavefunction, by executing a RUNTYP=PROP job. This will require only $CONTRL, $BASIS/$DATA, $GUESS (pick MOREAD), the converged $VEC, possibly $SCF or $DRT to define your wavefunction, and optionally some $LOCAL input.

There is an option to restrict all rotations that would mix orbitals of different symmetries. SYMLOC=.TRUE. yields only partially localized orbitals, but these still possess symmetry. They are therefore very useful as starting orbitals for MCSCF or GVB-PP calculations. Because they still have symmetry, these partially localized orbitals run as efficiently as the canonical orbitals. Because it is much easier for a user to pick out the bonds which are to be correlated, a significant number of iterations can be saved, and convergence to false solutions is less likely.

The most important reason for localizing orbitals is to analyze the wavefunction. A simple example is to make contour plots of the resulting orbitals with the PLTORB graphics codes, or perhaps to read the localized orbitals in during a RUNTYP=PROP job to examine their Mulliken populations. MCSCF localized orbitals can be input to a CI calculation, to generate the corresponding density matrix, which contains much useful information about electron populations and chemical bonding. For example:

J.Am.Chem.Soc., 104, 960-967 (1982)
J.Am.Chem.Soc., 113, 5231-5243 (1991)
Theoret.Chim.Acta, 83, 57-68 (1992)

In addition, the energy of your molecule can be partitioned over the localized orbitals so that you may be able to understand the origin of barriers, etc. This analysis can be made for the SCF energy, and also the MP2 correction to the SCF energy, which requires two separate runs. An explanation of the method, and application to hydrogen bonding may be found in J.H.Jensen, M.S.Gordon, J.Phys.Chem. 1995, 99, 8091-8107.

Analysis of the SCF energy is based on the localized charge distribution (LCD) model: W.England and M.S.Gordon, J.Am.Chem.Soc. 93, 4649-4657 (1971). This is implemented for RHF and ROHF wavefunctions, and it requires use of the Ruedenberg localization method, since it needs the two electron integrals to correctly compute energy sums. All orbitals must be included in the localization, even the cores, so that the total energy is reproduced.

The LCD requires both electronic and nuclear charges to be partitioned. The orbital localization automatically accomplishes the former, but division of the nuclear charge may require some assistance from you. The program attempts to correctly partition the nuclear charge, if you select the MOIDON option, according to the following: a Mulliken type analysis of the localized orbitals is made. This determines if an orbital is a core, lone pair, or bonding MO. Two protons are assigned to the nucleus to which any core or lone pair belongs. One proton is assigned to each of the two nuclei in a bond. When all localized orbitals have been assigned in this manner, the total number of protons which have been assigned to each nucleus should equal the true nuclear charge.

Many interesting systems (three center bonds, backbonding, aromatic delocalization, and all charged species) may require you to assist the automatic assignment of nuclear charge. First, note that MOIDON reorders the localized orbitals into a consistent order: first comes any core and lone pair orbitals on the 1st atom, then any bonds from atom 1 to atoms 2, 3, ..., then any core and lone pairs on atom 2, then any bonds from atom 2 to 3, 4, ..., and so on. Let us consider a simple case where MOIDON fails, the ion NH4+. Assuming the nitrogen is the 1st atom, MOIDON generates

```
 NNUCMO=1,2,2,2,2
   MOIJ=1,1,1,1,1
          2,3,4,5
    ZIJ=2.0,1.0,1.0,1.0,1.0,
           1.0,1.0,1.0,1.0
```

The columns (which are LMOs) are allowed to span up to 5 rows (the nuclei), in situations with multicenter bonds. MOIJ shows the Mulliken analysis thinks there are four NH bonds following the nitrogen core. ZIJ shows that since each such bond assigns one proton to nitrogen, the total charge of N is +6. This is incorrect of course, as indeed will always happen to some nucleus in a charged molecule. In order for the energy analysis to correctly reproduce the total energy, we must ensure that the charge of nitrogen is +7. The least arbitrary way to do this is to increase the nitrogen charge assigned to each NH bond by 1/4. Since in our case NNUCMO and MOIJ and much of ZIJ are correct, we need only override a small part of them with $LOCAL input:

```
 IJMO(1)=1,2,  1,3,  1,4,  1,5
 ZIJ(1)=1.25, 1.25, 1.25, 1.25
```

which changes the charge of the first atom of orbitals 2 through 5 to 5/4, changing ZIJ to

```
  ZIJ=2.0,1.25,1.25,1.25,1.25,
         1.0, 1.0, 1.0, 1.0
```

The purpose of the IJMO sparse matrix pointer is to let you give only the changed parts of ZIJ and/or MOIJ.

Another way to resolve the problem with NH4+ is to change one of the 4 equivalent bond pairs into a "proton". A "proton" orbital AH treats the LMO as if it were a lone pair on A, and so assigns +2 to nucleus A. Use of a "proton" also generates an imaginary orbital, with zero electron occupancy. For example, if we make atom 2 in NH4+ a "proton", by

```
IPROT(1)=2
NNUCMO(2)=1
IJMO(1)=1,2,2,2   MOIJ(1)=1,0   ZIJ(1)=2.0,0.0
```

the automatic decomposition of the nuclear charges will be

```
NNUCMO=1,1,2,2,2,1
  MOIJ=1,1,1,1,1,2
           3,4,5
   ZIJ=2.0,2.0,1.0,1.0,1.0,1.0
               1.0,1.0,1.0
```

The 6th column is just a proton, and the decomposition will not give any electronic energy associated with this "orbital", since it is vacant. Note that the two ways we have disected the nuclear charges for NH4+ will both yield the correct total energy, but will give very different individual orbital components. Most people will feel that the first assignment is the least arbitrary, since it treats all four NH bonds equivalently.

However you assign the nuclear charges, you must ensure that the sum of all nuclear charges is correct. This is most easily verified by checking that the energy sum equals the total SCF energy of your system.

As another example, H3PO is studied in EXAM26.INP. Here the MOIDON analysis decides the three equivalent orbitals on oxygen are O lone pairs, assigning +2 to the oxygen nucleus for each orbital. This gives Z(O)=9, and Z(P)=14. The least arbitrary way to reduce Z(O) and increase Z(P) is to recognize that there is some backbonding in these "lone pairs" to P, and instead assign the nuclear charge of these three orbitals by 1/3 to P, 5/3 to O.

Because you may have to make several runs, looking carefully at the localized orbital output before the correct nuclear assignments are made, there is an option to skip directly to the decomposition when the orbital localization has already been done. Use

```
$CONTRL RUNTYP=PROP
$GUESS GUESS=MOREAD NORB=
$VEC containing the localized orbitals!
$TWOEI
```

The latter group contains the necessary Coulomb and exchange integrals, which are punched by the first localization, and permits the decomposition to begin immediately.

SCF level dipoles can also be analyzed using the DIPDCM flag in $LOCAL. The theory of the dipole analysis is given in the third paper of the LCD sequence. The following list includes application of the LCD analysis to many problems of chemical interest:

W.England, M.S.Gordon J.Am.Chem.Soc. 93, 4649-4657 (1971)
W.England, M.S.Gordon J.Am.Chem.Soc. 94, 4818-4823 (1972)
M.S.Gordon, W.England J.Am.Chem.Soc. 94, 5168-5178 (1972)
M.S.Gordon, W.England Chem.Phys.Lett. 15, 59-64 (1972)
M.S.Gordon, W.England J.Am.Chem.Soc. 95, 1753-1760 (1973)
M.S.Gordon J.Mol.Struct. 23, 399 (1974)
W.England, M.S.Gordon, K.Ruedenberg, Theoret.Chim.Acta 37, 177-216 (1975)
J.H.Jensen, M.S.Gordon, J.Phys.Chem. 99, 8091-8107 (1995)
J.H.Jensen, M.S.Gordon, J.Am.Chem.Soc. 117, 8159-8170 (1995)
M.S.Gordon, J.H.Jensen, Acc.Chem.Res. 29, 536-543 (1996)

It is also possible to analyze the MP2 correlation correction in terms of localized orbitals, for the RHF case. The method is that of G.Peterssen and M.L.Al-Laham, J.Chem.Phys., 94, 6081-6090 (1991). Any type of localized orbital may be used, and because the MP2 calculation typically omits cores, the $LOCAL group will normally include only valence orbitals in the localization. As mentioned already, the analysis of the MP2 correction must be done in a separate run from the SCF analysis, which must include cores in order to sum up to the total SCF energy.

Typically, the results are most easily interpreted by looking at "the bigger picture" than at "the details". Plots of kinetic and potential energy, normally as a function of some coordinate such as distance along an IRC, are the most revealing. Once you determine, for example, that the most significant contribution to the total energy is the kinetic energy, you may wish to look further into the minutia, such as the kinetic energies of individual localized orbitals, or groups of LMOs corresponding to an entire functional group.

Additional references:
Boys orbital localization-
S. F. Boys, "Quantum Science of Atoms, Molecules, and Solids" P. O. Lowdin, Ed, Academic Press, NY, 1966, 253-262.

Population orbital localization
J. Pipek, P. Z. Mezey J. Chem. Phys. 90, 4916(1989).

The NBO program

From the very beginning of its history, Firefly includes a fully-functional NBO module. However, the NBO part of the Firefly requires activation as it is a commercial code. As such, you have to purchase the Firefly/PC GAMESS NBO license codes and NBO manual from TCI/NBO if you would like to use the NBO module. The Firefly NBO license is very inexpensive ($ 30).

To activate the NBO module, you should add the following strings to your Firefly input:

```
 $LICENSE NBOLID=LID NBOKEY=KEY $END
 $NBO <NBO options> $END
```

Normally, you need to receive both NBO activation key (NBOKEY, 8-digit hexadecimal code), and NBO license ID (NBOLID, some decimal number) from TCI. Nevertheless, they sometimes forget to provide users by the NBOLID value. In this case, please contact us as we can recover your NBOLID using information on the NBO activation key by just looking it up in our NBO database.

The NBO code version incorporated into the current distributions of Firefly is NBO v.5.G (July 16th, 2008).

If you would like to post your input and output files on the Firefly forum (or share them with others in any other way), please remember to remove your NBO license data from them.

Morokuma energy decomposition

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

The Morokuma-Kitaura energy decomposition is an analysis of the energy contributions to dimerization. This type of calculation was already discussed shortly in the section on basis set superposition error (BSSE, see the chapter on basis sets). The $MOROKM group controls how the supermolecule input in the $DATA group is divided into two or more monomers. Both the supermolecule and its constituent monomers must be well described by RHF wavefunctions (avoid breaking chemical bonds!).

The present implementation has some quirks:

1. The initial guess of the monomer orbitals is not controlled by $GUESS. The program first looks for a $VEC1, $VEC2, ... group for each monomer. If they are found, they will be MOREAD. If any of these are missing, the guess for that monomer will be constructed by HCORE. Check your monomer energies carefully! The initial guess orbitals for the supermolecule are formed by a block diagonal matrix of the monomer orbitals.

2. The use of symmetry is turned off internally.

3. There is no direct SCF option. File ORDINT will be a full C1 list of integrals. File AOINTS will contain whatever subset of these is needed for each particular decomposition step. So extra disk space is needed compared to RUNTYP=ENERGY.

4. This kind of run applies only to ab initio cases.

5. This kind of run will work in parallel.

References:
C.Coulson in "Hydrogen Bonding", D.Hadzi, H.W.Thompson, Eds., Pergamon Press, NY, 1957, pp 339-360.

C.Coulson Research, 10, 149-159 (1957).

K.Morokuma J.Chem.Phys. 55, 1236-44 (1971).

K.Kitaura, K.Morokuma Int.J.Quantum Chem. 10, 325 (1976).

K.Morokuma, K.Kitaura in "Chemical Applications of Electrostatic Potentials", P.Politzer,D.G.Truhlar, Eds. Plenum Press, NY, 1981, pp 215-242.

The method coded is the newer version described in the latter two papers. Note that the CT term is computed separately for each monomer, as described in the words below equation 16 of the 1981 paper, not simultaneously.

Reduced Variational Space:
W.J.Stevens, W.H.Fink, Chem.Phys.Lett. 139, 15-22(1987).

A comparison of the RVS and Morokuma decompositions can be found in the review article:
"Wavefunctions and Chemical Bonding" M.S.Gordon, J.H.Jensen in "Encyclopedia of Computational Chemistry", volume 5, P.V.R.Schleyer, editor, John Wiley and Sons, Chichester, 1998.

The present implementation:
"Energy decomposition analysis for many-body interactions, and application to water complexes" W.Chen, M.S.Gordon J.Phys.Chem. 100, 14316-14328(1996)

Interfacing with other programs

The following text is a 'stub' and contains only minimal information. It will be expanded in the future.

Firefly is capable of interfacing with a number of external program. Below is an overview:

MOLPLT – the MOLPLT keyword (.T. or .F.) in $CONTRL can be used to produce an input deck for a molecule drawing program that is distributed with GAMESS (US)

PLTORB – the PLTORB keyword (.T. or .F.) in $CONTRL can be used to produce an input deck for a molecule drawing program that is distributed with GAMESS (US)

AIMPAC – interfacing met AIMPAC is possible through the AIMPAC keyword in $CONTRL. Possible values are 0 (do not produce AIMPAC input), 1 (produce standard AIMPAC input), and 2 (special AIMPAC input: for UHF wavefunctions, separate data is printed for alpha and beta orbitals rather than unified data for UHF natural orbitals).

The PLTORB, MOLPLT, and AIMPAC decks are written to file PUNCH at the end of the job. PLTORB and MOLPLT are written even for EXETYP=CHECK. AIMPAC requires at least RUNTYP=PROP.

In addition, the FRIEND keyword in $CONTRL is able to prepare input for HONDO 8.2, MELDF, GAMESS-UK, and Gaussian 9x. Selecting FRIEND turns the job into a CHECK run only, no matter how you set EXETYP. Thus, the geometry is that encountered in $DATA. The input is added to the PUNCH file, and may require some (usually minimal) modification.

Firefly 8.0.0 keyword list

Version: 2013-11-11

This list of keywords is currently a work in progress. At the moment, the following keyword groups have been fully documented:

$CONTRL
$MOORTH
$GUESS
$BASIS
$D5
$ZMAT
$LIBE
$SYSTEM
$SMP
$NUMGRD
$RAMAN
$CONIC
$FMM
$SCF
$DFT
$DFTD
$MP2
$MP2GRD
$MCSCF
$MCAVER
$TRACK
$XMCQDPT and $MCQDPT
$MCQFIT

Documentation for all other keyword groups is currently under review and not yet included in this document. Instead, the (incomplete) descriptions for these groups can be found in the “old” Firefly manual (Firefly_input_rev002.pdf) which already covers > 95 % of all functionality in Firefly. Some additional keyword descriptions not present in the old manual can be found on the Firefly discussion forums.

Overview of groups

Name	Function

Global settings, calculation type, molecule, basis set, wavefunction, initial guess, external field specification:

$CONTRL	chemical control data
$MOORTH	molecular orbital orthogonalization control
$GUESS	initial orbital selection
$BASIS	basis set
$EXTRAF	basis set projection control
$AODEL	basis set reduction control
$D5	pure spherical functions control
$ECP	effective core potentials
$EFIELD	external electric field
$DATA	molecule, basis set
$ZMAT	coded z-matrix
$LIBE	linear bend data
$MASS	isotope selection

Hardware-related specification, control and tuning:

$SYSTEM	computer related control data
$SMP	SMP related control data
$CUDA	CUDA-related control data
$MPI	MPI-related control data
$P2P	P2P interface control data
$LP2P	local P2P interfaces control data
$MMM	manual performance tuning and instrumentation for matrix operations

Additional details on calculation type:

$DRC	dynamic reaction path
$FFCALC	finite field polarizabilities
$FORCE	Hessian, normal coordinates
$GRADEX	gradient extremal method
$IRC	intrinsic reaction path
$NUMGRD	numerical gradient control
$MOROKM	Morokuma energy decomposition
$RAMAN	Raman calculation options
$STATPT	geometry search control
$CONIC	conical intersection search control
$SURF	potential surface scan
$TRANST	transition moments, spin-orbit

$TRUDGE	non-gradient optimization

Details on two-electron integrals computation, four-index transformation, and Coupled-Perturbed Hartree-Fock equation solvers:

$INTGRL	format for two-electron integrals
$FMM	quantum fast multipole method
$ECPINP	parameters controlling the details of effective core potentials computations
$NUMINT	control numerical quadratures used to compute integrals for HLS's inequality
$TRANS	integral transformation
$CPHF	Coupled-Perturbed Hartree-Fock options

Semiempirical, HF and DFT SCF-related options:

$MOPAC	control over details of semi-empirical code
$SCF	HF-SCF and GVB wavefunction control
$SCFMI	SCF-MI input control
$DFT	DFT control
$DFTD	empirical dispersion correction scheme control

Møller-Plesset perturbation theory specific options:

$MP2	2nd order Møller-Plesset perturbation theory
$MP2GRD	MP2 gradient control
$MP3	3rd order Møller-Plesset perturbation theory
$MP4	4th order Møller-Plesset perturbation theory

CI Singles, TDHF, and TDDFT options:

$CIS	CI Singles control data
$TDHF	time-dependent HF (for excitations) control data
$TDDFT	time-dependent DFT (for excitations) control data

GUGA CI specific options:

$DRT	distinct row table for MCSCF
$CIDRT	distinct row table for CI
$CIDRT1-$CIDRT20	used as input for computation of transition moments and spin-orbit couplings
$CIINP	control of GUGA CI process
$CISORT	integral sorting
$GUGEM	Hamiltonian matrix formation
$GUGDIA	Hamiltonian eigenvalues/vectors
$GUGDM	one-electron density matrix

$GUGDM2	two-electron density matrix
$TRFDM2	two-electron density back transformation
$LAGRAN	CI lagrangian matrix

Determinant-based CI options:

$DET	determinantal full CI for MCSCF
$CIDET	determinantal full CI

MCSCF-related options:

$MCSCF	parameters for MCSCF
$MCAVER	parameters for state-averaged MSCSF
$SOSCF	parameter tuning for SOSCF (HF/DFT/GVB and MCSCF) convergers
$TRACK	parameters for MCSCF state-tracking

Options specific to multistate multireference perturbation theories:

$XMCQDPT	extended multi-configurational quasi-degenerate perturbation theory
$MCQDPT	multi-configurational quasi-degenerate perturbation theory
$MCQFIT	resolvent fitting (X)MCQDPT control

Orbital localization options:

$LOCAL	orbital localization control

Solvation models: self-consistent reaction field:

$SCRF	self-consistent reaction field

Solvation models: polarizable continuum model:

$PCM	polarizable continuum model
$PCMCAV	PCM cavity generation
$PCMRAD	fine control over PCM radii
$DISBS	PCM dispersion basis set
$DISREP	PCM dispersion/repulsion
$NEWCAV	PCM escaped charge cavity

Solvation models: effective fragments:

$EFRAG	effective fragment potentials
$"FRAGNAME"	specific named fragment potentials
$FRGRPL	inter-fragment repulsion

Options controlling computation of various molecular properties, population analysis, and preparation of data for orbital/density visualization:

$TDHF	time-dependent HF and DFT NLO properties
$ELDENS	electron density
$ELFLDG	electric field/gradient
$ELMOM	electrostatic moments
$ELPOT	electrostatic potential
$PPA	nonlinear pair population analysis control
$CISPRP	CIS properties
$PDC	MEP fitting mesh
$VDWRAD	control over van-der-Waals radii used in MEP fitting
$CUBE	control over Firefly's CUBE feature
$GRID	property calculation mesh
$POINTS	property calculation points
$STONE	distributed multipole analysis
$MOLGRF	orbital plots
$NBO	NBO control data
$LICENSE	NBO license data

Formatted input groups providing various restart capabilities:

$VEC	orbitals, commonly used specification of starting MOs
$VEC1/$VEC2	orbitals, used for transition moment/spin-orbit coupling calculations
$VEC1-$VEC9	orbitals, provide initial guess for Morokuma decomposition runs
$TDVEC	coefficients of TD excitations and de-excitations
$CISVEC	coefficients of CIS excitations and de-excitations
$HESS	force constant matrix
$HESS2	force constant matrix. Required for some types of DRC runs
$GRAD	gradient vector
$VIB	RUNTYP=HESSIAN and RAMAN restart data
$DIPDR	dipole derivative matrix
$ALPDR	alpha derivative tensor
$ENERG	restart data for Morokuma decomposition runs
$TWOEI	two-electron integrals
$RSTART	restart data for $STATPT geometry searches
$TRURST	restart data for RUNTYP=TRUDGE

$CONTRL group

This is a free format group specifying global switches. SCFTYP together with MPLEVL or CITYP specifies the wavefunction.

SCFTYP	Chooses a wavefunction type.	
	RHF	Restricted Hartree-Fock calculation. (Default)
	UHF	Unrestricted Hartree-Fock calculation
	ROHF	Restricted open shell Hartree-Fock. (high spin, see GVB for low spin)
	GVB	Generalized valence bond wavefunction a.k.a. low spin a.k.a. OCBSE (orthogonality-constrained basis set expansion) type ROHF. (needs $SCF input)
	MCSCF	Multiconfigurational SCF wavefunction (this requires $DET or $DRT input)
	NONE	Indicates a single point computation, reading a converged SCF function. This option requires that you select CITYP=GUGA or ALDET, RUNTYP=ENERGY, TRANSITN, or SPINORBT, and GUESS=MOREAD.

MPLEVL	Chooses Møller-Plesset perturbation theory level, after the SCF. See $MP2, $MP3, $MP4, $MCQDPT and $XMCQDPT input groups.	
	0	Skips the MP computation. (Default)
	2	Performs a second order energy correction. MP2 is implemented only for RHF, UHF, ROHF, TCSCF and MCSCF wave functions. Gradients are available only for RHF, so for the others you may pick from RUNTYP=ENERGY, TRUDGE, SURFACE, or FFIELD only, or use numerical gradients.
	3	Performs a third (MP3) order energy correction. Implemented for SCFTYP=RHF only. See $MP3 group.
	4	Performs a fourth (MP4) order energy correction. Implemented for SCFTYP=RHF only. See $MP4 group.

CITYP	Chooses a CI computation after the SCF, for any SCFTYP except UHF.	
	NONE	No CI calculation. (Default)
	CIS	Perform CI Singles (also known as Tamm-Dancoff Approximation) i.e. Single excitations CI from a SCFTYP=RHF reference, only. This is for excited states, with analytic nuclear gradients available. See the $CIS input group.
	TDHF	Perform Time-dependent Hartree-Fock calculation (while TDHF is rather similar to CIS it is not actually a CI!) for excited states. Available for SCFTYP=RHF only. Analytic gradient is not programmed yet. See the $TDHF input group.
	TDDFT	Perform Time-dependent DFT or Tamm-Dancoff Approximation to TDDFT calculations (note these are not a CI as well!). Available for SCFTYP=RHF only. DFTTYP must be set to one of the supported functionals. Analytic gradients are

		not programmed yet. See also DFTTYP in $CONTRL and the $TDDFT input groups.
	GUGA	Runs the Unitary Group CI package, which requires $CIDRT input. Analytic gradients are available only for RHF, so for other SCFTYPs, you may choose only RUNTYP=ENERGY, TRUDGE, SURFACE, FFIELD, TRANSITN, or SPINORBT, or use numerical gradients.
	ALDET	Runs the Ames Laboratory determinant full CI package, requiring $CIDET input. RUNTYP=ENERGY only. Analytic gradients are not so you may choose only RUNTYP=ENERGY, TRUDGE, SURFACE, FFIELD, or use numerical gradients.

Note: At most one of MPLEVL or CITYP may be chosen.

DFTTYP	Controls DFT functional. Should be explicitly given to activate DFT code. See the chapter on DFT for the complete list of functionals. Default is DFTTYP=NONE i.e. no DFT calculations.

DFTD	Flag that provides control over the use of Grimme's empirical dispersion correction scheme. See the $DFTD group for details.	
	.TRUE.	Use Grimme's empirical dispersion correction.
	.FALSE.	No empirical dispersion correction. (Default)

RUNTYP	Specifies the type of computation.	
	ENERGY	Molecular energy at a single geometry point. (Default)
	GRADIENT	Molecular energy plus gradient at a single geometry point.
	HESSIAN	Molecular energy plus gradient plus second derivatives at a single geometry point, including harmonic vibrational analysis. See the $FORCE and $CPHF input groups.
	OPTIMIZE	Optimize the molecular geometry using analytic or numeric energy gradients. See $STATPT.
	TRUDGE	Non-gradient total energy minimization. See groups $TRUDGE and $TRURST.
	SADPOINT	Locate saddle point (transition state). See the $STATPT group.
	IRC	Follow intrinsic reaction coordinate. See the $IRC group.
	GRADEXTR	Trace gradient extremals. See the $GRADEX group.
	DRC	Follow dynamic reaction coordinate. See the $DRC group.
	SURFACE	Scan fragment of the potential energy surface. See $SURF.
	RSURFACE	Relaxed scan of the potential energy surface. See $SURF.
	RSURFX	Advanced relaxed scan of the potential energy surface. See $SURF.

	PROP	Calculation of properties for previously computed Hartree-Fock-type wavefunction. A $DATA deck and converged $VEC group should be input. Optionally, orbital localization can be done. See $ELPOT, etc.
	MOROKUMA	Performs monomer energy decomposition. See the $MOROKM group.
	TRANSITN	Compute radiative transition moment. See the $TRANST group.
	SPINORBT	Compute spin-orbit coupling. See the $TRANST group.
	FFIELD	Applies finite electric fields, most commonly to extract polarizabilities. See the $FFCALC group.
	TDHF	Analytic computation of frequency-dependent polarizabilities. See the $TDHF group.
	RAMAN	Computes Raman activities, see $RAMAN.

Note: RUNTYPs involving the energy gradient are: GRADIENT, HESSIAN, OPTIMIZE, SADPOINT, IRC, GRADEXTR, DRC and RAMAN, cannot be used with analytic gradients for any CI or MP2 computation, except when SCFTYP=RHF.

NUMDER	Flag that enables/disables the use of numerical derivatives.	
	.TRUE.	Use of numerical derivatives.
	.FALSE.	Do not use numerical derivatives. (Default)

EXETYP	Sets the execution type.	
	RUN	Actually do the run. (Default)
	CHECK	Wavefunction and energy will not be evaluated. This lets you speedily check input and memory requirements.
	DEBUG	Massive amounts of output are printed, useful only if you hate trees.
	ROUTINE	Maximum output is generated by the routine named. Please contact developers for the correct name of routine of interest.

MAXIT	N	Maximum number of SCF iteration cycles. If given, it pertains to RHF, UHF, ROHF, GVB, and MCSCF runs. If not given, a default value of 30 is used for RHF, UHF, ROHF, and GVB. For MCSCF, the number of iterations depends on the value of MAXIT in $MCSCF (which overrules MAXIT in $CONTRL).

ICHARG	N	Molecular charge. (Default is 0)

MULT	N	Multiplicity of the electronic state. 1 = singlet, 2 = doublet, 3 = triplet, etc. (Default is 1)

Note: ICHARG and MULT are used directly for RHF, UHF, ROHF. For GVB, these are implicit in the $SCF input, while for MCSCF or CI, these are implicit in $DRT/$CIDRT or $DET/$CIDET input. You must however still give them correctly.

ECP	Effective potential control.	
	NONE	All electron calculation. (Default)
	READ	Read in the potentials from the $ECP group.
	SBKJC	Use Stevens, Basch, Krauss, Jasien, Cundari potentials for all heavy atoms (Li-Rn are available).
	HW	Use Hay, Wadt potentials for all the heavy atoms (Na-Xe are available).

OLDECP	Flag that selects the code used for ECP integral evaluation.	
	.TRUE.	Use the old code for ECP integral evaluation. The old code is slower, and does not allow high-angular value ECPs and basis functions but is more numerically stable.
	.FALSE.	Use the new code for ECP integral evaluation (default)

The next three keywords control molecular geometry:

COORD	Choice for molecular geometry in $DATA.	
	UNIQUE	Only the symmetry unique atoms will be given, in Cartesian coordinates (default).
	HINT	Only the symmetry unique atoms will be given, in Hilderbrandt style internals.
	CART	Cartesian coordinates will be input.
	ZMT	GAUSSIAN style internals will be input.
	ZMTMPC	MOPAC style internals will be input.
	FRAGONLY	Means no part of the system is treated by *ab initio* means, hence $DATA is not given. The system is specified by $EFRAG.

UNITS	Distance units, any angles must be in degrees.	
	ANGS	Angstroms (Default)
	BOHR	Bohr atomic units

NZVAR	M	Control over Z-matrix NZVAR refers mainly to the coordinates used by OPTIMIZE or SADPOINT runs, but may also print the internal coordinate values for other run types. You can use internals to define the molecule, but Cartesians during optimizations!

	0	Use Cartesian coordinates. (Default).
	M > 0	If COORD does not equal ZMT or ZMTMPC and a $ZMAT is not given: the delocalized internal coordinates (DLCs) will be generated automatically.
	M > 0	If COORD=ZMT or ZMTMPC and a $ZMAT is not given: the internal coordinates will be those defining the molecule in $DATA. In this case, $DATA must not contain any dummy atoms. M is usually 3N–6, or 3N–5 for linear.
	M > 0	For other COORD choices, or if $ZMAT is given: the internal coordinates will be those defined in $ZMAT. This allows more sophisticated internal coordinate choices. M is ordinarily 3N–6 (3N–5), unless $ZMAT has linear bends.

LOCAL	Controls orbital localization. See the $LOCAL group. Localization does not work for SCFTYP=GVB, CITYP, or any MP.	
	NONE	Skip localization. (Default).
	BOYS	Do Foster-Boys localization.
	RUEDNBRG	Do Edmiston-Ruedenberg localization.
	POP	Do Pipek-Mezey population localization.

D5	Flag that enables/disables the use of a spherical basis set.	
	.TRUE.	Use pure spherical basis sets. Note that the current implementation of D5 option is not compatible with non-standard molecular input frames (custom orientations of axes). See the description of the relevant $D5 input group.
	.FALSE.	Use Cartesian basis sets. (Default)

MOLPLT	Allows Firefly to interface with a molecule drawing program (MOLPLT) distributed with GAMESS (US).	
	.TRUE.	Produce an input deck for MOLTPLT.
	.FALSE.	Do not produce an input deck for MOLTPLT. (Default)

PLTORB	Allows Firefly to interface with an orbital plotting program (PLTORB) distributed with GAMESS (US).	
	.TRUE.	Produce an input deck for PLTORB.
	.FALSE.	Do not produce an input deck for PLTORB. (Default)

AIMPAC	Allows Firefly to interface with Bader's atoms in molecules properties code (AIMPAC). For information about this program, contact:

	Richard F.W. Bader Dept. of Chemistry McMaster University Hamilton, Ontario L8S-4M1 Canada bader@sscvax.cis.mcmaster.ca	
	.TRUE.	Produce an input deck for AIMPAC.
	.FALSE.	Do not produce an input deck for AIMPAC. (Default)

RPAC	Flag to create the input files for Bouman and Hansen's RPAC electronic excitation and NMR shieldings program. RPAC works only with RHF wavefunctions. (This option is inactive.)	
	.TRUE.	Produce an input deck for RPAC.
	.FALSE.	Do not produce an input deck for RPAC. (Default)

FRIEND	String to prepare input for other quantum chemistry programs. If given, the corresponding input file will be created and punched.	
	No value	Do not prepare input for other QC programs. (Default)
	HONDO	Prepare input for HONDO 8.2.
	MELDF	Prepare input for MELDF.
	GAMESSUK	Prepare input for GAMESS (UK Daresbury version).
	GAUSSIAN	Prepare input for Gaussian 9X.
	ALL	Prepare input for all of the above.

Note: PLTORB, MOLPLT, and AIMPAC decks are written to file PUNCH at the end of the job. The two binary disk files output by RPAC are written at the end of the job. Thus all of these correspond to the final geometry encountered during the job. In contrast, selecting FRIEND turns the job into a CHECK run only, no matter how you set EXETYP. Thus the geometry is that encountered in $DATA. The input is added to the PUNCH file, and may require some (usually minimal) massaging. PLTORB and MOLPLT are written even for EXETYP=CHECK. AIMPAC requires at least RUNTYP=PROP. RPAC requires at least RUNTYP=ENERGY and you must take action to save the binary files AOINTS and WORK15.

NPRINT	Print/punch control flag. See also EXETYP for debug info. Options -7 to 5 are primarily for debugging.	
	-7	Extra printing from Boys localization.
	-6	Debug for geometry searches.
	-5	Minimal output.
	-4	Print 2e-contribution to gradient.
	-3	Print 1e-contribution to gradient.
	-2	Normal printing, no punch file.

	1	Extra printing for basis, symmetry, ZMAT.
	2	Extra printing for MO guess routines.
	3	Print out property and 1e- integrals.
	4	Print out 2e- integrals.
	5	Print out SCF data for each cycle. (Fock and density matrices, current MOs)
	6	Same as 7, but wider 132 columns output. This option isn't perfect.
	7	Normal printing and punching (Default)
	8	More printout than 7. The extra output is (AO) Mulliken and overlap population analysis, eigenvalues, Lagrangians, etc...
	9	Everything in 8 plus Löwdin population analysis, final density matrix.

NOSYM	Controls the use of symmetry	
	0	The symmetry specified in $DATA is used as much as possible in integrals, SCF, gradients, etc. (Default)
	1	The symmetry specified in the $DATA group is used to build the molecule, then symmetry is not used again. Some runs require you request no symmetry, see the symmetry section in the chapter on coordinate types of the manual for more details.

INTTYP	Provides control over two-electron integral code.	
	POPLE	Use fast Pople-Hehre routines for sp integral blocks, and HONDO Rys polynomial code for all other integrals. (Default).
	HONDO	Use HONDO/Rys polynomial package for all integrals. This option produces slightly more accurate integrals but is also slower.

FSTINT	Flag that provides control over the use of the fastints direct SCF code.	
	.TRUE.	Enables the use of fastints. (Default).
	.FALSE.	Disables the use of fastints.

REORDR	Flag that controls reordering of shells for even better direct SCF performance.	
	.TRUE.	Enable reordering of shells. (Default)
	.FALSE.	Disable reordering of shells.

GENCON	Flag that enables/disables the use of a special version of the fastints code designed for general contraction (GC) type basis sets. It is mainly intended to dramatically speedup calculations involving large GC-type basis sets like the ANO basis sets by Roos *et al* (the

	example of pure GC basis sets), and to some degree the cc-pVXZ basis sets (which are only partially of the GC type), and many others. The code is very efficient, but requires some additional amount of memory and has a minor addition computational overhead for setup. It can result in slightly different energies compared to those obtained with the standard fastints code using the same value of ICUT and ITOL parameters. Also, it does not at all improve performance for pure segmented contraction basis sets. This is why the gencon code automatically disables itself if the basis set is not of GC type. GENCON has no effect on QFMM calculations.	
	.TRUE.	Enable the use of the special version of the fastints code.
	.FALSE.	Disable the use of the special version of the fastints code

NORMF	Controls normalization of the basis functions.	
	0	Normalize the basis functions. (Default)
	1	No normalization.

NORMP	Controls whether or not input contraction coefficients refer to normalized Gaussian primitives.	
	0	Input contraction coefficients refer to normalized Gaussian primitives. (Default)
	1	Input contraction coefficients do not refer to normalized Gaussian primitives.

ICUT	N	Cutoff for two-electron integrals. In conventional mode, integrals less than 10^{-ICUT} are not stored. In direct mode, 10^{-ICUT} is the cutoff for Schwarz inequality screening. (Default is 9)

ITOL	N	Primitive cutoff factor. Products of primitives whose exponential factor is less than 10^{-ITOL} are skipped. (Default is 20)

LEXCUT	N	Secondary cutoff for two-electron integrals and for Schwarz inequality screening. Affects only the direct exchange operator formation fastints/gencon code.
	1	A cutoff of $10^{-ICUT-N}$ is used. (Default)

The following keywords are restart options:

IREST	Restart control options (for OPTIMIZE run restarts, see $STATPT). Note these options should not normally be used.	
	-1	Reuse dictionary file from previous run, useful with GEOM=DAF and/or GUESS=MOSAVED. Otherwise, this option is the same as 0.

	0	Normal run. (Default)
	1	2e restart (1–e integrals and MOs saved in the old DICTNRY).
	2	SCF restart (1–, 2–e integrals and MOs saved). Restart with the AOINTS file only 1–e integrals will be recomputed if the old DICTNRY file does not exist.
	3	1e gradient restart.
	4	2e gradient restart.

GEOM	Selects where to obtain the molecular geometry.	
	INPUT	Obtain the geometry from the $DATA input. (Default for IREST=0)
	DAF	Read the geometry from the DICTNRY file. (Default for `IREST≠0`)

WIDE	Flag that provides control over formatted orbitals punchout.	
	.TRUE.	Use wide format for vectors in $VEC groups for both INPUT and PUNCH files. Should be used for post-SCF computations and to get better precision in SCF when reading in orbitals.
	.FALSE.	Default format. (Default)

$MOORTH group

This group controls various symmetry- and orthogonality-related checks and procedures.

SYMS	Flag that controls additional explicit symmetrization of the AO overlap matrix.	
	.TRUE.	Perform an extra explicit symmetrization of the overlap matrix.
	.FALSE.	Do not perform an extra explicit symmetrization of the overlap matrix. (Default)

SYMDEN	Flag that controls explicit symmetrization of the density matrix during the SCF procedure. This affects RHF, ROHF, and UHF wavefunctions (Hartree-Fock as well as DFT) only.	
	.TRUE.	Perform explicit symmetrization of the density matrix during SCF procedure.
	.FALSE.	Do not perform explicit symmetrization of the density matrix during SCF procedure. (Default)

SYMVEC	Flag that controls explicit symmetrization of the MOs after the SCF procedure and/or before the integral transformation stage(s).	
	.TRUE.	Perform this explicit symmetrization of the MOs.
	.FALSE.	Do not perform explicit symmetrization of MOs. (Default)

SYMVX	Flag that controls an extra symmetrization of the initial MOs for subsequent SCF procedures. Use of this symmetrization is enforced when SCFTYP=MCSCF.	
	.TRUE.	Perform an extra symmetrization of MOs at the beginning of each of subsequent SCF procedures. (Enforced for SCFTYP=MCSCF)
	.FALSE.	Do not perform an extra symmetrization of MOs at the beginning of SCF procedures (Default for SCFTYP≠MCSCF)

TOLSYM	N	Maximum permissible symmetry contamination allowed when determining symmetry of the MOs for various integral transformation stages. (Default is 1.0D-10)

TOLE	N	The threshold below which MO coefficients will be forcibly equated. This is a relative level; coefficients are set equal if one agrees in magnitude to TOLE multiplied with the other. A value of 0.0 disables equating of MO coefficients. (Default is 0.0)

TOLZ	N	The threshold below which MO coefficients will be zeroed forcibly. Coefficients are zeroed if they are below TOLZ in magnitude. A value of 0.0 effectively disables zeroing of MO coefficients. (Default is 0.0)

NOZERO	Controls the zeroing of various computed intermediate quantities. Small quantities are zeroed forcibly if they fall below the dedicated cutoffs (controlled with TOLE and TOLZ, see above) in magnitude.	
	.TRUE.	Disable forcible zeroing. (Default)
	.FALSE.	Enable forcible zeroing.

RCRIT	N	This option controls the pruning of density matrices during QFMM SCF computations. Matrix elements of the density matrices will be zeroed if the distance (in Bohr) between two orbital centers is greater than RCRIT. This option can speed up computation of the linear exchange (LEX, see documentation on QFMM), but should be used with caution, especially for conjugated systems, metal clusters, etc. For non-conjugated systems, RCRIT=25 seems to be safe enough. A value of 0.0 disables pruning. (Default is 0.0)

NOSTF	Controls the enforcing of standard phases of MOs and CI vectors by multiplying them by -1.0 when needed. A standard phase corresponds to the positive sign of the maximum in the magnitude element of a vector.	
	.TRUE.	Do not enforce standard phase on vectors. (Default)
	.FALSE.	Enforce standard phase on vectors as appropriate.

The remaining keywords in this group are obsolete and should not be used.

SORTH	.FALSE.	The default option.

SOONLY	N	(Default is 0)

NFIRST	N	(Default is 0)

THRX	N	(Default is 1.0D-10)

THRD	N	(Default is 1.0D-10)

THRS	N	(Default is 1.0D-10)

$GUESS group

This group controls the selection of initial molecular orbitals.

GUESS	Selects type of initial orbital guess. All GUESS types except SKIP permit reordering of the orbitals, carry out an orthonormalization of the orbitals (unless disabled with the ASIS keyword), optional orbital purification, and generate the correct initial density matrix. The initial density matrix cannot be generated for CI and MCSCF runs, so property restarts for these wavefunctions are disabled. Note that the correct computation of a GVB density matrix requires CICOEF in $SCF. A possible use for SKIP is to speed up a EXETYP=CHECK job, or a RUNTYP=HESSIAN job where the Hessian is supplied.	
	HUCKEL	Carry out an extended Hückel calculation using a Huzinaga MINI basis set for all atoms with no ECPs and a HW basis set for all atoms with ECPs if the ECP core count is the same, and project this onto the current basis. This is implemented for atoms up to Rn, and will work for any all electron or ECP basis set. (Default for most runs)
	HCORE	Diagonalize the one electron Hamiltonian to obtain the initial guess orbitals. This method is applicable to any basis set. In general, it does not work as well as HUCKEL but it might give better results for systems containing transition metals. This is because HUCKEL does not always treat ECPs on transition metals properly.
	MOREAD	Read in formatted vectors punched by an earlier run. This requires a $VEC group, and you MUST pay attention to the NORB keyword below.
	RDMINI	Read in the $VEC deck from a converged calculation that used GBASIS=MINI without any polarization functions, and project these orbitals onto the current basis.
	MOSAVED	The initial orbitals are read from the DICTNRY file of the earlier run. Note, one needs to specify IREST=1 in $CONTRL for this feature to work correctly! (Default for restarts).
	SKIP	Bypass initial orbital selection. The initial orbitals and density matrix are assumed to be in the DICTNRY file. This GUESS assumes IREST=1.

ASIS	Flag that controls orthonormalization of the initial guess orbitals.	
	.TRUE.	Use the orbitals ‘as is’, *i.e.* do not orthonormalize the initial guess orbitals.
	.FALSE.	Orthonormalize the initial guess orbitals. (Default)

PRTMO	Flag that controls printing of the initial guess orbitals.	
	.TRUE.	Print the initial guess orbitals.
	.FALSE.	Do not print the initial guess orbitals. (Default)

PUNMO	Flag that controls punching of the initial guess orbitals.	
	.TRUE.	Punch the initial guess orbitals.
	.FALSE.	Do not punch the initial guess orbitals. (Default)

NORB	N	The number of orbitals to be read in the $VEC group. This applies only to GUESS=MOREAD. For RHF, UHF, ROHF, and GVB wavefunctions, NORB defaults to the number of occupied orbitals. NORB must be given for CI and MCSCF wavefunctions. For UHF wavefunctions, if NORB is not given, only the occupied alpha and beta orbitals should be given, back to back. Otherwise, both alpha and beta orbitals must consist of NORB vectors. NORB may be larger than the number of occupied MOs, if you wish to read in the virtual orbitals. If NORB is less than the number of atomic orbitals, the remaining orbitals are generated as the orthogonal complement to those read.

EXTRA	Flag that enables/disables input basis set modification. This applies only to GUESS=MOREAD. This keyword can be used together with the DELETE keyword. If both are given, the deletion is performed first.	
	.TRUE.	The $VEC deck read in is to be expanded to a larger basis. The $EXTRAF group should be used to provide expansion instructions.
	.FALSE.	No projection. (Default)

DELETE	Flag that enables/disables input basis set modification. This applies only to GUESS=MOREAD. This keyword can be used together with the EXTRA keyword. If both are given, the deletion is performed first.	
	.TRUE.	The $VEC deck read in is to be trimmed to a smaller basis. The $AODEL group should be used to provide deletion instructions.
	.FALSE.	No projection. (Default)

NORDER	Orbital reordering switch.	
	0	Do not reorder orbitals.
	1	Reorder orbitals using according to instructions provided with the IORDER and JORDER keywords.

IORDER	N1,N2,…	Array of reordering instructions. Input to this array gives the new molecular orbital order. For example, IORDER(3)=4,3 will interchange orbitals 3 and 4, while leaving the other MOs in the original order. A minus sign can be used as shorthand to swap the positions of two MOs, *i.e.* IORDER(6)=-9 will swap MOs 6 and 9. IORDER applies to all orbitals (alpha and beta) except for UHF

		wavefunctions, where it only affects the alpha MOs. (Default is IORDER(i)=i)

JORDER	N1,N2, …	Functions the same as IORDER, but applies to the beta MOs of a UHF wavefunction. This keyword also has an alternative meaning. By default, the IORDER and JORDER reordering instructions are applied after any processing of orbitals like orthogonalization or symmetry adaptation. However, for inputs with no separate beta orbital set (*e.g.*, RHF or MCSCF), the JORDER array, if given, allows one to reorder orbitals prior to the orbital processing stages. This feature is intended to work together with IORDER which additionally reorders orbitals *after* processing them. An example of a typical usage scenario is the use of MOs generated for one molecular geometry on a different molecular geometry. By putting the most important orbitals (say, orbitals forming the active space in the MCSCF) to the beginning of the orbital list before executing the processing steps, one can minimize the impact of the processing on these particular orbitals. The correct order can then be recovered using the IORDER array.

TOLZ	N	The level below which MO coefficients will be set to zero. (Default is equal to the value of TOLZ in $MOORTH)

TOLE	N	The level at which MO coefficients will be equated. This is a relative level, coefficients are set equal if one agrees in magnitude to TOLE times the other. (Default is equal to the value of TOLE in $MOORTH)

SYMDEN	Controls the use of a routine which projects all but the totally symmetric components out of the density matrix in order to obtain a symmetric density matrix. This may be useful if the HUCKEL or HCORE give orbitals with inexact symmetry, resulting to a non-totally symmetric density matrix. Since the generated density matrix may not be idempotent, this can generate a non-variational energy on the first iteration of HF and DFT SCF.	
	.TRUE.	Project all but the totally symmetric components out of the density matrix.
	.FALSE.	Do not perform this processing step. (Default)

MIX	Flag which controls the rotation of the alpha and beta HOMO and LUMO orbitals so as to generate inequivalent alpha and beta orbital spaces. This pertains to singlet UHF wavefunctions only. This may require the use of NOSYM=1 in $CONTRL depending on your situation.	
	.TRUE.	Rotate the alpha and beta HOMO and LUMO orbitals.
	.FALSE.	Do not rotate the alpha and beta HOMO and LUMO orbitals. (Default)

KDIAG	Selects the diagonalization routine used during initial guess generation.	
	0	Selects a very stable and fast diagonalization routine which requires large amount of extra memory. This routine is recommended for all Firefly input files, especially if the number of basis functions is large. (Default)
	-1	Selects a potentially less stable but even faster diagonalization routine which uses less memory than KDIAG=0.
	-2	Selects a combination of the two above methods which can be more stable than KDIAG=-1, is usually as fast as KDIAG =-1, but requires as much memory as KDIAG=0.
	1	Use EVVRSP diagonalization. This may be more accurate than the KDIAG=1 option in GAMESS (US).
	2	Use GIVEIS diagonalization (not as fast as EVVRSP but is more reliable).
	3	Use JACOBI diagonalization. This is the slowest method. This default is not sensible, but assures compatibility with Gamess (US).

PURIFY	Flag that can be used to symmetry-purify (*i.e.*, symmetry adapt) the input orbitals. This option is useful for removing symmetry contaminants in the input MOs which could cause a warning during the integral transformation phase.	
	.TRUE.	Symmetry-purify input orbitals.
	.FALSE.	Do not symmetry-purify input orbitals. (Default)

USEQMT	Flag that can be used to speed up the Hückel guess for semiempirical jobs by reusing the pre-computed Q matrix. It should not be used for any computations other than semiempirical runs and is of use only for very large systems. This option is experimental.	
	.TRUE.	Speed up the Hückel guess.
	.FALSE.	Do not try to speed up the Hückel guess. (Default)

$BASIS group

This group allows certain standard basis sets to be easily given. The choices of COORD=CART, ZMT, and ZMTMPC input coordinates require the use of a $BASIS group to define the basis set. The use of COORD=UNIQUE or HINT might or might not use $BASIS, as you wish. If this group is omitted, the basis set must be given instead in the $DATA group.

GBASIS	Chooses an internally stored basis or semi-empirical method.	
	MINI	Huzinaga's 3 gaussian minimal basis set (commonly known as MINI-1). Available for H-Rn.
	MIDI	Huzinaga's 21 split valence basis set (commonly known as MIDI-1). Available for H-Rn.
	STO	Pople's STO-NG minimal basis set. Available for H-Xe, for NGAUSS=2,3,4,5, 6.
	N21	Pople's N-21G split valence basis set. • Available for H-Xe, for NGAUSS=3. • Available for H-Ar, for NGAUSS=6.
	N31	Pople's N-31G split valence basis set. • Available for H-Ne, P-Cl for NGAUSS=4. • Available for H-He, C-F for NGAUSS=5. • Available for H-Ar, for NGAUSS=6. • For Ga-Kr, N31 selects the BC basis.
	N311	Pople's "triple split" N-311G basis set. • Available for H-Ne, for NGAUSS=6. • Selecting N311 implies MC for Na-Ar.
	DH	Dunning/Hay "double zeta" basis set. • (3s)/[2s] for H. • (9s,4p)/[3s,2p] for Li. • (9s,5p)/[3s,2p] for Be-Ne. • (11s,7p)/[6s,4p] for Al-Cl.
	DZV	"Double zeta valence" basis set. • A synonym for DH for H, Li, Be-Ne, Al-Cl. • (14s,9p,3d)/[5s,3p,1d] for K-Ca. • Selecting DZV implies BC for Ga-Kr.
	TZV	"Triple zeta valence" basis set. • (5s)/[3s] for H. • (10s,3p)/[4s,3p] for Li. • (10s,6p)/[5s,3p] for Be-Ne. • A synonym for MC for Na-Ar. • (14s,9p)/[8s,4p] for K-Ca. • (14s,11p,6d)/[10s,8p,3d] for Sc-Zn.
	BC	Binning/Curtiss "double zeta" basis. • (14s,11p, 5d/[6s,4p,1d] for Ga-Kr. • Selecting BC for other elements implies DZV.

	MC	McLean/Chandler "triple split" basis. • (12s,9p)/[6s,5p] for Na-Ar. • Selecting MC implies 6-311G for H-Ne.
The next two are ECP bases only		
	SBKJC	Stevens/Basch/Krauss/Jasien/Cundari valence basis set, for Li-Rn. This choice implies an unscaled -31G basis for H-He.
	HW	Hay/Wadt valence basis. This is a -21 split, available for Na-Xe, except for the transition metals. This implies a 3-21G basis for H-Ne.
The next four values specify semi-empirical methods		
	MNDO	Selects MNDO model Hamiltonian.
	AM1	Selects AM1 model Hamiltonian.
	PM3	Selects PM3 model Hamiltonian.
	RM1	Selects RM1 model Hamiltonian.

An overview of elements for which semi-empirical methods are available can be found in the 'Semi-empirical methods' chapter of this manual. If you pick one of these, all other data in this group is ignored. Note that semi-empirical runs actually use valence-only STO bases, not GTOs.

NGAUSS	N	The number of Gaussians (N). This parameter pertains only to GBASIS=STO, N21, N31, or N311. There is no default value.

NDFUNC	N	The number of heavy atom polarization functions to be used. These are usually d functions, except for MINI/MIDI. The term "heavy" means Na on up for GBASIS=STO, HW, or N21, and from Li on up otherwise. The value may not exceed 3. The variable POLAR selects the actual exponents to be used, see also SPLIT2 and SPLIT3. (Default is 0)

NFFUNC	N	The number of heavy atom f type polarization functions to be used on Li-Cl. This may only be input as 0 or 1. (Default is 0)

NPFUNC	N	The number of light atom, p type polarization functions to be used on H-He. This may not exceed 3, see also POLAR, SPLIT2, and SPLIT3. (Default is 0)

DIFFSP	Enables/disables the addition of an extra diffuse sp (L) shell for heavy atoms. Heavy means Li-F, Na-Cl, Ga-Br, In-I, and Tl-At.	
	.TRUE.	Add diffuse sp (L) shell to heavy atoms.
	.FALSE.	Do not add diffuse sp (L) shell to heavy atoms. (Default)

DIFFS	Enables/disables the addition of an extra diffuse s shell for hydrogen atoms.	
	.TRUE.	Add extra diffuse s shell to hydrogens.
	.FALSE.	Do not add extra diffuse s shell to hydrogens. (Default)

ELNEG	N1,N2,...	Array that instructs Firefly to add diffuse functions only to heavy elements specified by the ELNEG array. Values given correspond to atomic nuclear charges. For example, ELNEG(1)=7,8,9 will add diffuse functions to N, O, and F atoms, but not to other elements. This option does not have an effect is DIFFSP is not set. If no ELNEG is given, diffuse functions will be added to all supported heavy atoms (the default), provided DIFFSP=.true.

POLAR	Selects a set of polarization functions to use.	
	POPLE	Default if not explicitly stated otherwise.
	POPN311	Default for GBASIS=N311, MC
	DUNNING	Default for GBASIS=DH, DZV
	HUZINAGA	Default for GBASIS=MINI, MIDI
	HONDO7	Default for GBASIS=TZV

SPLIT2	N1,N2	An array of two splitting factors used when NDFUNC or NPFUNC is 2. Default is SPLIT2(1)=2.0,0.5

SPLIT3	N1,N2,N3	An array of three splitting factors used when NDFUNC or NPFUNC is 3. Default is SPLIT3(1)=4.00,1.00,0.25

EXTFIL	Flag that instructs Firefly to read the basis from an external file.	
	.TRUE.	Read the basis set *name* defined by GBASIS=*name* from the external basis set library file defined by the -b command line option.
	.FALSE.	Do not read the basis set from the external basis set library file. (Default)

Note: See the "basis sets" section of the manual for more information on how to use basis sets from an external file.

$D5 group

This group provides control over the use of pure spherical basis functions. The $D5 group only has an effect if the D5 option of the $CONTRL group is set as .TRUE. When set as .FALSE. (default value), Cartesian functions will be used and the D5, F7, and G9 keywords will be silently ignored.

D5	Flag that enables/disables the use of spherical functions for **d** shells.	
	.TRUE.	Use spherical functions for **d** shells. (Default)
	.FALSE.	Use Cartesian functions for **d** shells.

F7	Flag that enables/disables the use of spherical functions for **f** shells.	
	.TRUE.	Use spherical functions for **f** shells. (Default)
	.FALSE.	Use Cartesian functions for **f** shells.

G9	Flag that enables/disables the use of spherical functions for **g** shells.	
	.TRUE.	Use spherical functions for **g** shells. (Default)
	.FALSE.	Use Cartesian functions for **g** shells.

HSHIFT ESHIFT	Keywords used solely for debugging purposes.

$ZMAT group

This group lets you define the internal coordinates in which the gradient geometry search is carried out. These need not be the same as the internal coordinates used in $DATA. The coordinates may be simple Z-matrix types, delocalized coordinates, or natural internal coordinates. You must input a total of M=3N–6 internal coordinates (M=3N–5 for linear molecules). NZVAR in $CONTRL can be less than M if and only if you are using linear bends or alternatively if you are directing Firefly to automatically construct DLCs. It is also possible to input more than M coordinates if they are used to form exactly M linear combinations for new internals. These may be symmetry coordinates or natural internal coordinates. If NZVAR > M, you must input IJS and SIJ below to form M new coordinates. See DECOMP in $FORCE for the only circumstance in which you may enter a larger NZVAR without giving SIJ and IJS.

Basic input:

IZMAT	Defines simple internal coordinates. IZMAT is an array of integers defining each coordinate. The general form for each internal coordinate is: "encoded coordinate type",I,J,K,L,M,N. Possible input is:	
	1	followed by two atom numbers I and J. These specify a I-J bond length.
	2	followed by three numbers I,J, and K. These specify a I-J-K bond angle.
	3	followed by four numbers I,J,K, and L. These specify a torsion angle between planes I-J-K and J-K-L
	4	followed by four atom numbers I,J,K, and L. These specify an out-of-plane angle (Out of Plane bend, OPL) from bond I-J to plane J-K-L
	5	followed by three numbers I,J, and K. These specify a I-J-K linear bend. This counts as 2 coordinates for the degenerate bend, normally J is the center atom. See also the $LIBE group.
	6	followed by five atom numbers, I,J,K,L, and M. These specify a dihedral angle between planes I-J-K and K-L-M.
	7	followed by six atom numbers, I,J,K,L,M, and N. These specify a ghost torsion. Let A be the midpoint between atoms I and J, and B be the midpoint between atoms M and N. This coordinate is the dihedral angle A-K-L-B. The atoms I,J and/or M,N may be the same atom. (If I=J AND M=N, this is a conventional torsion.) Examples: N_2H_4, or, with one common pair, H_2POH

Example - a nonlinear triatomic, with atom 2 in the middle:

```
 $ZMAT IZMAT(1)=1,1,2, 2,1,2,3, 1,2,3 $END
```

This sets up two bonds and the angle between them. The blanks between each coordinate definition are not necessary, but improve readability mightily.

The next keywords define symmetry coordinates and natural internal coordinates.

SIJ	N1,N2,…	Specifies a transformation matrix of dimension NZVAR x M, used to transform the NZVAR internal coordinates in IZMAT into M new internal

		coordinates. SIJ is a sparse matrix, so only the non-zero elements are given, by using the IJS array described below. The columns of SIJ will be normalized by Firefly. (Default: SIJ = I, unit matrix)

IJS	N1,N2,…	Specifies an array of pairs of indices, giving the row and column index of the entries in SIJ.

For example, if the above triatomic is water, using

IJS(1) = 1,1, 3,1, 1,2, 3,2, 2,3

SIJ(1) = 1.0, 1.0, 1.0, -1.0, 1.0

gives the matrix S = 1.0 1.0 0.0

0.0 0.0 1.0

1.0 -1.0 0.0

which defines the symmetric stretch, asymmetric stretch, and bend of water.

The next keywords define delocalized coordinates (DLCs).

DLC	Flag that requests the use of delocalized coordinates.	
	.TRUE.	Use delocalized coordinates. This is the default if NZVAR in $CONTRL is non-zero and no $ZMAT group is given, and the input coordinates are not in the ZMT or ZMTMPC format.
	.FALSE.	Do not use delocalized coordinates. This is the default if NZVAR in $CONTRL equals zero, or if a $ZMAT group is given, or if the input coordinates are in the ZMT or ZMTMPC format.

AUTO	Flag that requests to generate all redundant coordinates automatically. The DLC space will consist of all non-redundant combinations of these which can be found. The list of redundant coordinates will consist of bonds, angles, torsions, and out of plane bends only.	
	.TRUE.	Generate redundant coordinates automatically. This is the default if NZVAR in $CONTROL is non-zero and no $ZMAT group is given, and the input coordinates are not in the ZMT or ZMTMPC format.
	.FALSE.	Do not generate redundant coordinates automatically. This is the default if NZVAR in $CONTRL equals zero, or if a $ZMAT group is given, or if the input coordinates are in the ZMT or ZMTMPC format.

NONVDW	N1,N2,…	An array of atom pairs which are to be joined by a bond, but might be skipped by the routine that automatically includes all distances shorter than the sum of van der Waals radii. Any angles and torsions associated with new bonds are also automatically included.

IXZMAT	N1,N2,…	An extra array of simple internal coordinates which you want to have added to the list generated by AUTO. Unlike NONVDW, IXZMAT will add only the coordinate(s) you specify. Its format is identical to that of IZMAT.

IRZMAT	N1,N2,…	An array of simple internal coordinates which you would like to remove from the AUTO list of redundant coordinates. It is sometimes necessary to remove a torsion if other torsions around a bond are being frozen, in order to obtain a nonsingular G matrix. Its format is identical to that of IZMAT.

IFZMAT	N1,N2,…	An array of up to 562 simple internal coordinates which you would like to freeze. Its format is identical to that of IZMAT. See also the FVALUE keyword below. Note that IFZMAT and FVALUE work only with DLCs. See the IFREEZ option in $STATPT if you wish to freeze simple or natural coordinates.

FVALUE	N1,N2,…	An array of up to 562 values to which the internal coordinates should be constrained. It is not necessary to input $DATA in such a way that the initial values match these desired final values, but it is helpful if the initial values are not too far away.

AUTOFV	A flag to generate the FVALUE array automatically using a current geometry. This option is especially useful for relaxed PES scans in DLCs.	
	.TRUE.	The FVALUE array is generated automatically. Any user input of FVALUE is thus ignored. (Default)
	.FALSE.	The FVALUE array is not generated automatically.

IFRZAT	N1,N2,…	An array of atom numbers you want to freeze during a geometry optimization using DLC. More precisely, the distances between all chosen atoms will be frozen, however, they will be allowed to rotate and translate as a single, united group, a rigid body.

IFDMOD	Selects one of three programmed methods used to freeze internals in DLCs.	
	0	The default method, which is the most stable one.
	1	A less stable method.
	2	An even less stable, experimental method.

DLCTOL	N	The threshold used at several stages during DLC generation as well as to impose geometrical constrains. Normally you do not need to alter it. However, in some cases setting it to, say, 1.0D-7 can help generating the required number of linearly independent DLC coordinates. (Default is 1.0D-5)

ORTTOL	N	Tolerance/threshold used to check the quality/completeness of DLCs after imposing geometry constrains. One may need to lower it to up to 1.0D-8 or below in some cases, for example when getting an "Unable to project DLC" error message. (Default is 1.0D-5)

FRATTL	N	Tolerance/threshold used to check the quality/completeness of constrains generated to freeze atoms specified in the IFRZAT array. One may need to lower it to up to 1.0D-7 or below in some cases. (Default is 1.0D-5)

CNVTOL	N	Threshold used in the conversion of the displacement from internal to Cartesian coordinates. When this conversion does not converge, Firefly will abort. In this case, you should increase the value of CNVTOL to 1.0D-7 or 1.0D-6. (Default is 1.0D-8)

STPMAX	N	The value of the maximum step size used during the conversion of internals to Cartesians. One can try to decrease it to 0.25 or below if the conversion routine diverges. (Default is 0.5)

UNSTABLE	A flag which can be used to force the propagation of updated Cartesians from the master node to all slave processes. It should be used on some hardware to avoid situations when some instances of parallel Firefly process get different results from the conversion that other ones. The use of this flag helps to avoid program hangs or unexpected behavior, for example when running on a cluster of non-all-identical nodes.	
	.TRUE.	Propagate updated Cartesians from the master node.
	.FALSE.	Do not propagate updated Cartesians from the master node. (Default)

DELANG	N	Threshold (in degrees) to delete ill-defined angles, *i.e.* angles that are larger than DELANG. (Default is 178.0)

DELTOR	N	Threshold (in degrees) to delete ill-defined torsions, *i.e.* those that are larger than DELTOR by an absolute value. In addition, DELTOR affects the deletion of ill-defined OPLs, *i.e.* those that are larger than DELTOR/2 by an absolute value. (Default is 172.0)

TORTOL	N	Threshold value (in radians) to forcibly equate torsions and Out of Plane bends (OPLs) to zero or π (for torsions) and $+\pi/2$ (for OPLs) if they are close enough to these values, *i.e.* the difference is less than TORTOL. (Default is 0)

OPLTHR	N	Threshold (in degrees) used to replace torsions with an absolute value larger than OPLTHR with the corresponding OPLs to improve the quality of the DLCs. (Default is 150.0)

STRICT	Enables/disables the automatic elimination of some extra (superfluous) torsions and OPLs from DLCs. This elimination is optional as it is not required for the construction of the DLCs. However, the presence of the extra coordinates may affect the effectiveness of the DLCs.	
	.TRUE.	Enable this elimination.
	.FALSE.	Disable this elimination. (Default)

HBOND	N	The maximum possible hydrogen bond length (in Ångstrom) for the reference O-H-O hydrogen bond used during the automatic generation of hydrogen bonds. Setting it to zero disables this feature altogether and will revert the behavior to that of Firefly 7.1.F. Setting it to a negative value will revert the behavior to that of Firefly 7.1.G. (Default is 2.1)

SYMREP	Integer bitfield forcing symmetry replication of IXZMAT (bit 0), IRZMAT (bit 1), IFZMAT/IFZRAT (bit 2), and NONVDW (bit 3) arrays. *E.g.*, SYMREP=9 (or 0x9 as Firefly equally accepts decimal, binary, octal, and hexadecimal inputs) will symmetry replicate the IXZMAT and NONVDW arrays. The simplest way to enforce all arrays to be symmetry replicated is to enter SYMREP=-1. The default is 0, *i.e.* no replication at all.

SCAN	A flag that triggers the use of a specific scan engine that is designed to be used for scans in DLCs, and that is generally the most suitable for most types of scans.	
	.TRUE.	Enable this scan engine. This should be used for scans in DLCs.
	.FALSE.	Disable this scan engine. (Default)

$LIBE group

This group is required if linear bends are used in $ZMAT. It contains only one keyword.

APTS	X1,Y1,Z1, X2,Y2,Z2,…	An array that specifies linear bends.

A degenerate linear bend occurs in two orthogonal planes, which are specified with the help of a point A. The first bend occurs in a plane containing the atoms I,J,K and the user input point A. The second bend is in the plane perpendicular to this, and containing I,J,K. One such point must be given for each pair of bends used.

APTS(1)= x1,y1,z1,x2,y2,z2,... for linear bends 1,2,...

Note that each linear bend serves as two coordinates, so that if you enter 2 linear bends (HCCH, for example), the correct value of NZVAR is M–2, where M=3N–6 or 3N–5, as appropriate.

$SYSTEM group

This group provides global control information for your computer's operation. This is system related input and will not seem particularly chemical to you.

The following keywords provide control over execution time and memory usage.

TIMLIM	N	CPU time limit, in minutes. When running in parallel, this is the CPU time allotted to each instance of Firefly. When running Firefly using batch queue systems like PBS, SLURM, etc..., set TIMLIM to about 95 percent of the time limit given to the batch job so that Firefly can stop itself gracefully if this is possible. Note, not all types of jobs honor TIMLIM. (Default is 2880)

MWORDS	N	The maximum amount of dynamic memory that can be used by each instance of Firefly. This value is given in units of 1,000,000 words (as opposed to 1024*1024 words), where a word is always a 64 bit/8 byte quantity (*i.e.*, a word is a double precision (DP) number that consumes eight bytes of memory). Firefly allocates this memory at runtime. In case finer control over the amount of memory is needed, this value can be given in units of words by using the keyword MEMORY instead of MWORDS. (Default is 20)

MEMORY	N	Has the same function as MWORDS, but uses words instead of MWords. If not given, the MWORDS keyword is used. If MWORDS and MEMORY are specified, the larger of the two values is used. (Default is 20000000)

SAFMEM	N	The size of the safety memory pool, in words, which can be used by some parts of Firefly in the case when there is not enough dynamic memory to complete the current operation. In some cases, Firefly may require to use a bit more memory than the amount reserved via MWORDS or MEMORY. Firefly will attempt to allocate this additional amount from the safety memory pool. Thus, increasing SAFMEM can help with some badly behaving jobs. (Default is 16384)

MASMEM	N	Same as MWORDS, but applies to Firefly's grand master process as well as for local masters in extended XP mode. If not given, MWORDS/MEMORY is used instead. (Default is 0, *i.e.* no explicit MASMEM is given).

SHMEM	N	The size of the shared memory pool, in words. This memory can optionally be used by Firefly’s MP4 code when running in SMP parallel mode on a standalone SMP/multi-core system. This option is supported under Windows OSs only. (Default is 0)

WSCTL	Specifies who may control the amount of physical memory. For this option to work one may need to have elevated privileges.	
	.FALSE.	The operating system controls the amount of physical memory (the size of the working set) used by Firefly.
	.TRUE.	Firefly dynamically allocates the working set that is as large as the amount of memory needed for the current operation. The process working set is the amount of physical (note: not virtual) memory that is allotted to this process. The dynamic allocation strategy usually reduces paging for memory-demanding jobs considerably. On the other hand, this also reduces the amount of physical memory available to other processes and to the operating system. (Default)

MAXWS	N	The upper limit on the size of the Firefly's working set, in words. Firefly will never try to set its working set larger than this value. A value of 0 asks for the automatic selection of an appropriate limit. (Default is 0)

DECOMM	Provides control over the memory management behavior.	
	.FALSE.	Old-style memory manager behavior. Use this option only if you encounter some unexpected problems with the default setting.
	.TRUE.	Turns on the advanced memory management features. The advanced memory management is enabled by default on most systems. (Default is .TRUE. under Windows OS. Under Linux OS, the default is .TRUE. if not running over InfiniBand and .FALSE. otherwise)

IDLE	Provides control over the CPU scheduling priority.	
	.TRUE.	Flag to reduce Firefly’s CPU scheduling priority. This feature may not work well under some operating systems.
	.FALSE.	Use the default scheduling priority. (Default)

The following keywords provide control over file I/O.

BLKSIZ	N	The size of the I/O buffer to be used by the standard Fortran I/O routines. The default value is 0, which tells Firefly not to set the size explicitly and instead use the Watcom Fortran runtime library's default value of 8192 bytes.

LDAR	N	The value of LDAR parameter (the size of the direct access file record) for $INTGRL, $GUGDM2, $TRFDM2, $CISORT, and $MCSCF groups. (Default is 2045)

FASTF	Provides controls over file I/O implementation to be used by Firefly

	.FALSE.	Use standard Fortran Input/output routines for file I/O
	.TRUE.	Turns on the use of the fast non-Fortran file I/O routines, the so-called FSF routines. FSF stands for the Fast System Files and is a powerful set of fast non-Fortran file I/O routines that is specific to Firefly. By default, FSF routines are used by Firefly to perform all file I/O operations. This increases the overall I/O performance and allows Firefly to handle large (> 2GBytes) files. When Firefly is run on a file system supporting large files with the FASTF option enabled, all files handled by the FSF routines are allowed to have their size up to the native limit of that file system. On file systems which do not support large files, the latter will be split automatically into several smaller chunks. (Default is .TRUE. i.e. to use FSF)

The following keywords (TRUNCF to IORTRY) pertain only to jobs for which FASTF is enabled.

TRUNCF	Provides controls over the truncation of files	
	.FALSE.	Existing files of non-zero length will not be truncated when being reopened for writing. This allows one to (pre)allocate (almost) all the required disk space at the beginning of a Firefly run. This may be useful in some cases as it prevents other disk-intensive programs from causing Firefly to abort when they temporarily exhaust the free disk space.
	.TRUE.	The files are truncated while being (re)opened for writing. This results in faster write operations. (Default)

FLUSH	Provides control over the use of file cache flushes.	
	.FALSE.	Disables the use of file cache flushes for Firefly's working files. It is recommended to disable file flushing if: the total size of all Firefly working files is less than the available size of the OS file cache, or if a high performance RAID or SSD volume is used for scratch storage
	.TRUE.	Enables the use of file cache flushes. Firefly will flush the cache buffers of its working files onto disk as necessary. (Default)

ASYNC	Provides control over asynchronous file I/O.	
	0	Disables asynchronous file I/O. (The default on Linux OSs)
	1	Activates asynchronous file I/O for most of files. (The default on Windows OSs)
	2	Activates asynchronous file I/O, even for files for which asynchronous I/O usually does not improve performance.

AIOBUF	N	The buffer size for asynchronous file I/O, in kilobytes. Buffers are allocated on a per file basis. (Default is 2048)

AIOPTY	One of -15, -2, -1, 0, 1, 2, 15	The priority boost for dedicated asynchronous file I/O threads. This option is available under Windows OSs only. (Default is 2)

SPLITF	Provides control over the splitting of large files.	
	.TRUE.	Enables splitting of large files into chunks. This is a workaround for file systems which do not support large files natively,
	.FALSE.	Disables splitting of large files into chunks. It is safe to use this option if the file system in use supports large files i.e. the files larger than 2 GBytes. (Default is .FALSE. except if running on FAT or HPFS file systems).

VOLSIZ	N	The chunk size used to split large files, in MBytes. (Default is 2047)

MXIOB	N1,N2	Sets the system wide buffer size of a single I/O operation, in Kilobytes. The first value sets the size of buffer used for file reads, the second value sets the size of buffer used for file writes. MXIOB(1)=512,256 usually works better with SSDs than the default values. (Default is 128,128)

MEMF	Provides control over memory-related behavior.	
	.FALSE.	The default behavior.
	.TRUE.	Directs Firefly to try to keep all important files in memory by using a dedicated file I/O API to provide special hints to the OS level cache/memory manager.

FRESHF	Provides control over I/O-related behavior.	
	.FALSE.	The default behavior.
	.TRUE.	Close and then reopen files instead of just rewinding them. This may improve performance a bit when running on Windows NT 4.0.

NOSEOF	Provides control over I/O-related behavior.	
	.FALSE.	The default behavior.
	.TRUE.	Does not explicitly set the end of files. This is a workaround for some buggy Linux kernels.

MEMCPY	One of -3, -2, -1, 0, 1, 2, 3	Selects the memory copying method used by the FSF routines. The default is 0. Use of other values e.g. 3 or -3 may improve performance a bit.

IOFLGS	N	An array of 100 elements that can be used to provide I/O optimization flags to FSF routines and the operating system on a per Fortran logical unit base. *E.g.*, IOFLGS(9) corresponds to Fortran's unit 9 which, in turn, for Firefly corresponds to the file "MOINTS". One can contact Firefly's developers for a complete list of units and corresponding files. The I/O optimization flags are the bitwise OR of the flags described below: 0x00000001 any writes to a file must go through the file cache directly to the disk (i.e. no lazy writes allowed) 0x00000002 optimize file I/O for sequential file access 0x00000004 optimize file I/O for random file access 0x00000008 does not explicitly set the end of file for this particular unit (exactly like the NOSEOF flag but on a per unit base) 0x00000010 split this particular unit into chunks (exactly like the SPLITF flag but on a per unit base) 0x00000020 use asynchronous I/O for this particular unit and use OS level blocking calls when waiting for I/O completion 0x00000040 use asynchronous I/O for this particular unit and use spin loops when waiting for I/O completion 0x00000080 try to keep this file in memory (exactly like the MEMF flag but on a per unit base) 0x00000100 set TRUNCF flag on for this particular file 0x00000200 file will be compressed if the underlying file system supports compression 0x00000400 file will be marked as sparse f the underlying file system support sparse files 0x00000800 records in the file will be protected by CRC32. Note this flag is currently implemented only for selected units 0x00001000 file will be deleted before opening for rewrite 0x00002000 use constant block size disk reads for this file 0x00004000 use constant block size direct non-buffered disk reads (no disk cache involved) for this file 0x00008000 use deferred I/O for this file 0x00020000 set FRESHF flag on for this particular file Some exotic combinations of I/O optimization flags cannot be used. For these combinations, the FSF layer will abort the job. Some flags are not supported under Linux OSs. The default for the IOFLGS array is that all elements are zeroed. Input elements are combined with the global flags described above (such as MEMF, FRUNCF, etc.) and are passed to FSF routines.

IORTRY	N	The maximum number of I/O attempts to try before reporting an I/O failure. The default is 1, *i.e.* there will be no I/O retries.

The following keywords provide control over parallel execution and can help in the case of MPI-related problems. Do not modify the default values unless you are absolutely aware that you need to do this.

MXBCST	N	The maximum size, in words, of the message used in a single broadcast operation. A value of -1 means there will be no limit. (Default is 32768)

MPISNC	Provides control over MPI synchronization calls.	
	.FALSE.	Selects the normal operation, except for OpenMPI which is buggy and requires this option to be enabled. (Default)
	.TRUE.	Activates a strategy where a call to the broadcast routines will periodically synchronize all MPI processes.

MXBNUM	N	The maximum number of broadcast operations which can be performed before the global synchronization call is done. Relevant only if MPISNC is .TRUE. (Default is 100)

LENSNC	N	The maximum aggregate length, in words, of the messages that can be broadcasted before the global synchronization call is done. Relevant only if MPISNC is .true. The default length depends on the number of cores used. A reasonable value is in the range of 4096 to 65536.

The following keyword provides control over AO integral storage during conventional parallel integral transformations.

AOINTS	Provides control over the AOINTS integral file duplication.	
	DUP	A duplicate of the AOINTS integral file will be stored on each node. (Default)
	DIST	The AOINTS integral file is distributed across all nodes.

The following keywords provide basic control over threading.

MKLNP	N	Sets the number of physical (not logical) CPU cores to be used by Firefly via multithreaded BLAS level 3 routines. If the NP option below is not given, the default is 1. If NP is given (but MKLNP is not), MKLNP equals NP.

MKLAFF	N	An array of 32 elements controlling the affinity mask for BLAS level 3 threads. The affinity is assigned by a cyclic fashion on a per process basis. The length of a cycle is defined by the number of consecutive non-zero elements in the MKLAFF array starting from its first element. *E.g.*, MKLAFF(1)=0xff,0xff00,0,… defines a cycle of the length two. MKLAFF(1)=-1 means no special affinity mask. The default is MKLAFF(1)=-1,0,0,... *i.e.* no special affinity mask for BLAS level 3 threads of all processes.

XPBIND	Flag that modifies the behavior of MKLAFF.	
	.FALSE.	Normal operation. (Default when not running in extended XP mode.)
	.TRUE.	Modifies the behavior of MKLAFF so that the cycle assigning affinities is restarted at the start of each XP group of processes. (Default in extended XP mode)

MKLSMP	Flag that provides control over the threading implementation.	
	.FALSE.	Normal operation. (Default)
	.TRUE.	For some of the threaded code inside Firefly, uses threading as implemented within BLAS level 3 routines rather than coarse-grained threading over BLAS level 3 routines.

BLAS3	Provides control over the state of additional threads created by BLAS level 3 routines.	
	SLEEP	Additional threads created by BLAS level 3 routines are suspended when not in use. (Default)
	NOSLEEP	Additional threads are permanently active. Normally, one should not use this option as it can cause severe performance degradation.

NP	N	Sets the number of physical (not logical) CPU cores to be used by Firefly via the native multithreaded code. When the MKLNP option is not given, the default is 1. If MKLNP is given (but NP is not), NP equals MKLNP.

The following keywords are basic floating point control switches.

KDIAG	Provides control over the matrix diagonalization routine.	
	0	Selects the use of a very stable and fast diagonalization routine which requires a large amount of extra memory. (Default)
	-1	Selects a potentially less stable but even faster diagonalization routine which uses less memory than KDIAG=0 (experimental).
	-2	Selects a combination of the two methods above that is more stable than KDIAG=-1 and is usually as fast as KDIAG=-1, but that requires as much memory as KDIAG=0 (experimental).
	1	Selects the use of EVVRSP diagonalization (not recommended). This is much slower than the fast KDIAG=0,-1,-2 methods.
	2	Selects the use of GIVEIS diagonalization. This is even slower than EVVRSP but is much more reliable than EVVRSP.
	3	Selects the use of JACOBI diagonalization (this is the slowest method).

NOJAC	N	Never use Jacobi diagonalization for matrices of size N by N and above. (Default is 50)

L2SIZE	N	The size of the fast Level 2 cache per CPU core, in Kilobytes. The information on cache size is used by Firefly to optimize its performance. The default is to determine the correct size automatically based on the particular hardware in use. If Firefly fails to detect the size of the Level 2 cache properly, this keyword can be used to specify the correct amount.

FPECHK	Provides control over checks for floating point exceptions.	
	.TRUE.	Enables checks for floating point exceptions and abort the job if any are encountered.
	.FALSE.	Disables checks for floating point exceptions. (Default)

$SMP group

This group contains keywords aimed to control the execution of Firefly running on SMP and multi-core systems. Similarly to the $SYSTEM group, the $SMP group provides control over your computer's operation and does not contain any particular chemistry-related settings.

The following keywords provide fine control over threading.

MKLNP	N	Sets the number of physical (not logical) CPU cores to be used by Firefly via multithreaded BLAS level 3 routines. If the NP option below is not given, the default is 1. If NP is given (but MKLNP is not), MKLNP equals NP. Note, this option is currently duplicated in the $SYSTEM group for compatibility reasons.

NP	N	Sets the number of physical (not logical) CPU cores to be used by Firefly via the native multithreaded code. When the MKLNP option is not given, the default is 1. If MKLNP is given (but NP is not), NP equals MKLNP. This option defines the number of primary 'CPU worker' threads for threaded code. Note, this option is currently duplicated in the $SYSTEM group for compatibility reasons.

HTTNP	Sets the number of logical cores belonging to the same physical core to be used by the native multithreaded code.	
	1	Directs Firefly to use only the first of all siblings belonging to the same physical core. (Default on systems without HyperThreading)
	2	Allows one to use two siblings per physical core provided this is supported by the code. Compared to HTTNP=1, this option effectively doubles the number of “CPU worker" threads for this code by creating the secondary ”CPU worker” threads. (Default on HyperThreading capable systems)

MKLSMP	Provides control over threading with BLAS level 3 routines.	
	.TRUE.	For some of the threaded code inside Firefly, this option enables the use of threading as implemented within BLAS level 3 routines rather than coarse-grained threading over BLAS level 3 routines. Note, this option is currently duplicated in the $SYSTEM group for compatibility reasons.
	.FALSE.	Normal operation. (Default)

TPOOL	N	Directs to create a thread pool containing up to N threads. Thread pooling allows threads to be reused as needed instead of being created when needed and destroyed when not needed. Pooled threads do not consume CPU resources. A value of 0 disables thread pooling. The default is TPOOL=128, *i.e.* to create a thread pool of up to 128 threads.

THRSTK	N	The default thread's stack size, in Bytes. (Default is 16384)

PCGAFF	N	An array of 32 elements controlling the affinity mask for the entire Firefly process, *i.e.* this mask has effect over ***any*** threads created by Firefly. The affinity is assigned in a cyclic fashion on a per process basis. The length of a cycle is defined by the number of consecutive non-zero elements in the PCGAFF array starting from its first element. *E.g.*, PCGAFF(1)=0xff, 0xff00, 0,... defines a cycle of the length two. -1 means no special affinity mask. The default is PCGAFF (1)=-1,0,0... *i.e.* no special affinity mask for all processes. Note, the semantics of this option is affected by the XPBIND option of the $SYSTEM group in the same way as is explained for the MKLAFF option below.

WRKAFF	N	An array of 32 elements controlling the affinity mask of all primary 'CPU worker' threads. The affinity is assigned in a cyclic fashion on a per process basis. The length of a cycle is defined by the number of consecutive non-zero elements in the WRKAFF array starting from its first element. *E.g.*, WRKAFF(1)=0xff, 0xff00,0,... defines a cycle of the length two. -1 means no special affinity mask. The default is WRKAFF(1)=-1,0,0... *i.e.* no special affinity mask for all 'CPU worker' threads of all processes. Note, the semantics of this option is affected by the XPBIND option of the $SYSTEM group in the same way as is explained for the MKLAFF option below.

MKLAFF	N	An array of 32 elements controlling the affinity mask of all BLAS level 3 threads. The affinity is assigned in a cyclic fashion on a per process basis. The length of a cycle is defined by the number of consecutive non-zero elements in the MKLAFF array starting from its first element. *E.g.*, MKLAFF(1)=0xff,0xff00,0,... defines a cycle of the length two. -1 means no special affinity mask. The default is MKLAFF(1)=-1,0,0... *i.e.* no special affinity mask for BLAS level 3 threads of all processes. If the XPBIND option of the $SYSTEM group is set it modifies the behavior of MKLAFF so that the cycle assigning affinities are restarted at the start of each XP group of processes. Note, this option is duplicated in $SYSTEM group for compatibility reasons.

EXPAFF	Provides control over the 'CPU worker' thread's affinities.	
	0	Normal operation. (Default)
	1 or 2	When supported by the code, set the primary 'CPU worker' thread's affinities explicitly by binding each CPU working thread to its own core, and ignoring the WRKAFF option. With EXPAFF=1 threads are bound to cores # 0, 1, 2, 3, etc..., while with EXPAFF=2 they are bound to cores # 0, 2, 4, 6, etc. (the latter option is useful on Windows systems with HyperThreading enabled in hardware and software).

RODUP	Provides control over the replication of some frequently used read-only data for threaded code.	
	.TRUE.	Use a per-thread separate copy of this data.
	.FALSE.	Share this data across all threads. (Default)

ALLOC	Provides control over the allocation of unused core siblings (logical cores) of HyperThreading enabled cores.	
	.TRUE.	Allocate unused core siblings of HyperThreading enabled cores by creating special additional low-priority light-weighted threads assigned to these cores.
	.FALSE.	Do not allocate unused core siblings. (Default)

HTTFIX	Provides control over the affinity masks of Firefly's primary 'CPU worker' threads on HyperThreading-enabled systems.	
	.TRUE.	Modify affinity masks of Firefly's primary 'CPU worker' threads for optimal performance. Note this will disable the use of several core siblings belonging to the same core by the primary 'CPU worker' threads *i.e.* at most only a single sibling of each physical core will be used. (Default)
	.FALSE.	Do not modify affinity masks of Firefly's primary 'CPU worker' threads for better performance. Note this may enable the use of several core siblings belonging to the same core by the primary 'CPU worker' threads.

HTTALT	Allows one to modify the behavior of HTTFIX option.	
	.TRUE.	Modify the behavior of the HTTFIX option so that the sibling used for computations will be selected complementary to that assigned by default by the HTTFIX option. Use of this option in one of two Firefly jobs running on the same computer system allows them to use all logical cores (hereby sharing physical cores) and avoid OS scheduler over-subscription. The default is HTTALT=.FALSE.
	.FALSE.	Do not modify the behavior of the HTTFIX option. (Default)

HTTPAR	Provides control over HTTALT flags.	
	.TRUE.	Inverts the HTTALT flag of the every second Firefly instance of the parallel Firefly process. This option can be used to allow parallel Firefly jobs to use all logical cores of HyperThreading-enabled systems.
	.FALSE.	Normal operation. (Default)

SMPPAR	Enables/disables the use of a mixed parallel/multithreaded mode.

	.TRUE.	If set, this option means that Firefly should be executed in the mixed parallel/multithreaded mode, hereby switching back and forth between a parallel mode of execution and a threaded mode of execution as needed. The user must provide valid MKLNP and NP values. Additional threads will only be used by the grand master and the local master processes executing the threaded part of the computation.
	.FALSE.	Do not to use a mixed mode of execution. (Default)

XHTT	Provides control over the creation of secondary 'CPU worker' threads.	
	.TRUE.	Forces the SMP/multicore aware parts of Firefly, which do not normally use additional (secondary) 'CPU worker' threads for execution on the extra logical processors of HT capable multi-core, SMP, or HTT systems, to use these additional logical cores by creating secondary 'CPU worker' threads. The default is XHTT=.FALSE., *i.e.* use the code's default heuristics when running on SMT-capable system rather than forcing the use of HT. This option does not has an effect if HTTNP is set to 1.
	.FALSE.	Normal operation (Default)

CSMTX	Provides control over thread synchronization.	
	.TRUE.	Use critical sections for threads synchronization. Critical sections are normally faster. (Default)
	.FALSE.	Use mutexes for thread synchronization .

FTIMER	Provides control over Windows system interval timer. This option is supported under Windows OSs only.	
	.TRUE.	Set the highest possible resolution of the Windows internal system interval timer. This decreases the thread's time slice and allows threads to be preempted/rescheduled with a higher frequency at the costs of some additional OS overhead.
	.FALSE.	Normal operation. (Default)

STTF	N	If nonzero, this option allows the use of the Windows' explicit thread switching API from inside the thread's spin wait loops. The particular types of waiting primitives allowed to use the thread switching API are coded by the bits of the STTF variable. Contact Firefly's developers for a complete list of the available bitfields and their exact meaning. This option is supported under Windows OSs only. The default is STTF=0, *i.e.* not to use this feature.

SLEEP	Provides control over the use of the OS's sleep functions API to initiate thread

	rescheduling when appropriate.	
	.TRUE.	Use the OS's sleep functions API.
	.FALSE.	Do not use the OS's sleep functions API. (Default)

THRTMT	N	An internal timeout used when waiting for threads, in milliseconds. A value of -1 disables timeouts. The default value depends on the operating system. There is normally no need to change the value of this keyword.

EVTTMT	N	An internal timeout used when waiting for events, in milliseconds. A value of -1 disables timeouts. The default value depends on the operating system. There is normally no need to change the value of this keyword.

MTXTMT	N	An internal timeout used when waiting for mutexes, in milliseconds. A value of -1 disables timeouts. The default value depends on the operating system. There is normally no need to change the value of this keyword.

DLLHLT	Provides control over the suspension of threads created by Firefly's dynamic extension libraries.	
	.TRUE.	When not in use, suspend execution of the OpenMP working and control threads created by Firefly's dynamic extension libraries such as fastdiag. (Default on Windows OSs)
	.FALSE.	Normal operation. (Default on non-Windows OSs)

BLKTIM	N	Thread blocking timeout for the OpenMP parts of the code, in milliseconds. The default is 200 under Windows and 0 under other operating systems.

SMPFCK	Provides control over the use of the multithreaded code for Fock matrix construction when running in conventional (*i.e.* with two-electron integrals stored on disk) mode.	
	.TRUE.	Enables the use of this multithreaded code. (Default)
	.FALSE.	Disables the use of this multithreaded code.

The following keywords provide control over the Firefly's runtime library.

RTLPAR	N	An integer array allowing one to modify the behavior of the compiler's runtime libraries (RTL) used by Firefly. The default settings depend on the particular OS in use. Normally, there is no need to modify the default settings. Contact Firefly's developers for a complete list of available options and their exact meaning.

MXHEAP	N	The maximum allowed size of the heap area, in Kilobytes. The default is MXHEAP=-1 which means there is no limit to the heap size. MXHEAP=0 disables heap usage.

CSTACK	N	The size of the stack of the main Firefly thread to pre-commit at the beginning of run, in Kilobytes. The default is CSTACK=0, *i.e.* stack will not be pre-committed.

The following keywords are used for execution tuning and optimization.

L2SIZE	N	The size of the fast Level 2 cache per CPU core, in Kilobytes. The information on cache size is used by Firefly to optimize its performance. The default is to determine the correct size automatically based on the particular hardware in use. If Firefly fails to detect the size of the Level 2 cache properly, this keyword can be used to specify the correct amount. Note, this option is currently duplicated in the $SYSTEM group for compatibility reasons.

L3SIZE	N	The size of the slow Level 3 cache per CPU core, in Kilobytes, when available. The information on cache size is used by Firefly to optimize its performance. The default is to determine the correct size automatically based on the particular hardware in use. If Firefly fails to detect the size of the Level 3 cache properly, this keyword can be used to specify the correct amount.

CALL64	Provides control over the use of fast 64-bit computational kernels.	
	.TRUE.	Allow calls to fast 64-bit computational kernels. By default, this option is enabled on under Windows and Linux, and is disabled under Mac OS X as it results in a crash of the operating system. Note, the use of this option may conflict with para-virtualized Linux kernels running under Xen virtualization software. This option is only supported under 64-bit OSs.
	.FALSE.	Do not call fast 64-bit computational kernels.

CUDA	Provides control over the use of CUDA.	
	.TRUE.	Allow Firefly to use CUDA, if supported by the particular code being used. See the $CUDA group for CUDA-related options.
	.FALSE.	Do not use CUDA (Default)

NOAVX	Provides control over the use of AVX instructions (when supported by hardware and OS).	
	.TRUE.	Disables the use of AVX instruction. (Default)

	.FALSE.	Enabled the use of AVX instruction.

The following keywords provide fine control over the DGEMM code. By default, Firefly selects the use of the most suitable implementation of the DGEMM routine. The algorithm used for selection is rather involved, especially on non-Intel CPUs. The default selection can be modified using the keywords described below.

MKL64	N (allowed values are 1 to 14)	Forces the use of one of the available 64-bit MKL's DGEMM implementations. The default is to pick up the most suitable one automatically, when appropriate. It is generally not recommended to change the default value. This option does not have any effect if the CALL64 option is disabled.

MMMTYP	N (allowed values are -9 to 9)	Forces the use of one of the available alternative 32-bit and 64-bit DGEMM implementations. The default is to pick up the most suitable one automatically, if appropriate. It is generally not recommended to change the default value.

MSU32	Provides control over the use of a specific 32-bit DGEMM version optimized for the first generation of Intel Core 2 processors (those with codenames Merom, Conroe, Woodcrest, and Clovertown).	
	.TRUE.	Use this specific 32-bit DGEMM version.
	.FALSE.	Do not use this specific 32-bit DGEMM version. (Default)

SSG32	Provides control over the use of a specific 32-bit DGEMM version optimized for the second generation of Intel Core 2 processors (those with codenames Penryn, Wolfdale, Yorkfield, Harpertown, etc.)	
	.TRUE.	Use this specific 32-bit DGEMM version.
	.FALSE.	Do not use this specific 32-bit DGEMM version. (Default)

LOAD	Provides control over the use of one of the 32-bit optimized DGEMM implementations from the dynamic load library p4stuff.dll (Windows) or p4stuff.ex (Linux).	
	.TRUE.	Use one of these 32-bit optimized DGEMM implementations.
	.FALSE.	Do not use one of these 32-bit optimized DGEMM implementations. (Default)

MKLLVL	0,1,2, or 3	Controls to what extent MKL's DGEMM implementations will be preferred over the non-MKL DGEMM ones. MKLVL=3 disables most of the non-MKL code. The default value is MKLLVL=2.

FSTDIA	Provides control over the use of fast diagonalization routines from the fastdiag dynamic library (fastdiag.dll/fastdiag.ex).	
	.TRUE.	Use these diagonalization routines, provided the fastdiag library is available. (Default)
	.FALSE.	Do not use these diagonalization routines.

DIANP	N	The maximum number of threads to be used by the fast diagonalization routines from the fastdiag dynamic library. The default is equal to the value of the MKLNP option.

EXPMEM	Provides control over the export of Firefly's advanced memory management routines for use by the fastdiag dynamic library.	
	.TRUE.	Export Firefly's advanced memory management routines. (Default)
	.FALSE.	Do not export Firefly's advanced memory management routines.

EXPMMM	Provides control over the export of Firefly's advanced DGEMM routines for use by the fastdiag dynamic library.	
	.TRUE.	Export Firefly's DGEMM routines. (Default)
	.FALSE.	Do not export Firefly's DGEMM routines.

$NUMGRD group

The $NUMGRD allows control over the numerical calculation of gradients, which are requested by specifying NUMDER=.TRUE. in $CONTRL. The primary purpose of the numerical gradient code is to allow computations normally requiring analytic energy gradients, to be performed using those QC methods for which analytic gradients are not yet programmed. As with any finite differencing, numerical gradients require extra high precision in computed energy values. It is therefore the user's responsibility to increase the precision of all the stages involved into computation of target energies.

DELTA	N	Step size (in atomic units) for finite differencing. (Default is 0.01)

ORDER	The order of finite difference formulas to use. Finite differencing of order X requires X × (3N – 6) + 1 energy evaluations (assuming a non-linear molecule with C1 symmetry) with N being the number of atoms. The use of symmetry, if any, can dramatically reduce this number.	
	1	X = 1. This option is the least expensive and least accurate.
	2	X = 2. (Default)
	4	X = 4.
	6	X = 6. This methods is the most expensive and most accurate.

NGRADS	N	For methods computing energies of several states at once (e.g. CI, MCSCF, TDHF, TDDFT, MCQDPT2, XMCQDPT2) this option requests Firefly to compute gradients for an N number of (lowest) energy states. The default is NGRADS=1, *i.e.* the gradient will only be computed for the lowest energy state. Use of this option does not introduce any additional computational overhead. For TDHF/CIS/TDDFT, the NGRADS numbering excludes the reference ground state energy. Note, the value of NGRADS should not be larger than the actual number of states available! (Default is 1)

ISTATE	N	This option selects the target state of interest, *i.e.* the state which' gradient is to be used by other parts of the program, for example during geometry optimization involving numerical gradients. This option is intended for use together with the NGRADS option. For instance, to optimize the second root of TDDFT or XMCQDPT, set NGRADS equal to or greater than 2, set ISTATE to 2, and properly set other relevant parameters controlling the number of computed roots during TDDFT or XMCQDPT procedure to capture at least two states. (Default is 1)

JSTATE	N	Selects the second state of interest. The default is JSTATE=0, *i.e.* no second state of interest. One must provide a nonzero JSTATE together with a valid NSTATE and ISTATE for the numerical location of interstate crossings or conical intersections. (Default is 0)

DELROT	Logical variable affecting the behavior of numerical differencing code with respect to the elimination of rigid body rotations. Normally, DELROT does not affect optimized geometries but does slightly affect double-numerical Hessians. This option must be manually set .FALSE. for any RUNTYP other than OPTIMIZE or SADPOINT and for any optimization runs that uses HSSEND=.TRUE.	
	.TRUE.	Selects an approximate elimination of rigid body rotations during computations of numerical gradients, thus reducing the number of required reevaluations of energy. (Default)
	.FALSE.	Do not eliminate rigid body rotations.

CHEBSH	Flag that selects the use of Chebyshev grids for finite differencing.	
	.TRUE.	Use Chebyshev grids.
	.FALSE.	Use equidistant central finite differences. (Default)

SPLINE	N	Selects the use of spline differentiation of order SPLINE over the selected grid (either equidistant or Chebyshev). It has an effect only for ORDER=4 and ORDER=6 finite differences. Valid values of SPLINE are 1, 2, 3, and 4. SPLINE=0 disables spline differentiation. (Default is 0)

CONFIT	N	Variable that selects one of three dedicated finite differencing algorithms that are the most appropriate and most precise in the vicinity of conical intersections. Available for finite differences of ORDER=4 and 6 only. CONFIT=1, 2, and 3 select one dedicated algorithm each while CONFIT=0 selects the use of the default (non-dedicated) algorithm. (Default is 0)

CFCOND	N	Parameter controlling the activation of the CONFIT procedure. When one of the three CONFIT algorithms is requested, it will not be in action unless the condition number of the fit obtained with the CONFIT procedure is below CFCOND. The default is CFCOND=6.0. Smaller values effectively enable CONFIT to handle more displacements. Larger values effectively disable CONFIT except in the close vicinity of conical intersection. (Default is 6.0)

PRAXES	Controls the use of principal axes for the coordinate frame for finite differencing.	
	.TRUE.	Use principal axes.
	.FALSE.	Do not use principal axes. This value is recommended for symmetric molecules. (Default)

FNOISE	N	The threshold for automatic numerical noise detection. The automatic noise detection is available only for ORDER=4 or 6 differencing. Values larger than the default make the noise detection algorithm less sensitive to numerical noise. (Default is 1.0D-7)

TOL	N	The threshold used by the numerical gradient code during the elimination of redundant geometry displacements. It is generally not recommended to change the default value. (Default is 1.0D-6)

The last three options were introduced to facilitate debugging of the numerical gradient code. They are typically not useful to users, but can be used to verify the code.

CART	Flag that selects the use of either Cartesian coordinates or their symmetrized combinations in finite differencing.	
	.TRUE.	Use Cartesian coordinates.
	.FALSE.	Use the symmetrized combinations. (Default)

IASTRT IAEND	N	Integers defining the first (IASTRT) and last atom (IAEND) for which the partial gradient will be computed. These options can only be used with CART=.TRUE. These keywords are useful for verifying numerical gradients without performing the entire, more costly computation of the numerical gradient (in the normal way). The default is to use all atoms defined in $DATA.

$RAMAN group

The $RAMAN group can be used to control the calculations of Raman activities (RUNTYP=RAMAN) by numerical differentiation against an electric field.

EFIELD	N	The applied electric field strength. Recommended values are in the range 0.001 to 0.005 a.u. (Default is 0.002)

DDONLY	Instructs Firefly not to perform a Raman calculation but instead obtain only the dipole derivative tensor (required for obtaining IR intensities). Note that the dipole derivative tensor is already calculated at the end of a successful Hessian calculation.	
	.TRUE.	Calculate only the dipole derivative tensor. Do not calculate the alpha polarizability derivative tensor and do not print frequencies at the end of the job. This requires 7 gradient evaluations: 1 in the absence of electric field and 6 with the field turned on. This option was added as a rather cheap way to compute GAPT charges (Cioslowski, J. *J Am Chem Soc* **1989**, *111*, 8333) with Firefly. (Although the present implementation does not actually compute them, they can be easily computed manually using components of the printed dipole derivative tensor.)
	.FALSE.	Calculate the alpha polarizability derivative tensor. Calculate the dipole derivative tensor when the $DIPDR group is missing from the input. Print frequencies, IR intensities, and Raman activities at the end of the job. This requires 19 gradient evaluations: 1 in the absence of electric field and 18 with various components of the electric field turned on. (Default)

$CONIC group

This group provides fine control over the behavior of the $STATPT METHOD=CONIC Lagrange multiplier based CIs/ISCs optimizer called CONIC. In addition to the keyword of this group, CONIC is also controlled by keywords of the $STATPT group.

SHIFT	N	The final target value of $\Delta = E_j - E_i$ (in Hartree). For true CIs, it is almost impossible to reliably converge calculations to values of Δ less than approx. 0.0001 Hartree. For ISCs, it is possible to use tighter convergence criteria, up to SHIFT=1.0D-7 or 1.0D-8. The difference in convergence behavior between true CIs and ISCs is related to non-zero non-adiabatic coupling between quasi-degenerate states in the case of CIs and the absence of this coupling in the case of ISCs. (Default is 0.0001)

SHIFT0	N	The intermediate target value of $\Delta = E_j - E_i$ (in Hartree). For CIs, the recommended value is 0.001 Hartree. For ISCs, the recommendation is to always set SHIFT0 to the same value as SHIFT. The CONIC optimizer first tries to converge CIs/ISCs to Δ values equal to or below of SHIFT0 using loosened values of various convergence thresholds. During this process (which is internally called “phase 1”), the CONIC optimizer gathers information on the behavior of the particular system of interest. Upon achieving the initial convergence, the optimizer switches to the original values of various convergence thresholds and continues optimization trying to achieve tighter convergence with a value of Δ equal to or below SHIFT. This is internally called “phase 2”. Here, the previously gathered information is used to achieve the final convergence. In particular, the first few optimization steps of phase 2 attempt to rapidly decrease Δ using directed steps along the “most significant” modes while the rest of the steps take care of the final relaxation of the “less significant” modes. (Default is 0.0001)

HDGSS	N	The value to put on the diagonal of the initial diagonal guess to the Hessian matrix of the Lagrange function. It is ignored if internal coordinates are used for the optimization; in this case, Firefly's standard guess for the Hessian in internal coordinates is used. (Default is 1.0)

TDE	N	The value of the delta energy convergence threshold (in Hartree). It defines the first of the five tests, all of which needs to be positive for the optimization process to complete successfully. This first test is satisfied if |TDE| is less than |Δ - shift|. (Default is 1.0D-5)

TDXMAX	N	The value of the maximum primitive coordinate displacement convergence threshold. It defines the second of the five convergence tests. The displacement along each primitive coordinate should be less than TDXMAX. (Default is 20.0*OPTTOL of $STATPT)

TDXRMS	N	The value of the RMS coordinate displacement convergence threshold. It defines the third of the five convergence tests. The RMS displacement should be less than TDXRMS. (Default is 15.0*OPTTOL of $STATPT)

TGMAX	N	The threshold value for the maximum primitive component of the gradient of the Lagrange function. It defines the fourth of the five convergence tests. The component of the gradient along each primitive coordinate should be less than TGMAX. (Default is 5.0*OPTTOL of $STATPT)

TGRMS	N	The threshold value for the RMS gradient of the Lagrange function. It defines the last of the five convergence tests. The RMS gradient should be less than TGRMS. (Default is 3.0*OPTTOL of $STATPT)

STPSIZ	N	The maximum permissible RMS length of the geometry update step in Bohr. (Default is 0.05)

DEMAX	N	The maximum permissible change of the Lagrange function on every geometry update step (in Hartree). As the change in the Lagrange function can only be estimated, DEMAX is applied to this estimate instead of to the actual but unknown change. (Default is 0.03)

RESET	N	Parameter controlling whether and when to reset the current approximation to the Hessian of the Lagrange function. A reset is performed automatically using the value of RESET in some internal consistency checks. The default value for reset is 10.0. Smaller values will reset the approximate Hessian more often. To disable reset-based tests, specify RESET= –1.0. A disabled reset can result in either negative or positive effects on the convergence.

LSTART	N	Provides the initial value for Lagrange multiplier λ. A value of 0.0 instructs Firefly to automatically compute a starting approximation to λ rather than to use a user's input. This parameter can be used to facilitate restart of CONIC runs. (Default is 0.0)

LMAX	N	The upper bound on the absolute value of the Lagrange multiplier λ: $\|\lambda\| \leq$ LMAX. (Default is 3.0)

TRACK	Flag that provides control over the use of an additional state tracking procedure that is based on the analysis of the values of Lagrange multipliers. This additional tracking can correct mispredictions of the base tracking and detect the flip-flopping of states.	
	.TRUE.	Enable additional state tracking. (Default)
	.FALSE.	Disable additional state tracking.

STAGE2	Flag that instructs the CONIC code whether the CI/ISC optimization should be started directly from the "phase 2" of the algorithm (thus skipping the first phase) or not. This flag is useful for restarts.	
	.TRUE.	Start the CI/ISC optimization directly from the "phase 2" of the algorithm.
	.FALSE.	Do not start the CI/ISC optimization directly from the "phase 2" of the algorithm. (Default)

NDOWN	N	Defines the number of directed steps performed at the beginning of "phase 2" of the CONIC algorithm in order to decrease Δ to its final target value. The more steps performed the less the "strain" is on each particular step so that the molecular system of interest has more possibilities to relax during this procedure. (Default is 5)

RADOWN	N	Defines the relative decrease of Δ which the CONIC code attempts to achieve on each step at the beginning of "phase 2" of the CONIC algorithm in order to decrease Δ to its final target value. If non-zero, this triggers an alternative approach to the directed decrease of Δ, namely an exponential decrease of the following form: $\Delta_{next} = \Delta_{current}$ * (1.0 - RADOWN). The default value is 0.0, *i.e.* not to use this approach. Recommended values are in the range 0.1 to 0.2

TRDOWN	N	The multiplier for the TRMAN and TRMIN variables of the $STATPT group, which is applied when switching from "phase 1" to "phase 2" of the CONIC code. (Default is 0.2)

FILTER	N	The upper limit on the absolute values of the approximate Hessian matrix eigenvalues of the Lagrange function. Any values larger than FILTER are filtered off and zeroed. (Default is 5.0D+5)

DIABAS	Flag that directs the CONIC code whether or not to construct approximate diabatic states based on the extremals of the gradients and to use these diabatic states rather than adiabatic ones during the optimization of MECIs/MECPs. Note: with this option turned on, the value of λ at the MECI/MECP point will always be equal 1.0, so it can no longer be used to characterize the MECI/MECP type. This option is still experimental as of Firefly version 8.0.0.	

	.TRUE.	Construct approximate diabatic states based on the extremals of the gradients and use them during the optimization.
	.FALSE.	Use adiabatic states during the optimization. (Default)

TOLS	N1,N2,…	Array containing undocumented parameters defining various details of the CONIC algorithm's behavior. These parameters are implementation-dependent and are therefore not documented at this time.

MULTIW	This keyword is a duplicate of the MULTIW keyword of the $MCAVER group and exists for historical reasons. See the $MCAVER group for a description. It is recommended to use the $MCAVER version of this keyword. If specified, this keyword overrides the value of MULTIW of the $MCAVER group.

$FMM group

This group controls the quantum fast multipole method (QFMM) evaluation of Fock matrices. QFMM is turned on by the logical variable QFMM in the $INTGRL group. The defaults of the keywords in $FMM are reasonable, so there is little need to give this input.

ITGERR	N	Target error in final energy, to $10^{-ITGERR}$ Hartree. The accuracy is usually better than the setting of ITGERR. In fact QFMM runs should suffer no loss of accuracy or be more accurate than a conventional integral run. (Default is 7)

QOPS	A flag that provides control over the use of the Quantum Optimum Parameter Searching technique, which finds an optimum FMM parameter set.	
	.TRUE.	Use Quantum Optimum Parameter Searching. (Default)
	.FALSE.	Do not use Quantum Optimum Parameter Searching. See the note below.

If QOPS=.FALSE. the ITGERR value is not used. In this case the user should specify the following parameters:

NP	N	The highest multipole order for FMM. (Default is 15)

NS	N	The highest subdivision level. (Default is 2)

IWS	N	The minimum well-separateness. (Default is 2)

IDPGD	N	Point charge approximation error 10^{-IDPGD} of the Gaussian products. (Default is 9)

IEPS	N	Very fast multipole method (vFMM) error 10^{-IEPS}. (Default is 9)

The following are additional useful options which are either Firefly specific or not yet documented in the GAMESS (US) manual:

METHOD	Controls disk *vs.* CPU usage during the first (FMM) part of the calculation. At present, FULLDRCT is equivalent to SEMIDRCT. SEMIDRCT uses less disk space and is usually faster than DISK, especially for very large systems.	
	DISK	Use the disk-based method.
	SEMIDRCT	Use the semi-direct method. (Default)
	FULLDRCT	Use the fully direct method.

NUMRD	N	A positive integer that controls disk read caching during FMM, as well as the granularity of the static/dynamic load balancing during parallel QFMM runs. The default value is reasonable in most cases. (Default is 10)

MODIFY	A flag to allow the QOPS code to modify the ICUT and ITOL variables of $CONTRL group.	
	.TRUE.	Allow. Though it provides better compatibility with GAMESS (US), it is generally not recommended to use this option.
	.FALSE.	Do not allow. (Default)

MQOPS	Determines whether it is allowed to manually input SCLF and NS when QOPS=.TRUE.	
	0	SCLF and NS are determined automatically. (Default)
	1	User provided values of SCLF and NS override those found by QOPS.

SCLF	N	The FMM cube scaling factor must be greater or equal to 1.00. (Default is 1.00)

STATIC	Provides control over the use of static load balancing (SLB) during FMM part of calculations even if DLB is activated. The default is .TRUE. because in the case of homogeneous environment the static load balancing is implemented much more efficient.	
	.TRUE.	Enable SLB. (Default)
	.FALSE.	Disable SLB.

NEARJ	Selects the routine used to calculate near field Coulomb terms.	
	0	Selects an optimal default based on FSTINT & REORDR settings in $CONTRL. (Default)
	1	Selects use of a bugfixed/improved GAMESS (US)-based routine using the HONDO integral package.
	2	Selects use of a Firefly specific routine based on the fastints code, which is generally much faster. This option requires FSTINT and REORDR to be set in $CONTRL.

LEX	Selects the routine used to calculate HF exchange terms.	
	0	Selects an optimal default based on the FSTINT & REORDR settings in $CONTRL, as well as on the molecular symmetry. (Default)
	1	Selects the use of a bugfixed/improved GAMESS (US)-based routine using HONDO integral package. One should take into account that this implementation of linear exchange is by design to some degree approximate and is not fully equivalent to direct SCF, although in most cases one can safely neglect this fact.
	2	Selects the use of a fastints based routine which evaluates some extra integrals but is the only part of the QFMM which can take into account the molecular symmetry. It is strictly equivalent to the direct SCF. For highly-symmetrical systems, this is faster than any other available method. It requires FSTINT to be set in $CONTRL.
	3	Selects the use of a fastints based routine which evaluates the minimal number of all the necessary two-electron integrals and is also strictly equivalent to direct SCF. It does not exploit molecular symmetry though. For low-symmetry systems, this is the fastest method available. It requires FSTINT and REORDR to be set in $CONTRL.

SKIP1	A flag that modifies the behavior of the inner loop of the two-electron integral selection code of the LEX=1 exchange routine. See also the description of the STRICT keyword.	
	.TRUE.	GAMESS (US)-style behavior. (Default)
	.FALSE.	Makes the loop more precise at the cost of some CPU overhead.

SKIP2	A flag that modifies the behavior of the outer loop of the two-electron integral selection code of the LEX=1 exchange routine. See also the description of the STRICT keyword.	
	.TRUE.	GAMESS (US)-style behavior. (Default)
	.FALSE.	Makes the loop more precise at the cost of some CPU overhead.

STRICT	A flag that modifies the behavior of the density matrix sorting and nonzero elements selection for the LEX=1 exchange routine. The default is .FALSE. because even if one sets SKIP1=.FALSE. SKIP2=.FALSE. STRICT=.TRUE. the LEX=1 routine is not exactly equivalent to direct SCF while the CPU overhead is very significant.	
	.TRUE.	Make the density matrix sorting and nonzero elements selection more precise at the cost of some CPU overhead.
	.FALSE.	GAMESS (US)-style behavior. (Default).

$SCF group

This group provides additional control over the Hartree-Fock, DFT, or GVB SCF steps. Some of its keywords are required for GVB open shell or perfect pairing wavefunctions.

Important: unless indicated otherwise (such as for the DIIS and SOSCF keywords), the terms ‘RHF’, ‘ROHF’, and ‘UHF’ refer to both Hartree-Fock and DFT wavefunctions.

DIRSCF	Specifies whether the SCF type is direct or conventional.	
	.TRUE.	Direct SCF calculation. For every iteration, the integrals are recomputed as needed. This options allows use of the fastints code (see $CONTRL). This keyword also selects direct MP2, MP3, or MP4 computations, if requested.
	.FALSE.	Conventional SCF calculation. The integrals are stored on the disk. (Default)

FDIFF	Flag used to request incremental Fock matrix formation. This will compute only a change in the Fock matrices since the previous iteration, rather than recomputing all two-electron contributions. This saves much CPU time in the later iterations as small contributions are neglected (as controlled by various cutoff thresholds – of these, the ICUT of $CONTRL is the most important one). Incremental Fock matrix formation pertains only to direct SCF and is implemented only for the RHF, ROHF, and UHF. Cases with many diffuse functions in the basis set sometimes oscillate at the end rather than converging. Firefly is able to detect this and will automatically turns off the FDIFF option when needed, so there is usually no need to turn this option off manually. The use of this detection is controlled through the FDCTRL option below.	
	.TRUE.	Use incremental Fock matrix formation. (Default for SCFTYP=RHF, ROHF, and UHF)
	.FALSE.	Do not use incremental Fock matrix formation. (Enforced for SCFTYP=GVB)

FDCTRL	Flag used to request automatic control over incremental Fock matrix formation code.	
	.TRUE.	Turn off incremental Fock matrix formation as appropriate. (Default)
	.FALSE.	Do not turn off incremental Fock matrix formation automatically.

XFDIFF	Flag used to request a variation of the incremental Fock matrix formation approach which is based on the orthogonal factorization of the density matrix. This approach can be a bit faster than the standard FDIFF code. This flag does not have effect if FDIFF is turned off. This option is programmed only for RHF.	
	.TRUE.	Use orthogonal factorization of the density matrix during incremental Fock matrix formation.
	.FALSE.	Do not use orthogonal factorization of the density matrix during incremental

		Fock matrix formation (Default).

XFDNR	N	The maximum number of re-orthogonalizations to be used by the XFDIFF code. Values larger than the default may improve stability somewhat by the cost of additional CPU overhead. (Default is 2)

UHFNOS	Flag controlling generation of the natural orbitals of a UHF wavefunction.	
	.TRUE.	Generate natural orbitals of the UHF wavefunction.
	.FALSE.	Do not generate natural orbitals of the UHF wavefunction. (Default)

MVOQ	N	Form modified virtual orbitals, using a cation with N electrons removed. Implemented for restricted, restricted open, and GVB wavefunctions. If necessary to reach a closed shell cation, the program might remove N+1 electrons. A typical value of N is 6. A value of 0 skips the formation of MVOs. (Default is 0)

NPUNCH	SCF punch option.	
	0	Do not punch out the final orbitals.
	1	Punch out the occupied orbitals. For GVB wavefunctions, orbitals in geminal pairs will be punched out as well.
	2	Punch out the occupied and virtual orbitals. (Default)

JKMAT	Controls the optional formation and printout of closed shell J and K matrices. This option is mainly intended for debug purposes. It is currently implemented for RHF only.	
	.TRUE.	Enable this optional formation and printout.
	.FALSE.	Disable this optional formation and printout. (Default)

The next options control the SCF convergence criteria.

NCONV	N	SCF density convergence threshold. Convergence is reached when the density change between two consecutive SCF cycles is less than $10^{(-N)}$. One more cycle is executed after reaching convergence. Note that accuracy higher than the default is sometimes needed, especially in cases with a partial linear dependency in the basis. Insufficient accuracy gives questionable gradients. Note that the DENTOL keyword (see below) can be used if finer control over the density convergence threshold is desired. (Default is 5, except for CI or MP2 gradients where it is 6)

ENGTHR	N	Energy convergence threshold for SCF, in Hartrees. (Default is 1.0D-9)

DIITHR	N	DIIS error convergence threshold for SCF, in Hartrees. (Default is 1.0D-7)

SOGTHR	N	SOSCF gradient convergence threshold for SCF, in Hartrees. (Default is 1.0D-12)

DENTOL	N	Maximum density matrix change convergence threshold for SCF. This keyword has been implemented to replace the NCONV keyword, providing finer control over the SCF density convergence threshold. A value of 0.0 means to use the value of NCONV to compute DENTOL such that DENTOL = $1.0*10^{-NCONV}$ (making the two keywords specify the same threshold). Valid values are from 1.0D-4 to 0.0. (Default is 0.0)

The next options control the SCF convergence procedure. Note that when either DIIS or SOSCF is specified, any other accelerator is put in abeyance. If DIIS and SOSCF are both specified, only SOSCF will be used.

DIIS	Controls the use of Pulay's DIIS extrapolation method. Note that DIIS can only be used for GVB wavefunctions with NPAIR=0 or NPAIR=1.	
	.TRUE.	Enables use of the DIIS method. (Default for unrestricted wavefunctions, non-Abelian group ROHF wavefunctions, and DFT)
	.FALSE.	Disables use of the DIIS method. (Default for RHF, Abelian group ROHF, GVB, and R and RO semiempirical wavefunctions)

SOSCF	Controls the use of the second order SCF orbital optimization method.	
	.TRUE.	Enables use of the SOSCF method. (Default for RHF, Abelian group ROHF, GVB, and R and RO semiempirical wavefunctions)
	.FALSE.	Disables use of the SOSCF method. (Default for unrestricted wavefunctions, non-Abelian group ROHF wavefunctions, and DFT)

EXTRAP	Controls the use of Pople extrapolation of the Fock matrix.	
	.TRUE.	Enables Pople extrapolation. (Default)
	.FALSE.	Disables Pople extrapolation.

DAMP	Controls the use of Davidson damping of the Fock matrix. The damping factor will be determined automatically, but can be controlled using other keywords (see below).	
	.TRUE.	Enables Davidson damping. (Default for DFT)

	.FALSE.	Disables Davidson damping. (Default for HF, GVB, and semi-empirical wavefunctions)

SHIFT	Controls the use of variable level shift of the Fock matrix. This option is cannot be used with the DIIS and SOSCF convergers.	
	.TRUE.	Enables level shifting. (Default for DFT wavefunctions.)
	.FALSE.	Disables level shifting. (Default for non-DFT wavefunctions.)

FSHIFT	N	Specifies a fixed level shift to be used during SCF iterations (in Hartree). Note: unlike SHIFT level shifting, FSHIFT can be used together with DIIS. The default value of 0.1 is appropriate in most cases. Use of larger values of FSHIFT can be helpful in converging difficult SCF cases. While larger values of FSHIFT may help diverging SCF to converge, they usually negatively impact the SCF convergence rate. FSHIFT=0.0 completely disables fixed level shifting. This option is not programmed for GVB. Use of FSHIFT disable level shifting with SHIFT. (Default is 0.1)

RSTRCT	Controls the use of restriction for orbital interchanges.	
	.TRUE.	Restrict orbital interchanges.
	.FALSE.	Do not restrict orbital interchanges. (Default)

DEM	Controls the use of the direct energy minimization method. This option has been implemented only for restricted wavefunctions (DFT wavefunctions are not allowed though). In addition, it can only be used with conventional SCF.	
	.TRUE.	Enables the use of direct energy minimization.
	.FALSE.	Disables the use of direct energy minimization. (Default)

The next options provide further control over the SCF convergence procedure.

FSTDII	Flag used to request the use of the new, faster code in DIIS computations. New code forms DIIS error vector significantly faster than an older, standard DIIS code. This flag does not have effect if DIIS is turned off. This option not programmed for GVB.	
	.TRUE.	Use the new code in DIIS computations. (Default for RHF, ROHF, and UHF)
	.FALSE.	Do not use the new code in DIIS computations. (Default for GVB)

ETHRSH	N	Energy error threshold for initiating DIIS. The DIIS error is the largest element of e=FDS-SDF. Increasing the value of ETHRSH forces DIIS to be activated sooner.

		(Default is 0.5 Hartree)

MAXDII	N	Maximum size of the DIIS linear equations, so that at most MAXDII-1 Fock matrices are used in the interpolation. (Default is 10)

DIIMOD	N	Selects the particular algorithm to be used to remove old error vectors from the DIIS subspace. Valid values are 0 to 5. One can experiment with this setting in the case of poor DIIS convergence. (Default is 0)

DIIERR	N	Selects the particular form of DIIS error vector to use. Valid values are 0 and 1. A value of 0 instructs Firefly to use a standard FDS-SDF SCF DIIS error vector. A value of 1 is experimental. One can experiment with this setting in the case of bad DIIS convergence. (Default is 0)

DIITOL	N	The threshold used to remove quasi-linearly dependent vectors from DIIS subspace. Values below the default value may remove vectors more often. (Default is 100.0)

SOGTOL	N	Second order gradient threshold. SOSCF will be initiated when the orbital gradient falls below this threshold. (Default is 0.25 Hartree)

SODIIS	Flag used to request the use of a combination of the DIIS and SOSCF convergence accelerators. This option is programmed only for RHF and is currently experimental, so it should normally not be used.	
	.TRUE.	Use SOSCF+DIIS combo code.
	.FALSE.	Do not use SOSCF+DIIS combo code. (Default).

DIONCE	Flag used to request DIIS for the first SCF procedure and SOSCF for all subsequent SCF procedures, *e.g.* at subsequent geometries during geometry optimization. This option is currently programmed only for RHF. This option requires the DIIS option to be turned on.	
	.TRUE.	Use DIIS at first and SCF at all subsequent SCFs.
	.FALSE.	Always use DIIS if it is turned on. (Default)

DEMCUT	N	Direct energy minimization threshold. DEM will not be done once the density matrix change falls below this threshold. (Default is 0.5)

DMPCUT	N	Damping factor lower bound cutoff. The damping factor will not be allowed to drop below this value. Note that the damping factor does not have to equal zero to achieve valid convergence (see Hsu, Davidson and Pitzer, *J. Chem. Phys.* **1976**, *65*, 609, especially the section on convergence control), but it should not be astronomically high either. (Default is 0.0)

The next four options apply to virial scaling. This has been implemented for SCFTYP=RHF, ROHF, and UHF (though not for DFT wavefunctions), with RUNTYP=ENERGY, OPTIMIZE, and SADPOINT. For more information on this functionality, which is most economically employed during a geometry search, see M. Lehd and F. Jensen, *J. Comput. Chem.* **1991**, *12*, 1089-1096.

VTSCAL	Flag that requests that the virial theorem be satisfied. An analysis of the total energy as an exact sum of orbital kinetic energies is printed.	
	.TRUE.	The virial theorem is to be satisfied.
	.FALSE.	The virial theorem does not have to be satisfied. (Default)

SCALF	N	Initial exponent scale factor when VTSCAL is in use, useful when restarting. (Default is 1.0)

MAXVT	N	Maximum number of iterations (at a single geometry) to satisfy the energy virial theorem. (Default is 20)

VTCONV	N	Convergence criterion for the VT, which is satisfied when 2<T> + <V> + R * dE/dR is less than VTCONV. (Default is 1.0D-6 Hartree)

The next parameters define the GVB wavefunction. Note that ALPHA and BETA also have meaning for ROHF. See also MULT in the $CONTRL group. The GVB wavefunction assumes orbitals are in the order core, open, pairs. Note that the defaults for F, ALPHA, and BETA depend on the state chosen. Defaults for the most commonly occurring cases are stored internally. See the chapter on GVB in the Firefly manual for examples of other cases.

NCO	N	The number of closed shell orbitals. The default should almost certainly be changed! (Default is 0)

NSETO	N	The number of sets of open shells in the function. The maximum allowed value is 10. (Default is 0)

NO	N1,N2,…	Array giving the degeneracy of each open shell set. (Default is 0,0,...)

NPAIR	N	The number of geminal pairs in the GVB wavefunction. The maximum allowed value is 12. A value of 0 corresponds to open shell SCF. (Default is 0)

CICOEF	N1,N2,…	Array of ordered pairs of CI coefficients for the GVB pairs. For example, a two pair case for water could be CICOEF(1)=0.95,-0.05,0.95,-0.05. If not yet normalized (as in the default), CICOEF will be automatically normalized. This parameter is useful for restarting a GVB run with the current CI coefficients. (Default is 0.90,-0.20,0.90,-0.20,...)

COUPLE	Flag controlling the input of F, ALPHA, and BETA.	
	.TRUE.	Use input for F, ALPHA, and BETA.
	.FALSE.	Ignore input for F, ALPHA, and BETA. (Default)

F	N1,N2,…	Array of fractional occupations.

ALPHA	N1,N2,…	Array of A coupling coefficients given in lower triangular order (*i.e.* $alpha_{11}$,$alpha_{21}$,$alpha_{22}$,$alpha_{31}$,$alpha_{32}$,$alpha_{33}$,$alpha_{41}$,...).

BETA	N1,N2,…	Array of B coupling coefficients given in lower triangular order.

$DFT group

This group contains keywords relevant when the DFTTYP keyword of the $CONTRL group is specified, *i.e.* for DFT runs, TDDFT runs, and for any other runs which are directly or indirectly affected by DFTTYP.

NRAD	N	The default number of radial points per atom. (Default is 63)

NRDATM	N1,N2,…	Integer array of up to 128 elements. This array can be used to change the actual value of NRAD on a per atomic number basis. For example, setting NRDATM(6)=99 will apply a radial grid of 99 points to all carbon atoms. The entry NRDATM(128) is used for dummy atoms with no charge. The default is to use values equal to NRAD for all elements.

LMAX	The maximum order of the Lebedev angular grid to use. Below is a list of allowed orders and the number of points per radial shell associated.	
	3	6 points per radial shell
	5	14
	7	26
	9	38
	11	50
	13	74
	15	86
	17	110
	19	146
	21	170
	23	194
	25	230
	27	266
	29	302 (Default)
	31	350
	35	434
	41	590
	47	770
	53	974
	59	1202
	65	1454
	71	1730
	77	2030
	83	2354
	89	2702
	95	3074
	101	3470
	107	3890
	113	4334

	119	4802
	125	5294
	131	5810

LMIN	N	The minimum allowed order of the Lebedev angular grid to use. The default value is computed based on the basis set properties and minimizes some errors that arise from to the use of numerical quadratures. Normally, one does not need to change the default value. Furthermore, it is not recommended to decrease it.

LMXATM	N1,N2,…	Integer array of up to 128 elements. This array can be used to change the value of LMAX on a per atomic number basis. For example, setting LMXATM(1)=29 will apply an angular grid of 29^{th} order to all hydrogen atoms. The entry LMXATM(128) is used for dummy atoms with no charge. The default is to use values equal to LMAX for all elements.

ANGPRN	Flag that activates/deactivates angular grid pruning as a function of the radius through the use of a scheme similar to the one proposed by Murray, Handy and Laming (MHL).	
	.TRUE.	Use angular grid pruning. (Default)
	.FALSE.	Do not use angular grid pruning. This will slow down the calculations but will result in more accurate results.

KAP	N	The parameter K_{theta} used for angular pruning. The default value is 5.0 as recommended by MHL. Increasing this value to, say, 10.0 will improve precision at the cost of performance. This parameter has no effect if angular pruning is not used. (Default is 5.0)

RADPRN	Flag that activates/deactivates radial grid pruning. Note: in order to get correct results for systems having atoms with no associated basis, one must disable radial pruning.	
	.TRUE.	Use radial grid pruning. (Default)
	.FALSE.	Do not use radial grid pruning. This will slow down calculations to some degree but may improve precision.

RMXATM	N1,N2,…	Integer array of up to 128 elements. This array can be used to change the atomic cutoff radii which are used during radial pruning, on a per atomic number basis. For example, setting RMXATM(8)=6 will set the effective radius of all oxygen atoms to 6 Ångstrom. The entry RMXATM(128) is used for dummy atoms with no charge. The default values are probably much larger than is actually necessary and depend on the basis set used.

CUTOFF	N	Contributions to the DFT Fock matrix due to batch of angular points are ignored if they are smaller than CUTOFF by an absolute value. (Default is 1.0D-10)

CUTAO	N	If the absolute value of an atomic basis function is smaller than CUTAO, it will be set to zero during calculations. (Default is 1.0D-10)

CUTWGT	N	If the absolute value of a weight associated with a grid point is less than CUTWGT, this point will not be taken into account during DFT calculations. (Default is 1.0D-20)

CUTORB	N	Contributions to the DFT Fock matrix due to batch of orbitals are ignored if they are smaller than CUTORB by an absolute value. (Default is 1.0D-15)

CUTGG1 CUTGG2 CUTGG3 CUTGG4	N	Various cutoffs used during the calculation of the grid weights derivatives contributions to the molecular gradients. The default values are 1.0D-13, 1.0D-13, 1.0D-13, and 1.0D-30, respectively, and are probably too strict.

B3LYP	Selects which of the VWN functionals should be used in the B3LYP functional. Note that this choice can also be made by setting DFTTYP as either B3LYP1 or B3LYP5.	
	NWCHEM	Use VWN formula 1 RPA (sometimes referred to as VWN formula 3). This makes the B3LYP functional identical to the one used in NWChem and Gaussian. (Default)
	GAMESS	Use VWN formula 5. This makes the B3LYP functional identical to the one used in GAMESS.

O3LYP	Provides control over the weight of non-local exchange in the O3LYP1 and O3LYP5 functionals. This option exists because there are some ambiguities in O3LYP-related papers. It cannot be said which of the two weights is correct, instead, these two implementations should be considered as two different functionals.	
	DEFAULT	Set the weight so that the resulting functionals are identical to those used by various QC programs. (Default)
	GAUSSIAN	Set the weight so that the resulting functionals are identical to the ones used in Gaussian 03 Rev D.01 and above.

HFX	N1,N2	The weight of the exact (*i.e.* Hartree-Fock) exchange. This option is supported only for hybrid functionals. The default value is set as is appropriate for a particular value of DFTTYP. For non-self-consistent double-hybrid functionals an extended form of this keyword exists, namely HFX(2), which allows one to change the weight of the exact exchange in the parent DFT functional (see the description of PARENT keyword below).

PRJGRD	Flag that requests a special treatment of the computed DFT gradient so that the contributions of translational and rotational contaminants to the gradient are projected out and eliminated.	
	.TRUE.	Project out these contributions. This increases the precision of computed

		gradients. (Default)
	.FALSE.	Do not project out these contributions.

METHOD	Selects the method used to construct DFT contributions to the Fock matrix.	
	0	Use an adaptive strategy based on both density matrix and MOs. (Default)
	1	Use a density-matrix driven method.
	2	Use an MO driven method.

FIXP81	Flag that pertains to the Perdew-Zunger 1981 LDA correlation. In the paper defining this functional, its parameters were given with only four digits. As there are two branches of the Ec fit, the two parameters of this functional were selected to be such a functions of other parameters that Ec and Vc are globally continuous. However, the four-digit precision is not enough for this purpose as it causes small deviations from the exact continuity which, in turn, results in errors such as non-precise gradients. Tight geometry optimizations in particular are not possible with this functional. By default, Firefly uses parameters redefined with a 15 digit precision, thus allowing the globally continuous merging of the two branches of the fit and achieving a smooth Ec and Vc. This pertains to P81LDA as well as to all functionals using the P81 local correlation functional. However, this corrected fit does result in a slightly different functional.	
	.TRUE.	Use 15 digit precision. (Default)
	.FALSE.	Do not use the 15 digit precision. The causes the functional to reverted to its original, slightly discontinuous form.

CPT2	N	The overall amount of the MP2-like correlation to be used as a part of a double hybrid functional. This option pertains only to double hybrid functionals. The default value is set as is appropriate for a particular value of DFTTYP.

SCS	N1,N2	Array of two elements that provides control over the scaling of the individual contributions of the singlet and triplet pairs to the MP2-like energy. SCS(1) defines a scaling factor for the contributions from singlet pairs (a.k.a. the spin opposite part of the MP2-like term). SCS(2) defines a scaling factor for the contributions from triplet pairs (a.k.a. the same spin part of the MP2-like term). The two parts of an MP2-like term are first multiplied by the corresponding scaling factors, the results are added, and sum is multiplied by CPT2 to form the final contribution to the DFT energy. This option pertains to double hybrid functionals only. The default values of SCS are set as is appropriate for a particular value of DFTTYP.

PARENT	For non-self-consistent double-hybrid functionals, this keyword defines the parent DFT functional that is used to generate the MOs and density. This keyword can be used to give the name of any supported non-local functional. The default value is PARENT=DEFAULT which means to use the parent functional used in the original formulation of the double-hybrid functional used. This option pertains to non-self-consistent double hybrid functionals only.

CF3	N	Allows one to modify some of the complex functionals in Firefly. The particular meaning of this keywords depends on the functional. Historically, this was “coefficient #3” thus the name of variable.

$DFTD group

This group controls the various parameters and settings of Grimme's empirical dispersion correction scheme (a.k.a. DFT-D). The DFT-D scheme is enabled by setting DFTD=.T. in $CONTRL. Note that a few DFT functionals, when selected, automatically enable the use of a specific version of DFT-D.

VERSN	Selects the version of the DFT-D scheme.	
	2	DFT-D version 2.
	3	DFT-D version 3. (Default)
	4	DFT-D version 3 with Becke-Johnson damping. This version will be referred to as version 4 for the remainder of this keyword list.

The following keywords specify the most important parameters of the correction model. Optimal parameters recommended by Grimme have been implemented for various functionals (see the manual for an overview of functionals for which parameters are available). As such, the default values of these keywords depend on the functional and DFT-D scheme version that have been specified. It is important to note that, when specifying a custom set of parameters, all five of the above parameters should be given a value, even those which are zero or not used by the DFT-D version used!

S6	N	The s_6 global scaling factor, the main scaling factor in DFT-D version 2. For DFT-D version 3 and 4, this parameter is of lesser importance and is usually set to 1.0 (except for double hybrid functionals).

RS6	N	For DFT-D version 2, this parameter is used in calculating the dampening factor – its value is 1.1 (for all functionals). For DFT-D version 3, this parameter corresponds to the $s_{r,6}$ scaling factor, which is the main scaling factor in this version. For DFT-D version 4, it corresponds to the α_1 free fit parameter.

S18	N	The s_8 scaling factor used in DFT-D version 3 and 4. For DFT-D version 2, this parameter has no function.

RS18	N	For DFT-D version 3, this parameter is used in calculating the dampening factor, its value being 1.0 except with Slater exchange (where its value is 0.697). For DFT-D version 4, this parameter corresponds to the α_2 free fit parameter. For DFT-D version 2, this parameter has no function.

ALP	N	The global scaling parameter of the damping function (which dampens the dispersion correction at short ranges). Its value is usually 20 for DFT-D version 2, and 14 for DFT-D version 3 and 4. For DFT-D version 4, this parameter is only used when ABC=.T. (see below).

The following four keywords are aliases for some of the parameters above.

SR6	N	An alias for the RS6 keyword.

S8	N	An alias for the S18 keyword.

ALPHA1	N	An alias for the RS6 keyword.

ALPHA2	N	An alias for the RS18 keyword.

The remaining keywords in this group are used less often.

K1	N	The k_1 parameter, which controls the coordination number dependent dispersion. (Default = 16)

K2	N	The k_2 parameter, which controls the coordination number dependent dispersion. (Default = 1.3333333)

K3	N	The k_3 parameter, which controls the coordination number dependent dispersion. (Default = -40)

ABC	Flag that enables/disables the three-body non-additive contribution to the dispersion correction. Analytical gradients are not available for DFT-D with the three-body contribution enabled.	
	.TRUE.	Enable the three-body non-additive contribution. This forces the use of numerical gradients.
	.FALSE.	Disable the three-body non-additive contribution. (Default)

TZ	Flag that selects the use of a special set of DFT-D version 3 parameters optimized for Ahlrichs' TZVPP basis set. These parameters are available for a limited set of functionals only. See the main chapter on DFT-D in the manual for an overview.	
	.TRUE.	Enable the use of the special DFT-D3 parameters.
	.FALSE.	Disable the use of the special DFT-D3 parameters. (Default)

RTHR	N	A threshold that can speedup DFT-D calculations on large molecules. (Default is 20000.0D0)

RHTR2	N	A threshold that can speedup DFT-D calculations on large molecules. (Default is 1600.0D0)

NG	Flag which requests numerical DFT-D gradients. As analytical DFT-D gradients are available, the default is .FALSE.	
	.TRUE.	Calculate gradients numerically.
	.FALSE.	Calculate gradients analytically. (Default)

$MP2 group

The $MP2 group controls second order Møller-Plesset perturbation theory (MP2) runs, requested by MPLEVL=2 in $CONTRL. In addition, this group also controls the ALTTRF integral transformation stage of MCQDPT2 and XMCQDPT2 calculations.

MP2 energies are available for RHF, high spin ROHF, and UHF wavefunctions. Analytic gradients and the first order correction to the wavefunction (*i.e.* properties) are available for RHF only. The old-style and improved serial MP2 gradient programs are also controlled by the $MP2 group. See the description of the $MP2GRD group for control over the new MP2 energy gradient program. See the DIRSCF keyword in the $SCF group to select direct MP2.

See the $MCQDPT and $XMCQDPT groups for the second order perturbation theory based on CASSCF wavefunctions. See the $MP3 and $MP4 groups for third (MP3) and fourth (MP4) order Møller-Plesset perturbation theory.

NCORE	N	Omits the first N occupied alpha and beta orbitals from the calculation. The default equals is the number of chemical core orbitals.

METHOD	Selects one of several available MP2 programs (*i.e.* the integral transformation method).	
	1	Selects the use of the new RHF/ROHF/UHF MP2 energy program. This program is intended to handle large systems (*e.g.* 500 AOs or more). It is semi-direct, very fast, and requires much less memory compared to other MP2 methods. It is definitely the method of choice for large jobs. It can be used with either conventional or direct SCF. As this method requires the 2-electron AO integrals to be reevaluated four times, there can be a considerable overhead for small jobs. This is the only reason why this method is not turned on by default.
	2	Selects the serial or parallel segmented MP2 energy integral transformation. This transformation can be either conventional or direct, see the DIRSCF keyword in $SCF. This is the default method for RHF runs, provided enough memory is available. If this however is not the case, Firefly will try to use METHOD=3.
	-2	For UHF and ROHF, selects the modified serial or parallel segmented MP2 energy integral transformation which requires approximately half of memory required with METHOD=2. This transformation can be either conventional or direct, see the DIRSCF keyword in $SCF. This is the default method for ROHF and UHF runs, provided enough memory is available. If this however is not the case, Firefly will try to use METHOD=3.
	3	Alternative conventional out-of-core MP2 energy transformation (which can be used in serial runs only). It requires more disk, but less memory than the METHOD=2 transformation. It cannot run in direct mode. This method will be used if the default METHOD=2 or -2 cannot be run due to not enough memory being available.

NWORD	N	Controls the memory usage of the MP2 method, in words. The default value for NWORD is 0, which means that all available memory can be used.

CUTOFF	N	Transformed integral retention threshold. (Default is 1.0D-9).

MP2PRP	Flag that requests the calculation of properties for RHF MP2 jobs with RUNTYP=ENERGY. This is appreciably more expensive than just evaluating the 2nd order energy correction alone. Properties are always computed during gradient runs, when they are an almost free byproduct.	
	.TRUE.	Calculate properties.
	.FALSE.	Do not calculate properties. (Default)

LMOMP2	Requests Firefly to analyze the closed shell MP2 energy in terms of localized orbitals. Any type of localized orbital may be used. This option is implemented for RHF only, and its selection forces the use of the METHOD=3 transformation.	
	.TRUE.	Enables analysis in terms of localized orbitals.
	.FALSE.	Disables analysis in terms of localized orbitals. (Default)

OSPT	Selects the type of open shell spin-restricted perturbation. This parameter applies only to run with SCFTYP=ROHF.	
	RMP	Selects RMP (a.k.a ROHF-MBPT) perturbation theory. (Default)
	ZAPT	Selects Z-averaged perturbation theory. At present, ZAPT jobs can only be run in serial and can only use the METHOD=2 integral transformation.

AOINTS	Selects the method for AO integral storage during conventional integral transformations in parallel runs. The default value is defined by the AOINTS keyword of $SYSTEM group.	
	DUP	Stores duplicated AO lists on each node. Recommended for parallel computers with slow inter-node communication, e.g. Ethernet.
	DIST	Distributes the AO integral file across all nodes. Recommended for parallel computers with high speed communications.

SCS	N1,N2	Array of two elements that provides control over the scaling of the individual contributions of the singlet and triplet pairs to the MP2 energy. SCS(1) defines a scaling factor for the contributions from singlet pairs (a.k.a. the spin opposite part of the MP2-like term). SCS(2) defines a scaling factor for the contributions from triplet pairs (a.k.a. the same spin part of the MP2-like term). To compute SCS-MP2 energies, use SCS(1)=1.2,0.333333333333333. The use of non-default scaling factors forces METHOD=1 MP2 energy code. Gradient are available only for MP2 without scaling. The default is SCS(1)=1.0,1.0, *i.e.* not to use scaling.

The settings below pertain only to the METHOD=1 MP2 energy program and the ALTTRF integral transformation.

DIRECT	Provides control over the handling of AO integrals.	
	.TRUE.	Selects the fully direct handling of AO integrals. This somewhat reduces the amount of disk space needed at the cost of extra CPU time.
	.FALSE.	Selects the semidirect handling of AO integrals. (Default)

PACKAO	Provides control over the packing of AO integrals.	
	.TRUE.	Activates the packing of AO integrals during METHOD=1 semidirect MP2 calculations, thus slightly reducing the amount of disk space needed. (Default)
	.FALSE.	Do not pack AO integrals.

MNRECL	N	The minimum allowed record size (in 12-byte elements) for the direct access file used during METHOD=1 MP2 calculations. Reducing the value of MNRECL somewhat increases the job execution time but decreases the amount of memory needed for calculations. It is not recommended to set MNRECL below 500. (Default is 20000)

SVDISK	This option selects whether the ROHF MP2 program may use some extra disk space if this is needed to reduce the job CPU time.	
	.TRUE.	Disable the use of extra disk space.
	.FALSE.	Allow the ROHF MP2 program to use some extra disk space. (Default)

SPARSE	This options controls how the matrix-matrix multiplication is performed during MP2 METHOD=1 runs. By default, Firefly tries to automatically determine and use the most appropriate strategy. Note that the automatic selection may not work well in some cases!	
	.TRUE.	Selects the use of special matrix-matrix multiplication routines. This strategy is optimal for a sparse list of two-electron integrals, *i.e.* for large molecules. In addition, use the of these routines always requires less memory compared to the amount needed with SPARSE=.FALSE.
	.FALSE.	Selects the use of standard BLAS level 3 routines. This is the best choice when the list of two-electron integrals is dense, *i.e.* for small to medium size molecules.

The settings below pertain only to the MP2 gradient programs. There are three different MP2 gradient programs in Firefly: the old one, the improved one, and the new one. The *old MP2 gradient program* is less efficient except for small systems and does not run in parallel. The *improved MP2 gradient* program is usually much more efficient and requires less memory but does not run in parallel. The *new MP2 gradient program* is the most efficient for large systems, uses even less memory, and runs perfectly in parallel using the P2P interface.

GRDMET	Provides control over the selection of the MP2 gradient program. GRDMET=2 uses less memory and is usually faster, but for some combinations of computer systems and particular tasks GRDMET=1 can be the preferred option.

	1	If METHOD=1, selects alternative strategy for the new MP2 gradient program. Otherwise, selects old style MP2 gradient program.
	2	If METHOD=1, selects the default strategy for the new MP2 gradient program. Otherwise, selects improved MP2 gradient program. (Default)

DM2MET	Selects one of seven programmed methods for the calculation of the non-separable part of the MP2 two-body density matrix (DM2). Relevant to the improved MP2 gradient program only. Methods -3, -2, and -1 are precisely the same as methods 3, 2, and 1, respectively, but use asynchronous disk I/O to eliminate I/O latency and to reduce the total execution time. Which methods is the fastest depends on the amount of physical memory installed. One can experiment with this option to find the optimal settings for a particular task type and environment. Relevant only to the improved MP2 gradient program.	
	3 or -3	These methods require the least memory.
	2 or -2	These methods require the least CPU time, but probably not the least total time.
	0	This method is definitely the slowest one.
	1 or -1	These methods are designed as a reasonable compromise. (Default)

MINMEM	Flag to optionally reduce memory needs for the improved MP2 gradient program at the cost of extra I/O.	
	.TRUE.	Use extra I/O to decrease memory needs.
	.FALSE.	Do not use extra I/O to decrease memory needs. (Default)

MEMGRD	N	The maximum amount of memory (in words) to be used during the MP2 gradient calculation. This option controls only the last stage of calculation, namely the evaluation of the gradient integrals. A value of 0 means to use all available memory. Relevant only to the improved MP2 gradient program. (Default is 0)

MXRECL	N	The maximum record size for the direct access file used during METHOD=3 MP2 energy runs and old-style and improved MP2 gradient runs. (Default is 65536 words)

MXCPIT	N	Controls the maximum number of AO CPHF iterations during MP2 gradient runs. Relevant to all MP2 gradient programs. (Default value is 100)

CPTOL	N	Controls the maximum allowed RMS error of the approximate solution of the CPHF equations during MP2 gradient runs. Relevant to all MP2 gradient programs. (Default is 1.0D-10)

CHFSLV	Chooses the type of CPHF solver to use. Relevant to all MP2 gradient programs.	
	CONJG	Selects a preconditioned conjugate gradient solver.
	DIIS	Selects a DIIS-like iterative solver. (Default)

THRDII	N	Threshold to turn on the DIIS convergence accelerator if it was selected to solve CPHF. Relevant to all MP2 gradient programs. (Default is 0.05)

ICUTCP	N	Sets the effective value of ICUT to be used by the CONJG CPHF solver when solving CPHF equations in direct SCF mode. The main purpose of this keyword is to avoid numerical instabilities causing the conjugate gradient solver to diverge upon reaching near-convergence. For example, one may set ICUT to 9 or 10 while tightening ICUTCP to 10 or 11. Relevant to all MP2 gradient programs. The default is to use the value of ICUT of the $CONTRL group.

RENORM	N	A density matrix renormalization factor which may be optionally used when solving CPHF equations in direct SCF mode. The purpose of renormalization is to increase the numerical stability of the conjugate gradient solver working in direct SCF mode. A value of 0.0D0 disables renormalization. Relevant to all MP2 gradient programs. The values to try are in the range of 0.01D0 to 10.0D0. (Default is 0.0D0)

IOPARS	Specifies a set of specific I/O optimization flags for the direct access files (DAFs) used by the MP2 energy and MP2 gradient programs. The input is a 2-digit decimal value. The least significant digit controls generic DAF I/O operations. The most significant digit controls the behavior of "READ-THEN-WRITE-TO-THE-SAME-LOCATION" operations. Each digit can have one of eight values, which are listed below. The default value for this option is 74 in the case of a non-negative value of DM2MET, and 75 otherwise. For small jobs, it is recommended to set IOPARS to 64. This option does not pertain to METHOD=1 MP2 energy and gradient programs.	
	0	no particular I/O optimization;
	1	no particular I/O optimization, file cache write-through mode;
	2	optimizes file I/O for sequential access;
	3	optimizes file I/O for sequential access, file cache write-through mode;
	4	optimizes file I/O for random access;
	5	optimizes file I/O for random access, file cache write-through mode;
	6	optimizes file I/O for random access with some locality;
	7	optimizes file I/O for random access with some locality, file cache write-through mode;

IOFLGS	N1,N2,…	An array of 4 elements that can be used to define I/O optimization flags for the files DASORT and DAFL30 which are used by the METHOD=1 MP2 energy and gradient code. The format of these flags is the same as for the IOFLGS array of $SYSTEM group. In addition, the constant 0x00010000 (*i.e.* bit 16) is an alias for 0x00004000 (*i.e.* bit 14). IOFLGS(1) defines I/O flags used for file DASORT during the first stage of the MP2 calculation. IOFLGS(2) defines I/O flags used for file DASORT during the second stage of the MP2 calculation. Similarly, IOFLGS(3) and IOFLGS(4) defines I/O flags for file DAFL30.

$MP2GRD group

This group contains keywords relevant to the new serial/parallel MP2 gradient/properties code. Use of the new code is enabled by the presence of the METHOD=1 setting in the $MP2 group or by running MP2 gradient runs in parallel. Its behavior can be further controlled by the GRDMET keyword in the $MP2 group.

The keywords of the $MP2GRD group can be used for fine tuning the new code. A few additional keywords which control MP2 gradient code are part of the $MP2 group. These are CPTOL, MXCPIT, CHFSLV, THRDII, ICUTCP, and IOFLGS. The description of these options can be found in the documentation on $MP2 group.

CACHE	Instructs whether or not Firefly should reduce the amount of disk reads during the second MP2 gradient 2-electron integral half-transformation at the cost of some extra memory. The default value depends on the particular job.	
	.TRUE.	Reduce the amount of disk reads.
	.FALSE.	Do no reduce the amount of disk reads.

ASYNC	Activates/deactivates asynchronous disk I/O during the 2-electron gradient computation at the cost of some extra memory.	
	.TRUE.	Activates asynchronous disk I/O. (Default)
	.FALSE.	Deactivates asynchronous disk I/O.

XASYNC	Activates/deactivates asynchronous disk I/O during the second MP2 gradient 2-electron integral half-transformation at the cost of some extra memory.	
	.TRUE.	Activates asynchronous disk I/O. (Default)
	.FALSE.	Deactivates asynchronous disk I/O.

ASYSNC	Activates/deactivates asynchronous "sync" (*e.g.*, "disk flush") calls.	
	.TRUE.	Activates asynchronous "sync" calls.
	.FALSE.	Deactivates asynchronous "sync" calls. (Default)

FUSED	Used to change the ordering of the loops so that the disk I/O becomes less irregular.	
	.TRUE.	Change the ordering of the loops. (Default)
	.FALSE.	Do not change the ordering of the loops.

DBLBF	Enables/disables double buffering and asynchronous I/O for disk writes during the second MP2 gradient two-electron integral half-transformation, at the cost of minor memory overhead.	
	.TRUE.	Enables double buffering and asynchronous I/O for disk writes. (Default)
	.FALSE.	Disables double buffering and asynchronous I/O for disk writes.

GLBLSN	Enables/disables the global synchronization of disk I/O during the second MP2 gradient two-electron integral half-transformation when running in parallel.	
	.TRUE.	Enables the global synchronization of disk I/O.
	.FALSE.	Disables the global synchronization of disk I/O. (Default)

NBUFS	N	The number of buffers used for double buffered asynchronous I/O (see the DBLBF option). The default value is calculated such that it is close to the optimum.

TOL1	N	Threshold to store half-transformed 2-e integrals. (Default is 1D-9)

TOL2	N	Threshold to store half-transformed DM2 elements. (Default is 1D-9)

RANDOM	N	The random seed to initialize the random number generator used while distributing AO shell pairs over nodes. (Default value is 0.33333...)

The next four keywords, MXI, MXPQ, RSIZE1, and RSIZE2, allow one to fine tune the performance of the new MP2 gradient code. Their default values work fine in most situations. However, sometimes there might be a need to change them, either in order to further increase the program's efficiency or to reduce memory needs. As the new MP2 gradient algorithm is quite complex, it is not always easy to find optimal settings. Some guidelines can be found below. In addition, it is recommended to always run a check job. This should be run in parallel mode using the same number of processes as in the production run.

The MXI keyword affects both the amount of disk I/O and the memory needs (mainly the memory needs of the MP2 part of the second half-transformation are affected). Lowering MXI decreases the amount of memory required by the code but increases the amount of disk I/O. MXI cannot be larger than the number of active occupied orbitals (as it is the number of active orbitals per one slice (or batch), and at least one slice (batch) is always required). In addition, by code design, MXI cannot be larger than 256. The default settings are designed to minimize disk I/O and thus MXI is set as large as it is possible while obeying the two conditions above; hence by default MXI is min(N_{occ},256).

The MXPQ keyword cannot be set below some value defined by the properties of the particular basis set in use (the program will adjust settings if necessary) and affects the memory needs of the DM2 part of the second half-transformation stage as well as the memory needs of the 2-electron gradient integral calculation stage. Lowering MXPQ significantly increase the memory requirements of the former while (less significantly) decreasing the requirements of the latter.

The RSIZE1 keyword mainly affects the memory needs of the first half-transformation stage. Lowering RSIZE1 decreases memory needs but makes I/O less efficient due to the use of smaller record sizes for random access I/O.

The RSIZE2 keyword mainly affects the memory needs of the DM2 part of the second half-transformation stage. Lowering RSIZE2 decreases memory needs but makes I/O less efficient due to the use of smaller record sizes for random access I/O.

MXI	N	The maximum number of occupied orbitals to be processed in one slice during the first and the second integral half-transformation. Cannot be larger than 256 by design. (Default is min(N_{occ},256))

MXPQ	N	The maximum number of AO orbital pairs to be processed in one slice during the second 2-electron integral half-transformation and 2-electron gradient computation. (Default is 255)

RSIZE1	N	Record size (in 12-byte quantities) used to store half-transformed 2-electron integrals. (Default is 32768)

RSIZE2	N	Record size (in 16-byte words) used to store half-transformed DM2 elements. (Default is 32768)

$MCSCF group

This group controls various MCSCF-related options. It is recommended to carefully read the MCSCF chapter of the Firefly manual before attempting MCSCF computations.

CISTEP	Chooses the configuration basis.	
	ALDET	Chooses the determinant CI code, which requires $DET input. (Default when QUAD≠.TRUE.)
	GUGA	Chooses the graphical unitary group CSF code, which requires $DRT input. This is the only value usable with the QUAD converger. (Default when QUAD=.TRUE.)

ISTATE	N	For SA-MCSCF calculations this keyword specifies the state (root) of interest, *i.e.* the state of which the energy will be used during geometry optimizations, saddle point locations, or MECI searches. A value of 0 means no state of interest. Setting this keyword non-zero enables the use of code optimized for the calculation of state-specific gradients for SA-MCSCF. As such, this keyword should not be given for SS-MCSCF calculations or for SA-MCSCF calculations in which no state-specific data is needed. (Default is 0)

NTRACK	N	Enables state tracking for the N lowest roots. A value of 0 means no state tracking is used. Be sure to request enough states (of the right multiplicity) with the NSTATE keyword of $DET or $GUGDIA. (Default is 0)

The next four options choose an orbital optimizer. Only one of these may be set .TRUE. at a time.

FOCAS	Selects a first order convergence method (FOCAS). This optimizer may only be used with complete active spaces.	
	.TRUE.	Enable the FOCAS optimizer.
	.FALSE.	Disable the FOCAS optimizer. (Default)

SOSCF	Selects an approximately second order convergence method (SOSCF). This is the most efficient method.	
	.TRUE.	Enable the SOSCF optimizer. (Default)
	.FALSE.	Disable the SOSCF optimizer.

FULLNR	Selects a fully second order convergence method with an exact orbital Hessian (FULLNR).

	.TRUE.	Enable the FULLNR optimizer.
	.FALSE.	Disable the FULLNR optimizer. (Default)

QUAD	Selects a fully quadratic (orbital and CI coefficient) optimization method (QUAD) that is applicable to FORS and non-FORS wavefunctions. This converger cannot be used with state-averaging.	
	.TRUE.	Enable the QUAD optimizer.
	.FALSE.	Disable the QUAD optimizer. (Default)

The next options pertain to the FOCAS converger only.

CASDII	N	The threshold at which to start DIIS. (Default is 0.05)

CASHFT	N	The level shift value. (Default is 1.0)

NRMCAS	Enables/disables renormalization of the Fock matrix.	
	0	Do not renormalize the Fock matrix.
	1	Renormalize the Fock matrix. (Default)

The next option pertains to the SOSCF converger only.

NUMFO	N	The number of FOCAS iterations to perform before switching to the SOSCF converger. Values > 1 may help to converge MCSCF in difficult situations. Note that NUMFO will be reset to 0 upon completion of first MCSCF procedure. (Default is 1)

The next options pertain to the FULLNR converger. Note that these also influence QUAD as this converger always starts with one or more FULLNR iterations.

DAMP	N	The damping factor, which is adjusted by the program as necessary. (Default is 0.0)

METHOD	Selects one of two methods for the construction of the Newton-Raphson matrices.	
	DM2	Chooses a density driven construction. (Default)
	TEI	Chooses a two-electron integral driven construction. This method requires less memory than DM2, but is much slower.

MINMEM	Can be used to reduce the memory demands during FULLNR/DM2 runs by roughly 50 % at the cost of a performance penalty.	
	.TRUE.	Reduce memory demands.
	.FALSE.	Do not reduce memory Demands. (Default)

LINSER	Flag that controls the use of a method similar to direct minimization of SCF. The method is used if the energy rises between iterations. It may in some circumstances increase the chance of converging excited states.	
	.TRUE.	Use this method.
	.FALSE.	Do not use this method. (Default)

OPTACT	Flag that enables/disables minor changes in the behavior of the FULLNR converger. This may somewhat help in the case of slow convergence with non-FORS wavefunctions.	
	.TRUE.	Enable these changes to the FULLNR converger.
	.FALSE.	Disable these changes to the FULLNR converger. (Default)

FCORE	Flag used to freeze the optimization of the MCC core orbitals, which is useful in preparation for RUNTYP=TRANSITN jobs. This option is incompatible with gradients, so can only be used with RUNTYP=ENERGY.	
	.TRUE.	Freeze the optimization of the MCC core orbitals. Choosing this option will automatically force CANONC=.FALSE.
	.FALSE.	Do not freeze the optimization of the MCC core orbitals. (Default).

The next five options also pertain to the FULLNR converger, but are rarely used.

DROPC	Flag to include/exclude MCC core orbitals during the CI computation.	
	.TRUE.	Exclude MCC core orbitals by dropping them during the CI and instead form Fock operators used to build the correct terms in the orbital hessian. (Default)
	.FALSE.	Do not exclude MCC core orbitals.

NORB	N	The number of orbitals to be included in the optimization. When set different from the default, this option is incompatible with gradients and can only be used with RUNTYP=ENERGY. (Default is the number of AOs given in $DATA)

MOFRZ	N	An array of orbitals to be frozen out of the orbital optimization step. (Default is 0 *i.e.* not to freeze any orbitals).

NOROT	N1,N2,...	An array of up to 10000 pairs of orbital rotations to be omitted from the NR optimization process. The program automatically deletes all core-core rotations, all act-act rotations if FORS=.T., and all core-act and core-virt rotations if FCORE=.T. Additional rotations are input as I1,J1,I2,J2... to exclude rotations between orbital I running from 1 to NORB, and J running up to the smaller of I or NVAL in $TRANS. (Default is not to omit any orbital rotations.)

LDAR	N	The record size for temporary files. Reasonable values are in the range 4096-65536. The default is computed automatically. This option is currently not used by Firefly.

The next option pertains to the QUAD converger.

QUDTHR	N	The threshold on the orbital rotation parameter, SQCDF, to switch from the initial FULLNR iterations to the fully quadratic method. (Default = 0.05)

The remaining keywords of this group apply to all convergence methods.

FORS	Flag which should be used to specify whether or not the MCSCF function is of the Full Optimized Reaction Space type (also known as CASSCF). When set to .TRUE., act-act rotations are omitted from the calculation. However, with the FULLNR method the convergence is usually better when these are included, even for FORS-type wavefunctions.	
	.TRUE.	The wavefunction is assumed to be of the FORS type. Act-act rotations are omitted from the SCF procedure. (Default for FOCAS and SOSCF)
	.FALSE.	The wavefunction is assumed not to be of the FORS type. Act-act rotations are included in the SCF procedure. (Default for FULLNR and QUAD)

ACURCY	N	The primary convergence criterion, the maximum permissible asymmetry in the Lagrangian matrix. (Default is 1.0E-05)

ENGTOL	N	A secondary convergence criterion. The run is considered converged when the energy change is smaller than this value. (Default is 1.0E-10)

MAXIT	N	Maximum allowed number of iterations. The MAXIT keyword of the $CONTRL group may also be used (though it will be overruled by MAXIT of $MCSCF if both are specified). (Default is 100 for FOCAS, 60 for SOSCF, 30 for FULLNR and QUAD)

MICIT	N	Maximum number of microiterations within a single MCSCF iteration. It is generally not necessary to change its default value. (Default is 5 for FOCAS and SOSCF, 1 for FULLNR and QUAD)

NWORD	N	The maximum amount of memory to be used, in MWords. A value of 0 means to use all available memory. (Default is 0)

CHKPOP	Enables or disable a check for very small populations of active orbitals. Normally, small occupancies indicate some problems with active space selection, but should be ignored for extended active spaces intended for PES exploration.	
	.TRUE.	Enable this check. This allows Firefly to abort the job in the case of pathologically small occupancies of some orbitals within the active space. (Default)
	.FALSE.	Disable this check.

JKMAT	Controls the optional formation and printout of closed shell-like J and K matrices. This option is mainly intended for debug purposes.	
	.TRUE.	Enable this optional formation and printout.
	.FALSE.	Disable this optional formation and printout. (Default)

IJORBS	N1,N2	An array of two numbers denoting a pair of MOs for which two-electron Coulomb and exchange molecular integrals should be computed upon the completion of MCSCF procedure. This option should be used with JKMAT option above. (Default is 0,0 i.e. not to compute any integrals.)

CANONC	Flag used to request formation of the closed shell Fock operator, and generation of canonical core and virtual orbitals. This will order the MCC core by their orbital energies.	
	.TRUE.	Form the closed shell Fock operator and canonical core orbitals. (Default)
	.FALSE.	Do not form the closed shell Fock operator and canonical core orbitals.

The following three keywords can be used to generate additional data if the CANONC keyword was set .TRUE. Only one of these may be set .TRUE. at a time.

IFORB	Flag to additionally request generation of the canonical active orbitals and their energies. This option should be used together with the CANONC keyword.	
	.TRUE.	Generate canonical active orbitals and their energies.
	.FALSE.	Do not generate canonical active orbitals and their energies. (Default)

ILORB	Flag to additionally request generation of the Lagrangian's active orbitals and their pseudo-energies (i.e., Lagrange multipliers) by diagonalizing the MCSCF's Lagrangian. This option should be used together with the CANONC keyword.	
	.TRUE.	Generate the Lagrangian's active orbitals and their pseudo-energies.
	.FALSE.	Do not generate the Lagrangian's active orbitals and their pseudo-energies. (Default)

IGORB	Flag to request formation of the generalized closed shell Fock-like operator and generation of generalized canonical core, active, and virtual orbitals. This option should be used together with the CANONC keyword.	
	.TRUE.	Form the generalized closed shell Fock-like operator and generate conical orbitals.
	.FALSE.	Do not form the generalized closed shell Fock-like operator nor generate conical orbitals. (Default)

Additional keywords of this group that apply to all convergence methods:

GFDTOL	N	An orbital population threshold used in computing the generalized closed shell Fock-like operator. This parameter is used to distinguish between primary and secondary active orbitals. Values smaller than the default (down to 0.1) are recommended when exploring areas of PES. (Default is 0.5)

SD	Enables/disables the use of a symmetry-aware orbital canonicalization procedure which produces symmetry adapted MOs. It is safe to turn on this option.	
	.TRUE.	Use symmetry-aware orbital canonicalization.
	.FALSE.	Do not use symmetry-aware orbital canonicalization. (Default)

KEEPAS	Flag that instructs Firefly to minimize rotations within the active space. This flag does not affect computed energies but helps to track states more reliably.	
	.TRUE.	Minimize rotations within the active space. (Default)
	.FALSE.	Do not minimize rotations within the active space. (Default)

NOCI	Provides control over re-computation of CASCI using CASSCF's natural orbitals obtained upon completion of the MCSCF procedure and requests a printout of the expansion of the CI vector computed using NOs. This option should only be used for FORS-MCSCF/CASSCF wavefunctions.

	0	Do not re-compute CASCI using NOs. (Default)
	1	Re-compute CASCI using NOs.

EKT	Flag used to request generation of extended Koopmans' theorem orbitals and energies. For information on this option, see R. C. Morrison and G. Liu J. Comput. Chem., 1992, 13, 1004-1010. Note that the process generates non-orthogonal orbitals, as well as physically unrealistic energies for weakly occupied MCSCF orbitals. The method is meant to produce a good value for the first ionization potential.	
	.TRUE.	Generate extended Koopmans' theorem orbitals and energies.
	.FALSE.	Do not generate extended Koopmans' theorem orbitals and energies. (Default)

NPUNCH	The MCSCF PUNCH option, which controls the amount of data that gets written to the PUNCH file. Analogous to the NPUNCH option in $SCF.	
	0	Do not punch out the final orbitals.
	1	Punch out the occupied orbitals.
	2	Punch out occupied and virtual orbitals. (Default)

NPFLG	N	Array that controls the printing of debug data. This keyword is analogous to the same keyword in $CIINP. Elements 1, 2, 3, 4, 6, 8, and 9 are usable. The latter two control debugging the orbital optimization and extended CI coefficient expansion printout.

DUMPRF	Flag that instructs Firefly to punch the optimized reaction field that was generated by the solvent models in use, if any. This will produce a formatted $RFIELD group.	
	.TRUE.	Punch the optimized reaction field.
	.FALSE.	Do not punch the optimized reaction field. (Default)

READRF	Flag to read in a previously punched $RFIELD group from the input file and modify the one-electron Hamiltonian accordingly. Can be used to restart PCM/SCRF/EFP runs.	
	.TRUE.	Read in the $RFIELD from the input file.
	.FALSE.	Do not read in the $RFIELD from the input file. (Default)

$MCAVER group

The $MCAVER input group allows control over exactly how state-specific gradients for state-averaged MCSCF are computed. $MCAVER is also relevant for Minimum Energy Crossing Point (MECP or MECI) location runs, providing control over the penalty function and Lagrange multiplier based codes used during these runs.

The first three keywords control the calculation of SS gradients for SA MCSCF.

DELTAW	N	The step size (dimensionless) of the state ISTATE's weight used in finite differencing. Recommended values are in the range from 0.0005 to 0.0025. Finite differencing might also involve JSTATE (if specified). To clarify, the CI location code is smart enough to avoid finite differencing over JSTATE if there are only exactly two states used in SA-MCSCF. However, if there are three or more states in averaging, finite differencing will involve both ISTATE and JSTATE for the purpose of localizing CIs. (Default is 0.0015)

CONIC	Selects one of the three programmed approaches to finite differencing.	
	0	Selects the use of central (*i.e.* symmetric) second order finite differences. Calculations with CONIC=0 are performed as follows. First, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE increased by DELTAW. Second, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE decreased by DELTAW. Third, the calculation of SA-MCSCF energies is performed using unmodified original weights, and, finally, the state-specific expectation value type density matrix is computed for state ISTATE. This is the most economical way of computation. In addition, it provides the way to obtain the state-specific properties for the state of interest.
	1	Selects an alternative approach based on the use of a forward finite differencing scheme, which is more stable in the case of nearly quasi-degenerated CI roots and hence more suitable for the location of conical intersections. Calculations with CONIC=1 are performed as follows. First, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE increased by DELTAW. Second, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE increased by DELTAW/2. Finally, the calculation of SA-MCSCF energies and the effective gradient is performed using unmodified original weights, however, the state-specific density matrix is not computed (hence, no state-specific properties are available).
	2	Selects another approach which is similar to CONIC=1 but which is even more robust in the vicinity of conical intersections. First, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE. Second, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE increased by DELTAW/2. Finally, the calculation of SA-MCSCF energies and the effective gradient is performed with the original weight of state ISTATE increased by DELTAW. As with CONIC=1 no state-specific density matrix is computed and, thus, the state-specific properties are not

		available.

HPGRAD	Requests extra high precision during the computation of the two-electron contribution to the effective gradient. This may significantly slow down computations and usually does not increase the precision of the computed state-specific gradients considerably.	
	.TRUE.	Enable extra high precision during the computation of the two-electron contribution.
	.FALSE.	Disable extra high precision during the computation of the two-electron contribution. (Default)

The remaining keywords in this group provide control over the penalty functions and Lagrange functions used during the location of MECPs and MECIs. The penalty function based methods programmed in Firefly are based on the minimization of the following expression:

$$F(E_i, E_j) = \text{alpha}*E_i + (1\text{-alpha})*E_j + \text{Penalty}(E_j\text{-}E_i) = \min \quad (1)$$

whereby possible values of alpha are 0.5 and 1.0. The Penalty(ΔE) is the penalty function which depends on the energy splitting between two states.

The Lagrange function is as follows:

$$L(R) = E_i(R) + \lambda(R)*(E_j(R) - E_i(R)) \quad (2)$$

where λ is the Lagrange multiplier for constrain $E_i = E_j$ and (R) denotes the dependence of all quantities on the molecular geometry. A second form of L is a follows:

$$L'(R) = (E_i(R) + E_j(R))*0.5 + \lambda'(R)*(E_j(R) - E_i(R)) \quad (3)$$

These functions are explained in more detail in the section on the location of CIs and ISCs in the Firefly manual.

JSTATE	N	Selects the second state of interest. Normally JSTATE should be equal to ISTATE + 1, unless you are interested in searching for the simultaneous crossing of several states. A value of 0 indicates there is no second state of interest. (Default is 0)

MULTIW	A flag that should be set for runs computing semi-numerical SS gradients for SA-MCSCF for two states (i.e. states ISTATE and JSTATE) when other states are present in the averaging procedure. If not set, these runs will abort after performing some preliminary setup. If only the states ISTATE and JSTATE are present in the MCSCF averaging, this flag should not be used. Internally, this flag directs Firefly to compute SS gradients for two states independently on each other. When set to .FALSE., the SS gradient is computed only for state ISTATE after which the gradient for state JSATET is computed based on the averaged effective gradient and on the gradient of state ISTATE. The second approach is approximately 5/3 times faster but is only possible if there are no additional states in the state averaging.

	.TRUE.	Causes the code to assume that additional states to ISTATE and JSTATE are present in the MCSCF averaging procedure.
	.FALSE.	Causes the code to assume that only ISTATE and JSTATE are present in the MCSCF averaging procedure. (Default)

SHIFT	N	The value of energy difference shift during the location of MECIs/MECPs. Firefly always searches for geometries such that $E_j(R) - E_i(R) = SHIFT$. A value of 0.0 directs Firefly to search only for true MECIs/MECPs. This option only affects penalty function based code. (Default is 0.0)

TARGET	Provides control over the use of ISTATE and JSTATE in the penalty functions and Lagrange functions.	
	MIXED	Sets alpha in equation (1) to 0.5, *i.e.* use half-sum of state energies. This directs Firefly to use equation (3) for the computation of the Lagrange function. (Default)
	PURE	Directs Firefly to use the pure energy of state ISTATE in equation (1) and use equation (2) for the Lagrange multiplier based method.
	PURE2	Directs Firefly to use the pure energy of state JSTATE in equation (1) and use equation (2) for the Lagrange multiplier based method.

PENLTY	Selects one of the three programmed penalty functions. Note that these functions will only be used in MECP/MECI location runs which do not use Lagrange multiplier approach.	
	1	$Penalty(\Delta) = B*\ln(1.0 + (\Delta/A)^2)$ This is the so-called Ciminelli penalty function.
	2	$Penalty(\Delta) = B*(\Delta^2)/(\Delta+A)$ This is a penalty function suggested by Levine, Coe, and Martinez.
	3	$Penalty(\Delta) = B*((\Delta^2 + A^2)^{1/2}) - B*\|A\|$ This is a penalty function specific to Firefly.

A B	N	These keywords specify values of parameters A and B used in the penalty functions (selected with the PENLTY keyword). Their default values are: PENLTY=1 A = 0.008 and B = 0.2 PENLTY=2 A = 0.02 and B = 3.5 PENLTY=3 A = 0.01 and B = 3.5

XGRAD	Flag that selects the use of either the penalty function code or the Lagrange multiplier based code.	
	.TRUE.	Use the Lagrange multiplier based code. (Default when $STATPT METHOD=CONIC)
	.FALSE.	Use the penalty function code. (Default when $STATPT METHOD ≠ CONIC)

XHESS	Flag that requests the use of a specially optimized form of the approximate Hessian used by all quasi-Newton type geometry optimizers. This option is only supported for

	PENLTY=1 penalty function and does not have any effect otherwise.	
	.TRUE.	Use the specially optimized form of the approximate Hessian.
	.FALSE.	Use normal approximate Hessian. (Default)

SSGRAD	Flag that can be used to disable all features requesting Firefly's state-specific gradients for SA-MCSCF. The jobs depending on these features will then abort. This option is now obsolete and should not be used.	
	.TRUE.	Enable SS gradients for SA-MCSCF. (Default)
	.FALSE.	Disable SS gradients for SA-MCSCF.

$TRACK group

The $TRACK input group is used to control MCSCF state tracking. Tracking is enabled with the NTRACK keyword of the $MCSCF group.

TOL	N	The scaling factor for diagonal of the overlap matrix. (Default is 1.2)

UPDATE	Flag that controls the updating and replacement of reference vectors.	
	.TRUE.	Reference vectors will be updated and replaced by remapped current CI vectors at the end of each MCSCF iteration.
	.FALSE.	The CI vectors from the very first MCSCF iteration at the initial geometry will be used as the reference vectors throughout all calculations. (Default)

FREEZE	Flag that controls remapping during the calculation of state-specific gradients for SA-MCSCF. Normally, this flag must be set .TRUE. to get reliable results if two or more MCSCF states are quasi-degenerated.	
	.TRUE.	The final remapping scheme of the first of three MCSCF calculations used to compute SS gradient for SA-MCSCF will be applied “as is” during the second and third stage of gradient computations. (Default)
	.FALSE.	The dynamic tracking will be active throughout all three MCSCF procedures.

RESET	Flag that controls the resetting of the state tracking. See also the STICKY keyword.	
	.TRUE.	State tracking will be reset at the beginning of each MCSCF computation and the reference vectors will be re-initialized by the vectors from the first CI step.
	.FALSE.	Do not reset state tracking. (Default)

STICKY	Flag that controls the reuse of the existing reference vectors. This option is designed to be used together with the RESET keyword.	
	.TRUE.	If RESET is also set .TRUE., state tracking will be reset at the beginning of each MCSCF computation but the existing reference vectors will be reused.
	.FALSE.	Do not reuse existing reference vectors. (Default)

DELCIV	Flag that controls the storage of converged CI vectors.	
	.TRUE.	Delete the file with converged CI vectors of the previous CI stage before each new CI step so that the old vectors will not be used as the initial guess for the new CI procedure.
	.FALSE.	Reuse old CI vectors as the initial guess and thus keep the file intact. (Default)

The following two keywords should be used together and provide a way to restart an interrupted tracking-based run as if it was not interrupted at all.

READMAP	Flag that allows manually setting the initial mapping of states.	
	.TRUE.	Set the initial mapping of states as specified by the user. The order of states can be specified with the MAP keyword.
	.FALSE.	No manual mapping of states. (Default)

MAP	N	An array that specified the initial mapping of states. As an example, if set to MAP(1)=1,3 the states will be mapped $1 \rightarrow 1$ and $2 \rightarrow 3$.

$XMCQDPT and $MCQDPT groups

These groups provide control over XMCQDPT2 and MCQDPT2 calculations respectively and are relevant when SCFTYP=MCSCF with MPLEVL=2. As they share exactly the same input keywords, both methods are referred as (X)MCQDPT below, with their input groups collectively labeled as $(X)MCQDPT.

Printout control:

LPOUT	N	Printout control option. LPOUT=0 gives normal printout. LPOUT>0 removes most of the output, while LPOUT<0 gives debug printout (*e.g.* -1, -5, -10, -100). Default is LPOUT=0.

General execution modifiers:

INORB	Instructs Firefly whether to use orbitals from the preceding MCSCF step or read in orbitals from the input file.	
	0	Perform a MCSCF calculation before the (X)MCQDPT calculation and use the converged MCSCF orbitals for the (X)MCQDPT calculation. (Default)
	1	Do not perform an MCSCF calculation before the (X)MCQDPT calculation. Instead, read converged orbitals directly from the $VEC group of the input file and skip immediately to the (X)MCQDPT computation. A complete $VEC group including virtual orbitals must be given. It is recommended to use extra high precision when punching orbitals for subsequent use with this option (WIDE=.TRUE. in $CONTRL).
	2	Do not perform an MCSCF calculation before the (X)MCQDPT calculation. Instead, read converged orbitals using Firefly's standard protocol (*i.e.* as directed by the directives of the $GUESS group) and skip immediately to the (X)MCQDPT computation. A complete $VEC group including virtual orbitals must be given. The benefit of this option as compared with INORB=1 is that orbitals are orthogonalized and optionally purified, symmetrized, etc. It is recommended to use extra high precision when punching orbitals for subsequent use with this option (WIDE=.TRUE. in $CONTRL).

IFORB	Provides further control over the orbitals used for the (X)MCQDPT calculation.	
	0	Omit canonicalization of input orbitals.
	1	Determine the semi-canonical Fock orbitals. (Default)
	-1	Reuse semi-canonical Fock orbitals generated during the preceding MCSCF calculation, if semi-canonical orbitals are available. Otherwise, compute them. This option has a special extended form, namely IFORB(1)=-1,1,1 which means to reuse as much information from the preceding MCSCF calculation as possible. This option

		can save a lot of CPU time.
	2	Determine and use natural orbitals for QDPT calculations rather than semi-canonical orbitals. This option is generally not recommended.

IROT	Enables/disables the use of the simplified MP2-like formula for excitations from double occupied inactive orbitals to external orbitals. The MP2-like formula is used to compute (X)MCQDPT2' energies instead of (X)MCQDPT2 ones.	
	0	Disable the use of the MP2-like formula. (Default)
	1	Enable the use of the MP2-like formula.

ISELCT	Provides control over CSF selection.	
	0	No CSF selection is allowed. (Default)
	1 to 4	Allow selection of important CSFs to reduce memory needs and CPU time. The values of 1, 2, 3, and 4 select slightly different schemes for CSF selection based on their weights (see the description of THRWGT below for more information). The least advanced and simplest method of selection is method #1. The most advanced and recommended scheme is the method #4, although the difference between the different methods is rather subtle. Values of 2, 3, and 4 allows the use of the second element of the ISELCT() array, *i.e.* ISELCT(2). The value of ISELCT(2), if given, is interpreted as the maximum allowed number of CSFs to select. If ISELCT(2) is in effect, the actual threshold used in selection procedure may exceed THRWGT.
	-1 to -4	The values of -1, -2, -3, and -4 for ISELCT(1) are the counterparts of their positive analogs which, in addition, perform the rotation of CI roots computed in the incomplete space after CSF selection. This rotation is within the space spanned by these CI roots and is constructed to maximize the overlap of rotated states with the initial CASCI states before CSF selection. This option can be important for MCQDPT2 but is ignored for XMCQDPT2 because XMCQDPT2 depends only on the subspace spanned by the CI vectors rather than on the particular basis in this subspace (*i.e.* CI vectors).

THRWGT	N	Weight threshold for retaining CSFs in runs that use CSF selection. The weight of a CSF is the maximum over CI states included in PT of the square of its CI coefficient. (Default is 1.0D-6)

State specification and (X)MCQDPT-related details:

ISTSYM	N	The state symmetry of the target state(s). This is given as an integer. Note that only Abelian groups are supported in $DATA. The default is 1, the totally symmetric state. ISTSYM= –1 disables the use of symmetry by the (X)MCQDPT2 code and allows one to compute states having different symmetries in a single run.

		ISTSYM= 1 2 3 4 5 6 7 8 C1 A Ci Ag Au Cs A' A'' C2 A B C2v A1 A2 B1 B2 C2h Ag Bg Au Bu D2 A B1 B2 B3 D2h Ag B1g B2g B3g Au B1u B2u B3u

MULT=	N	The spin multiplicity of the target states. By default, its value is identical to that of the MULT keyword of the $CONTRL group.

NMOACT=	N	The (positive) number of active orbitals forming the CAS space. The default is to use the number of active orbitals as defined in $DET or $DRT, depending on the choice of CISTEP in $MCSCF.

NMODOC	N	The (non-negative) number of orbitals which are doubly occupied in every MCSCF configuration, *i.e.* the number of inactive orbitals, which are to be included in the perturbation calculation. The default is to use all valence orbitals between the chemical core and the active space.

NMOFZC	N	The (non-negative) number of frozen core orbitals, NOT correlated in the perturbation calculation. The default is the number of chemical core orbitals.

NMOFZV	N	The (non-negative) number of frozen virtual orbitals, NOT occupied in the perturbation calculation. (Default is zero, *i.e.* no virtual orbitals are frozen)

NSTATE	N	The number of target states, *i.e.* the dimension of the effective Hamiltonian. For example, NSTATE=5 selects the first five CASCI states for the PT treatment. (Default is 1)

KSTATE	N1,N2,...	One-dimension array of integer numbers that defines which CASCI states are used for the PT treatment. A maximum of 105 elements, including zeros, is allowed. KSTATE(*i*)=1 means that state #*i* should be used in the perturbation calculation, while KSTATE(*i*)=0 means the opposite. For example, if you want the perturbation correction to the second and the fourth roots, specify KSTATE(1)=0,1,0,1. The default is not to use the KSTATE array if it was not

		explicitly set by user, and use NSTATE instead. Notes: if KSTATE and NSTATE are given in the same input, KSTATE has priority over NSTATE. If a non-trivial KSTATE array is given, NSTATE will be redefined accordingly. Hereby, the numbering of the CASCI states will be redefined to skip any CASCI states disabled in KSTATE.

ISTATE	N1,N2	An integer array of two elements that defines the state(s) of interest. ISTATE(1) (or just ISTATE) is the number of the primary target state, *i.e.* the state which energy is considered as the final result of (X)MCQDPT2 calculations. The default is ISTATE(1)=1. If given, ISTATE(2) defines the number of the secondary target state. The latter option can be used for the purpose of state tracking and to indicate the secondary state for MECI/MECP location runs.

NS	Provides control over the effective Hamiltonian to use.	
	.TRUE.	Use eigenvalues of a non-symmetric effective Hamiltonian as the energies of the target states. This corresponds to the theory with intermediate normalization.
	.FALSE.	Use eigenvalues of a symmetric effective Hamiltonian as the energies of the target states. This corresponds to the theory with isometric (*i.e.* unitary) normalization. (Default)

WSTATE	N1,N2,...	One-dimension array of double precision numbers, maximum of 105 elements. WSTATE defines the weights of the CASCI states when computing the state-averaged density matrix used in the construction of a closed shell-like Fock matrix during the orbital canonicalization procedure. The default is to use weights equal for all states entering PT if no AVECOE array is given (*i.e.* states requested by NSTATE and/or KSTATE are used with equal weights). If an AVECOE array is specified, this array is used instead. Note that elements of WSTATE refer to the already renumbered states (see the description of KSTATE above).

AVECOE	N1,N2,...	One-dimension array of double precision numbers, maximum of 105 elements. AVECOE defines the weights of the CASCI states when computing the state-averaged density matrix used in the construction of a closed shell-like Fock matrix during computation of orbital energies. The default is to use weights equal for all states entering PT if no WSTATE array is given (*i.e.* states requested by NSTATE and/or KSTATE are used with equal weights). If a WSTATE array is specified, this array is used instead. Note that elements of AVECOE refer to the already renumbered states (see the description of KSTATE above).

EDSHFT	N	The Intruder State Avoidance (ISA) energy denominator shift, in Hartree. The ISA shift changes the energy denominators around poles where the denominator is zero. Each denominator x is replaced by x + EDSHFT/x, so that far from the poles (when x is large) the effect of such change is small. Setting this value to zero selects 'ordinary' (X)MCQDPT, and infinitely collapses to the MCSCF reference. A suggested value is 0.02, but experimentation with your system is recommended. In order to study the potential surface for any extended range of geometries, it is strongly recommended to use ISA, as it is quite likely that one or more regions of the PES will be unphysical due to intruder states. In order to maintain continuity when studying a PES, one usually uses the same EDSHFT value for all points on a PES. For a discussion of intruder state removal with (X)MCQDPT, see H.A.Witek, Y.-K.Choe, J.P.Finley, K.Hirao *J. Comput. Chem.* 23, 957-965 (2002) (Default is 0.0 *i.e.* no shift is used)

DFTOE	Provides control over the use of a DFT-based Fock matrix.		
	0	Do not use a DFT-based Fock matrix when computing semi-canonical orbitals and their energies. (Default)	
	N	Construct and use a DFT-based Fock matrix when computing semi-canonical orbitals and their energies as described below. DFTTYP must be given in \$CONTRL. DFTOE is the integer variable which is treated as a bitfield and interpreted as described below. Any combination of bits is allowed. Bits 0 to 3 trigger the use of the blocks of a DFT Fock matrix instead of the blocks of a closed-shell Hartee-Fock-like Fock matrix for definition of orbitals:	
		Bit 0.	Redefine frozen core orbitals from the diagonalization of the frozen core - frozen core block of a DFT Fock matrix.
		Bit 1.	Redefine double occupied orbitals from the diagonalization of the double occupied - double occupied block of a DFT Fock matrix.
		Bit 2.	Redefine active orbitals from the diagonalization of the active - active block of a DFT Fock matrix.
		Bit 3.	Redefine external orbitals from the diagonalization of the external - external block of a DFT Fock matrix
		Bits 4 to 7 trigger the use of the diagonal values of a DFT Fock matrix in the basis of the computed MOs instead of the diagonal values of a closed-shell Hartee-Fock-like Fock matrix in the basis of the computed MOs, for a definition of orbital energies:	
		Bit 4.	Define energies of frozen core orbitals using the diagonal values of a DFT Fock matrix.
		Bit 5.	Define energies of double occupied orbitals using the diagonal values of a DFT Fock matrix.
		Bit 6.	Define energies of active orbitals using the diagonal values of a DFT Fock matrix.
		Bit 7.	Define energies of external orbitals using the diagonal values of a DFT Fock matrix.

UNICOR	Provides further control over the use of a DFT-based Fock matrix, when DFTOE is non-zero.	
	.TRUE.	Combine frozen core and double occupied orbitals into a single block, then redefine the orbitals and their energies thus allowing intermixing of the initial subspaces of core and double occupied orbitals.
	.FALSE.	Do not modify the standard behavior of the DFTOE keyword. (Default)

IFITD	Allows the modification of the orbital energies of active orbitals.	
	0	Do not modify orbital energies of active orbitals. (Default)
	1 to 5	Modify computed orbital energies of active orbitals to provide the best possible one-particle fit to the diagonal of the CASCI Hamiltonian. Options 1 to 5 differ in minor details of how this fit is constructed exactly. The most advanced and thus recommended options are IFITD=4 and IFITD=2. Note, IFITD=4 should not be used with the DFTOE option. The use of modified orbital energies may improve the description of bi-radical states.

State tracking:

TRACK	Provides control over tracking of (X)MCQDPT states.	
	NONE	Do not track states. (Default)
	DIPOLE	Track (X)MCQDPT states based on their energies and zero-order dipole moments. This option requires ISTATE(1) and optionally ISTATE(2) input which define the states to track.

TRKPRS	N1,N2	Array of two double precision elements that affects the details of the tracking. TRKPRS(1) is the penalty (dimensionless) for the swap of states. The default value is 0.1. TRKPRS(2) defines the sensitivity of tracking to the change of the state's energies. The larger the value of TRKPRS(2), the less sensitive the tracking is. The default value is 0.04 Hartree. Smaller values of TRKPRS(1) and TRKPRS(2) will more likely trigger false alarms.

Control over CASCI procedure used by the (X)MCQDPT code:

NSTCI	N	Number of lowest states to be converged in the Davidson's CI diagonalization method. Diagonalization stops when at least a number of states equal to NSTCI has been converged. The default is either NSTCI=NSTATE or computed from the KSTATE input, if any. NSTCI must include at least all CASCI states to be included into the PT treatment.

NSOLUT	N	Number of states to be solved for in the Davidson's CI diagonalization method. Sometimes this might need to exceed the value of NSTCI in order to "capture" the correct roots using "extra" states. The default is NSOLUT=NSTCI *i.e.* no extra states.

MDI	N	Dimension of small Hamiltonian to be diagonalized to prepare initial guess CI states. (Default is 300)

MXBASE	N	Maximum number of expansion vectors in the Davidson diagonalization subspace (corresponds to MXXPAN in $GUGDIA and $DET). (Default is 300)

NSTOP	N	Maximum number of macro-iterations permitted in the Davidson diagonalization. (Default is 1000)

BLKSIZ	N	Block size to use when forming the matrix-vector product in the Davidson diagonalization. Blocking may (or may not) improve the performance of the diagonalization. Negative values of BLKSIZ mean to use internally stored machine-dependent block sizes. BLKSIZ=1 disables blocking. (Default is 1)

MAINCS	N1,N2,N3	Integer array of up to three elements with which the user can optionally specify the reference CFSs numbers belonging to the symmetries of the target states. This will be used to select CASCI states of proper symmetries. Can be useful in computations in which symmetry is disabled. The default is not use any reference CSFs.

THRCON	N	Threshold for the vector convergence in the Davidson's method CASCI. See THRGEN for more information. (Default is 1.0D-6)

Control over integral transformation procedures used by the (X)MCQDPT code:

ALTTRF	Provides control over the integral transformation for PT.	
	.TRUE.	Use a semi-direct high performance integral transformation code. This is the default and only possible option for direct runs. When ALTTRF is set to .TRUE., one can specify two additional elements of ALTTRF variable: ALTTRF(2) and ALTTRF(3). These options are only available when the semi-direct integral transformation code is enabled. If set, ALTTRF(2) allows an even more efficient implementation of integral

		transformation with no integral sorting stages. This option is only allowed for (X)MCQDPT2 runs which use Resolvent fitting code (see the $MCQFIT section for details) or runs with IROT set to 1. The default option is ALTTRF(2)=. TRUE. ALTTRF(3) is a modifier to ALTTRF(2) option which requests a different scheme for how disk I/O is performed. Typically, large jobs are even (much) faster with this option set, *i.e.* ALTTRF(3)=. TRUE. The default option is ALTTRF(3)=. TRUE.
	.FALSE.	Use a disk-based integral transformation code. This is the default option for conventional runs. Note, this default setting is not optimal for large jobs where the use of ALTTRF=. TRUE. option together with DIRTRF=. TRUE. in $TRANS is highly recommended.

THRERI	N	Threshold to keep transformed 2-electron integrals. See THRGEN for more information. (Default is 1.0D-12)

MAXERI	N	Length of the buffer used for I/O associated with 2-electron integrals. (Default is 4096)

MXTRFR	N	Maximum number of passes allowed during disk-based conventional integral transformation stage. Note, this option does not affect ALTTRF-based code. (Default is 1000)

SVDISK	Logical variable affecting the optimization of the disk space used during (X)MCQDPT2 runs.	
	.TRUE.	Optimize disk space usage. (Default)
	.FALSE.	Do not optimize disk space usage.

Numerical cutoffs used by the (X)MCQDPT code:

GENZRO	N	Threshold to discard small contributions when computing one, two, and three body coupling constants. See THRGEN for more information. (Default is 1.0D-12)

THRGEN	N	Threshold to discard small one, two, and three body coupling constants. (Default is 1D-8) Suggested values of various thresholds to be used for numerical gradients are: THRGEN=1D-20 THRERI=1D-20 GENZRO=1D-20 THRCON=1D-8

Memory control:

HALLOC		Provides control over the use of the heap.
	.TRUE.	Allow the use of the heap as a source of additional memory during (X)MCQDPT calculations.
	.FALSE.	Normal operation, *i.e.* the default option.

Seldom used keywords controlling lengths of various records used for I/O and other buffers:

LENGTH	N	Length of the buffer used for I/O associated with the calculation of coupling constants. Default is LENGTH=4096 for standard calculations and LENGTH=65536 for Resolvent fitting calculations.

MAXCSF	N	Maximum allowed length of the one-particle coupling constant buffer. The use of extra large active spaces might require larger buffers. Firefly will abort and warn if larger buffer is needed. (Default is 4096)

MAXROW	N	Maximum allowed number of rows in the Distinct Row Table (DRT). The use of extra large active spaces might require a larger value of MAXROW. Firefly will abort and warn if a larger value of MAXROW is needed. (Default is 10000)

$MCQFIT group

Activates and controls the Resolvent fitting-based MCQDPT2 and XMCQDPT2 code. Relevant to SCFTYP=MCSCF if MPLEVL=2.

The use of the Resolvent fitting-based code is triggered by the presence of the $MCQFIT group in the input file. In most cases, it is sufficient to specify only $MCQFIT $END without changing any parameters of the $MCQFIT group – the default parameters are selected so that the errors introduced in the computed energies are typically less than 10^{-8} Hartree. The use of the Resolvent fitting-based code is strongly recommended for large (X)MCQDPT2 jobs as it is much faster than the default code.

There are four adjustable parameters in the $MCQFIT group:

DELTAE=	N	Double precision variable. If nonzero, this variable is used to define the step of the interpolation grid. (Default is 0)

NPOINT=	N	Integer variable. Together with DELTAE, this variable is used to define the step size and the number of points of the interpolation grid. The actual grid size will be the least of the two, *i.e.* the number specified using the NPOINT variable and the number of points computed based on the value of DELTAE. Increasing NPOINT will result in an increase of the required CPU time while reducing errors in the computed energy. For IORDER=7 (see below), a step size of 0.01 to 0.02 is usually the optimal one. For XMCQDPT2, the step size is printed as the part of output of the MQLPR1 routine. (Default is 400)

IORDER=	N	Integer variable. Available values are 3, 5, and 7. Defines the order of the polynomial to use in interpolation. It is not recommended to change the default settings. (Default is 7)

IFMASK=	N	Integer variable. Interpreted as a bitfield. Can be used to selectively enable (if the corresponding bit is set to 1) or disable (if the corresponding bit is zeroed) the Resolvent fitting code for the calculation of one-body contributions (routine MQLMB1, controlled by the bit 0), two-body contributions (routine MQLMB2, controlled by the bit 1), three-body contributions (routine MQLMB3, controlled by the bit 2), and contributions from 2-electon integrals with two external orbitals (routines MQLMBR/MQLMBR0/MQLMBR1/MQLMBR2, controlled by the bit 3). The default value is that all bits are turned on.

The reference for the Resolvent fitting approach is:

A. A. Granovsky; URL: http://classic.chem.msu.su/gran/gamess/table_qdpt2.pdf