User's Guide
TeraChem v1.9

PetaChem, LLC
26040 Elena Road
Los Altos Hills, CA 94022
http://www.petachem.com

March 6, 2025

Contents

1 Introduction

TeraChem is general purpose quantum chemistry software designed to run on Nvidia CUDA-enabled GPU architectures under a 64-bit Linux operating system. Some of TeraChem features include:

- Restricted, unrestricted, and restricted open shell Hartree-Fock and grid-based Kohn-Sham ground state energy and gradient calculations
- Time-dependent density functional theory (TDDFT) and Configurational interaction singles (CIS) treatment of excited state energies
- Full support of s, p and d-type atom-centered Gaussian basis functions, and effective core potentials with angular momentum of up to L=4.
- Various DFT functionals, including range-corrected and Coulomb attenuated functionals (BLYP, B3LYP, PBE, PBE0, ωPBE, ωPBEh, ωB97, ωB97x, camB3LYP, etc) and DFT grids (800 - 80,000 grid points per atom)
  - Static grid (single grid used for the entire calculation) and dynamical grid (multigrid) integration.
  - Empirical dispersion correction (DFT-D3 and DFT-D2)
- Polarizable Continuum Models for solvation
- Geometry optimization (L-BFGS, Conjugate gradient, Steepest descent)
  - The optimization can be carried out either in Cartesian or internal coordinates as specified in the start file (all input geometries are provided in Cartesians). The Cartesian → internal → Cartesian coordinate transformation is performed automatically whenever required.
  - Constrained optimization with frozen atoms, constrained bond lengths, angles, and dihedrals.
- Transition state search (Nudged elastic band) in internal and Cartesian coordinates
- Ab initio molecular dynamics (NVE, NVT ensembles)
  - Reversible Born-Oppenheimer dynamics
  - Spherical boundary conditions
- Support of multiple-GPU systems
- Single/Dynamical/Double precision accuracy
- QM/MM treatment of surrounding water molecules using TIP3P force field[1]
- Natural bond orbital analysis through integration with NBO6
- Polarizabilities for HF and closed-shell DFT methods
- Runs on up to 16 GPUs in the same node in parallel, not parallelized to run on multiple nodes

See Section 8 for a complete list of TeraChem capabilities.

[1] W. L. Jorgensen, J. Chandrasekha, J. D. Madura, R. W. Impey and M. L. Klein, J. Chem. Phys. **79** 926 (1983).

1.1 Obtaining TeraChem

To purchase a copy of TeraChem visit http://store.petachem.com. Pricing information is available at http://www.petachem.com/pricing.html.

1.2 Citing TeraChem

Any published work that utilizes TeraChem shall include the following references:

- I. S. Ufimtsev and T. J. Martínez, "Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients and First Principles Molecular Dynamics," *J. Chem. Theory Comput.*, 2009, **5**, p2619.
- A. V. Titov, I. S. Ufimtsev, N. Luehr, and T. J. Martínez, "Generating Efficient Quantum Chemistry Codes for Novel Architectures," *J. Chem. Theory Comput.*, 2013, **9**, p213.

Additionally, there are certain modules of the code that should be referenced when used:

- Effective Core Potentials: C. Song, L.-P. Wang, and T. J. Martínez, "Automated Code Engine for Graphical Processing Units: Application the Effective Core Potential Integrals and Gradients," *J. Chem. Theory Comput.*, 2016, **12**, p92.
- Time-dependent Density Functional Theory (TDDFT): "Excited-State Electronic Structure with Configuration Interaction Singles and Tamm-Dancoff Time-Dependent Density Functional Theory on Graphical Processing Units," *J. Chem. Theo. Comp.*, 2011, **7**, p1814.
- Polarizable Continuum Models (PCM): "Quantum Chemistry for Solvated Molecules on Graphical Processing Units Using Polarizable Continuum Models," *J. Chem. Theory Comput.*, 2015, **11**, p3131.
- Hole-hole Tamm-Dancoff approximated (*hh*-TDA) DFT: C. Bannwarth, J. K. Yu, E. G. Hohenstein, T. J. Martínez, "Hole-hole Tamm-Dancoff-approximated density functional theory: a highly efficient electronic structure method incorporating dynamic and static correlation" J. Chem. Phys., 2020, **153**, p024110.
- Constrained QM/MM with flexible boundary layer using exchange (FlexiBLE): Z. Shen, W. J. Glover, "Flexible boundary layer using exchange for embedding theories. I. Theory and implementation" *J. Chem. Phys.*, 2021, **155**, p224112, and "FlexiBLEPlugin": https://github.com/wjg3/FlexiBLEPlugin.

TeraChem benefits from a number of third-party codes and these should be referenced appropriately when their functionality is used:

- Geometry optimization or transition state finding:
  - J. Kästner, J.M. Carr, T.W. Keal, W. Thiel, A. Wander and P. Sherwood, "DL-FIND: An Open-Source Geometry Optimizer for Atomistic Simulations," *J. Phys. Chem. A*, 2009, **113**, p11856.
  - T. P. M. Goumans, C. R. A. Catlow, W. A. Brown, J. Kästner, and P. Sherwood, "An Embedded Cluster Study of the Formation of Water on Interstellar Dust Grains," *Phys. Chem. Chem. Phys.*, 2009, **11**, p5431.
- Dispersion corrections:
  - S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, "A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu," *J. Chem. Phys.*, 2010, **132**, p154104.

  - S. Grimme, S. Ehrlich, and L. Goerigk, "Effect of the damping function in dispersion corrected density functional theory," *J. Comput. Chem.,* 2011, **32**, p1456.

- MAGMA routines for GPU-based matrix diagonalization:
  - S. Tomov, J. Dongarra, and M. Baboulin, "Towards dense linear algebra for hybrid GPU accelerated manycore systems," *Par. Comp.*, 2010, **36**, p232.

- OpenMM:
  - P. Eastman *et al*, "OpenMM 7: Rapid development of high performance algorithms for molecular dynamics." *PLOS Comp. Biol.*, 2017, **13**, e1005659.

1.3 Acknowledgements

This software was developed by Ivan Ufimtsev and Todd Martinez at the University of Illinois at Urbana-Champaign and PetaChem, LLC. The authors would like to especially thank Nathan Luehr for contribution of the GPU accelerated code for construction of numerical DFT grids, Alexey Titov for contribution of GPU accelerated code for electron repulsion integrals involving d functions, Chenchen Song for contribution of code for effective core potential integrals and derivatives, and Fang Liu for contribution of polarizable continuum solvation code. Geometry optimization and transition state search calculations use the DL-FIND library created by Johannes Kästner at Stuttgart University. Hartree-Fock and DFT dispersion correction code is used with permission from Stefan Grimme at the University of Münster. Matrix diagonalization on the GPU is performed using the MAGMA libraries available at http://icl.cs.utk.edu/magma. TeraChem uses standard Gaussian basis sets available at EMSL website http://www.emsl.pnl.gov/forms/basisform.html.

2 Getting started

2.1 System requirements

This version of TeraChem was compiled and tested under 64-bit RedHat Enterprise Linux 6.6 operating system running on Intel Core2 quad-core and Intel Xeon 5520 dual quad-core CPU machines. An Nvidia compute capability 2.0 (Tesla C2050 or similar) or higher (i.e. Tesla K20 or similar) graphics card is required to run the program. Please refer to the CUDA Programming Guide at http://www.nvidia.com/object/cuda_develop.html for the most current list of Nvidia GPU's that meet this requirement. A CUDA driver (352.93 or later) must be installed on the system. If desired, v7.5 of the CUDA Toolkit may also be installed (but this is not necessary). Details on how to obtain and install the CUDA driver are provided below.

Because the binary file is linked against the Intel MKL library, it is recommended to run TeraChem on Intel-based workstations.

The amount of CPU RAM needed depends on the size of the molecules that will be studied. If the molecules of interest are relatively small (less than 500 atoms), the usual 8Gb or 16Gb configuration is acceptable. For very large molecules (in excess of 10,000 basis functions), CPU RAM will often be a limiting factor. For example, molecules with 25,000 basis functions require almost 70GB of CPU memory.

2.2 Installation

First, obtain and install the latest CUDA driver (TeraChem 1.9 requires 352.93 or later). These can be obtained free of charge at

http://www.nvidia.com/drivers

Installation of the CUDA driver must be performed by a user with "root" permission, i.e. the superuser. Check with your local system administrator first, or type

```
cat /proc/driver/nvidia/version
```

since adequate CUDA drivers may already be installed. Make sure you select the 64-bit Linux operating system. After downloading the driver package, shut down the X server by typing

```
sudo init 3
```

Then launch the driver binary, and follow the instructions.

Previous versions of TeraChem (1.5K and earlier) required user installation of the CUDA toolkit. This is no longer required, as all the needed libraries are distributed with TeraChem. You **do** need to set the `LD_LIBRARY_PATH` environment variable correctly:

```
export LD_LIBRARY_PATH=$TeraChem/lib:$LD_LIBRARY_PATH
```

where you should replace `$TeraChem` with the path to your TeraChem installation.

To install the TeraChem software, first unpack the tc1.9.tar archive using the following command (in a temporary directory which you may later remove):

```
tar xvf tc1.9.tar
```

This will create a subdirectory **TCInstaller**. Run the install script by typing

```
cd TCInstaller
chmod u+x install
./install
```

This script will verify that your machine has a suitable graphics card, verify that you accept the license terms, and install the software in a location of your choosing (the "TeraChem installation directory" or `instdir` in the following). The script will create **SetTCVars.sh** which sets the appropriate environment variables. You can add **source $TeraChem/SetTCVars.sh** to your shell configuration file (which is **$HOME/.bashrc** if you use the bash shell) to automatically configure the TeraChem environment at every new login session. The `install` script will also put a temporary license file in place so you can begin using TeraChem immediately. However, this temporary license file is time-limited, so you will want to obtain a permanent license file. The `install` script ends with a form suitable for emailing to help@petachem.com:

```
-------------BEGIN HERE------------------

 Institution: ________________________

 Ordered By: _________________________

    MAC: 003048DB1D7E

    IP: 171.64.125.189

---------------END HERE------------------
```

You can regenerate this at any time by typing

```
$TeraChem/bin/machid
```

Fill in the "Institution" and "Ordered By" fields and email to help@petachem.com. When we receive this from you, we will send you a permanent license file (which should be saved as license.dat in the TeraChem installation directory) within a few days. The `TeraChem` environment variable indicates where the licensing file can be found, i.e. `$TeraChem/license.dat`.

2.3 Environment Variables

There are several environment variables that influence TeraChem:

1. `TeraChem` : This should point to the directory which contains the license files and also the `basis` subdirectory with basis set and effective core potential data. Throughout this document, this directory is called "the TeraChem directory."

2. `TeraChemLicense` : If defined, this should point to the directory which contains the license file (and will take precedence for this purpose over `TeraChem`). Specification of this environment variable is optional (and allows a single TeraChem installation to be used by multiple groups with their own license files).

3. `LD_LIBRARY_PATH` : This is a system variable, which points to the directories containing dynamic libraries. The `$TeraChem/lib` directory should be included at the beginning of this list.

4. `CUDA_VISIBLE_DEVICES` : This is a system variable which is used by the CUDA driver. If this variable is set, only GPUs listed will be visible to programs using CUDA. Setting this is the recommended way to control the usage of GPUs by TeraChem. Although it is also possible to control GPU usage in the input file or through the command line, setting this environment variable is the most effective way accomplish this goal. Note that the GPUs are reordered when this environment variable is used. For example,

```
export CUDA_VISIBLE_DEVICES=0,2
```

will use GPUs 0 and 2, but these will appear as GPUs 0 and 1 to subsequent CUDA programs such as TeraChem.

5. `MagmaMinN` : Matrices of size NxN and larger will be diagonalized using MAGMA routines (where N is the value of the environment variable).

6. `MagmaNGPUs` : When MAGMA is used, it will exploit N GPUs (where N is the value of the environment variable). These GPUs will be the first N which TeraChem sees. Please note that these are NOT necessarily the same as the GPUs which TeraChem is using for the electronic structure. As discussed below, you can control the GPUs which TeraChem uses either in the input file or through the command line. However, MAGMA will not respect that list. So, for example, if you run TeraChem as

   ```
   terachem --gpus=23 start.in
   ```

   then TeraChem will use the third and fourth GPU as the first and second GPU in the calculation. However, MAGMA will use the *first* GPU in the system (GPU #0). This can be very confusing and the easiest way to avoid problems is to restrict the visible GPUs to the ones desired by using the `CUDA_VISIBLE_DEVICES` environment variable. To accomplish the effect that the user probably desired above (that the TeraChem job uses only the third and fourth GPU), one would then type:

   ```
   export CUDA_VISIBLE_DEVICES=2,3
   terachem --gpus=01 start.in
   ```

   This will restrict the visible GPUs to the third and fourth, which are then addressed as 0 and 1 when TeraChem runs. If you were then to set `MagmaNGPUs` to 3, the program would terminate with an error (because only 2 GPUs are visible).

2.4 Licensing

There are two licensing schemes for TeraChem - node-locked and IP-checkout. Older versions of TeraChem (v1.5K and earlier) used the node-locked scheme exclusively. TeraChem will run in demo mode if an appropriate license is not found. In demo mode, the program will run for a maximum of 15 minutes and will use at most two GPU cards in parallel. If license files for both schemes are present in the TeraChem directory, TeraChem will use the node-locked licensing scheme corresponding to the `license.dat` file.

2.4.1 Node-locked Licenses

The node-locked license scheme is based on a file `license.dat` which should be in the directory given in the `TeraChem` environment variable. This file is obtained by sending the MAC addresses of the machines which are to be licensed to `help@petachem.com`. PetaChem will create an appropriate license file and send this back to you. Save the file as `license.dat` in the TeraChem directory and all restrictions should be removed. License files are typically created with a time limit of two years. This enables us to replace the license file if needed (for example if a machine is upgraded). If the license file expires, please send the old license file to `help@petachem.com` along with purchaser information and we will provide you with an updated license file. For reference, here is an example `license.dat` file (this file is not valid – it is for demonstration purposes only):

license.dat

```
# License created: 21-09-2014
# MAC: 60A44C5FC377
# Expires on DD-MM-YYYY: 10-09-2016
```



```
# Licensed application:
#    Name: "TeraChem"
#    Description: ""TeraChem v1.0""
# Features included:
# Custom data:
#         url=http://www.petachem.com
#         support=help@petachem.com
# === Start copy after the BEGIN LICENSE line and ===
# ===      finish before END LICENSE line         ===

----- BEGIN LICENSE -----
AAABAAAAAIkU0V0dJNIM3oS8SyO+bq6h+U/8jQZN
1Jd2/XC8MsVwWmYkZA6QqBPsFzSrha5uhgIo07zx
HFE1XdGgijDzhikkh2Y9SWqQ7MYMzpFPnk6ZRYUk
YHAwfbHtuo3T26qc/Ys8lHxg5mYsyTLC1YF7OggI
q0q1wSPFXdojtw8/mQ4TO4o3nFNOf9D6pqp3eZSf
jh9fF/nf5wfQx/k94Ho28wWp1TSmz84LduyyDDKB
+KYUFjeLFqma0nV3XSMVPsh4YL+ZevXXodW5meQm
hKy/BHOCGuDn5MBwJ3fnlna7wunUZ8kmItO2wOdL
2DdN3jT9F74pMquFO1ysN33eG6kBCIElAgABSAAA
AARub2lkQQAAAAhUZXJhQ2hlbUYAAAADAAAACCQk
TkdMMDMxAAAAQiQkTkdMMjAxYiR1cmw9aHR0cDov
L3d3dy5wZXRhY2hlbS5jb20kMTkkc3VwcG9ydD1o
ZWxwQHBldGFjaGVtLmNvbQAAAAckJE5HTDMxUgAA
B+AACQAKRQAAAAA=

----- END LICENSE -----
```

2.4.2 IP-Checkout Licenses

Starting with TeraChem v1.9, it is also possible to use a checkout type of license. This requires that your compute nodes have access to the open internet. If compute nodes do not have internet access, it is possible to forward the license traffic through a local HTTP proxy (for instance, running on a head node in a cluster) that has internet access and is accessible to the compute nodes. See section 2.4.3 for more details on HTTP forwarding. The basic idea of checkout type licenses is that the compute node will contact a PetaChem license server (directly or through an HTTP proxy), which will then check that a valid license exists. In this way, the identity of the node running a job is not restricted and the restriction is instead on the number of nodes that can simultaneously run jobs. To use this licensing mode, you must first contact `help@petachem.com` and they will determine the number of simultaneous nodes that you qualify for. They will also provide you with a license file which contains your unique token. This should be protected, since anyone with knowledge of this token could run TeraChem instances under your license (which would use up your available slots). This license file should be saved as `license.key` in the `TeraChem` directory. For reference, here is an example `license.key` file (the license key is not valid - this is for demonstration purposes only):

`license.key`

```
key Ki56jjjKKKK
server 54.208.252.40:8877
```

2.4.3 HTTP Forwarding

There are many proxy servers currently available free of charge or by payment, and here we use one, called `nginx`, as an example. It is a lightweight web server with low memory usage that can sustain heavy load through parallelism and load balancing. `nginx` can be downloaded from `http://nginx.org/en/download.html` and installed by running

```
./configure --prefix=install_directory --without-http_rewrite_module
make build
make install
```

To start nginx, simply

```
cd install_directory
sbin/nginx
```

Below is a simple configuration file that should be placed in `install_directory/conf` directory:

Simple nginx.conf configuration file

```
# number of CPU processes
# can be increased, CPU affinity can be specified
worker_processes  1;

# each worker can process up to 1024 connections
events {
    worker_connections  1024;
}

http {
    server {
    # proxy is running on host localhost
    # proxy is listening on port 8080
    # provided the proxy is running on localhost:8080,
    # TeraChem license.key file should contain 'server localhost:8080' line
        server_name  localhost;
        listen       8080;

        # all traffic is forwarded to PetaChem license server 54.208.252.40:8877
        # when proxy is used, 'server 54.208.252.40:8877' setting should be
        # commented out in license.key
        location / {
            proxy_pass   http://54.208.252.40:8877;
        }
    }
}
```

2.5 Updating TeraChem

Users with IP-Checkout licenses can update TeraChem from the command line by running

```
terachem --update (or terachem -u)
```

In this case the newest release will be downloaded, verified, and saved in the working directory:

```
  TRYING THE NETWORK LICENSE...
  Connecting to license server '54.208.252.40' port '8877'...
  Connected!
  Checking your license...
  Download started...  done
  Verifying...         OK
  The installer has been saved as tc1.93P.tar
```

In order to prevent potential data corruption issues, it is recommended to install the release in another directory and then follow the standard installation procedure described in Sec. 2.2. If a newer version is available, TeraChem notifies the user on startup:

```
TRYING THE NETWORK LICENSE...
Connecting to license server '54.208.252.40' port '8877'...
Connected!
Checking your license...

****************************************************************
  Greetings, Todd Martinez! You have 2 licenses in total
  IN USE: 0
  AVAILABLE: 2

  ATTENTION: A NEWER TERACHEM VERSION (v.1.93) IS AVAILABLE
  TO UPGRADE, PLEASE RUN 'terachem -u' OR CONTACT PETACHEM
****************************************************************
```

Neither version checking nor command line updating is available for users with node-locked licenses, because then TeraChem is not allowed to access the Internet. Users with node-locked licenses should contact PetaChem to obtain the latest release.

2.6 Command Line Options

Most often, the TeraChem program is run with a single command line option – the name of the input file containing directives. For example,

```
terachem start.sp
```

However, there are some other command line options that may be useful. These may be listed by running

```
terachem -h
```

which will show the following options:

```
  -h|--help              Show usage information
  -v|--version           Print version information
  -u|--update            Download the latest TeraChem release
  -f|--functionals       Print available DFT functionals
  -gxyz|--gpus=xyz       Use GPUs xyz (0-f). This command may
                         be repeated to use multiple GPUs.
  -Ux|--UseMPI=x         Use MPI named ports for input.
```



```
                        x=1  Amber interface.
                        x=2  FMS interface.
                        x=3  Single point engine mode.
 -Mx|--MPIPort=xxx      Use xxx as MPI named port
                        (default=terachem_port)
-ifile|--inputfile=file Input file for TeraChem
```

1. The MPI named ports are *not* for MPI-based parallelization. TeraChem currently only runs on a single node (using up to 16 GPUs). Instead, the commands `--UseMPI` and `--MPIPort` are for an advanced feature of TeraChem which allows TeraChem to run as a server, waiting for MPI socket communication to carry out calculations. This feature is being tested for coupling between TeraChem and Amber/FMS. The Amber/FMS interfaces are not currently documented or available for general use.

   The point engine mode is intended for use with a separately distributed Python module. It is possible to use the same interface for a general program which is described in section 7, but the protocol is subject to change. TeraChem will also print out specifications of expected communications when running in this mode. Only one program can be connected with each server.

2. Note that the following are equivalent:

   ```
   terachem start.sp
   ```

   and

   ```
   terachem --inputfile=start.sp
   ```

3. To use the second and fourth GPUs:

   ```
   terachem --gpus=13
   ```

   Note that the numbering starts from 0. Also, please note that the GPUs are numbered as TeraChem finds them. GPUs which are not visible to TeraChem will not be counted in the list. Note that the use of `--gpus` or `-g` will override any `gpus` line in the input file.

4. It is possible to hide GPUs from TeraChem and all other GPU programs using the `CUDA_VISIBLE_DEVICES` environment variable. For example, to ensure that only the second and third GPUs are visible, use:

   ```
   export CUDA_VISIBLE_DEVICES="1,2"
   ```

   Note that the numbering starts from zero.

5. To get a list of all the GPUs, use the `nvidia-smi` program supplied with CUDA by NVIDIA:

   ```
   nvidia-smi
   ```

2.7 Running sample jobs

TeraChem package contains several sample jobs located at

```
instdir/TeraChem/tests
```

After installation, cd to the caffeine directory and run TeraChem by typing

```
source instdir/TeraChem/SetTCVars.sh
instdir/TeraChem/bin/terachem start.sp
```

where `instdir` is the installation directory you chose during the install (defaults to your home directory) and `start.sp` is the name of a TeraChem input file. Note that the environment variable `TeraChem` is set by "source'ing" `SetTCVars.sh`. This is needed in order for TeraChem to locate the license and basis set library files.

The `start.sp` contains the required parameters of the job (including the filename of the file which contains the atomic coordinates for the molecule of interest). Most of the parameters have default values. The complete list of parameters available in this version is presented in Section 8. An example of the configuration file used for single point energy calculations of caffeine with the BLYP functional, DFT-D dispersion corrections, and the 6-31G* basis set is:

start.631.sp

```
# Job: Single point energy of caffeine
#
# basis set
basis           6-31g*
# coordinates file
coordinates     caffeine.xyz
# molecule charge
charge                  0
# SCF method (rhf/blyp/b3lyp/etc...): DFT-BLYP
method                  blyp
# add dispersion correction (DFT-D)
dftd            yes
# type of the job (energy/gradient/md/minimize/ts): energy
run             energy
end
```

All lines beginning with the '#' character are considered comments and are ignored by TeraChem. There is no requirement on the line ordering in the start file except that the last line should be '`end`'.

Below is the output from this example job. The program first lists the parameters followed by all GPUs used in the job. Each GPU has its amount of memory and compute capability printed next to it. The program then attempts to predict the maximum size of the molecule that can be handled on the current machine based on the amount of available CPU and GPU memory. *Note that the recommended maximum size of the system is approximate.* The SCF procedure, which includes the DIIS error (the maximum component of the DIIS error vector), integrated number of electrons, exchange-correlation energy, SCF energy, and the total time elapsed per iteration, completes the program's output.

```
$ $TeraChem/bin/terachem start.631.sp
Startfile from command line: start.631.sp


       ************************************************************
       *                    TeraChem v1.9                         *
       *                 Hg Version: 79db7953baf0+                *
       *                   Production Version                     *
       *                Compiled without textures                 *
       *           Chemistry at the Speed of Graphics!            *
       ************************************************************
       * This program may only be used in connection with        *
       * a valid license from PetaChem, LLC. Use of this program *
       * or results thereof indicates acceptance of all terms    *
       * and conditions stated in the license and that a valid   *
       * license agreement between the user and PetaChem, LLC    *
       * exists. PetaChem, LLC does not warrant the correctness  *
       * of results or their suitability for any purpose.        *
       * Please email bugs, suggestions, and comments to         *
       *                  help@petachem.com                      *
       *                                                         *
       ***********************************************************


       ************************************************************
       *  Compiled by toddmtz      Sun May 29 17:45:28 PDT 2016  *
       *  Supported architectures: sm_20 sm_30 sm_50             *
       *  Cuda compilation tools, release 7.5, V7.5.17           *
       ************************************************************


 Job started   Sun May 29 17:46:01 2016
 On localhost.localdomain (available memory: 21507 MB)

######################################### RUNTIME INFO ##########################################
/home/toddmtz/production/build/bin/terachem start.631.sp

NVRM version: NVIDIA UNIX x86_64 Kernel Module  367.18  Mon May 16 18:13:16 PDT 2016
GCC version:  gcc version 4.8.5 20150623 (Red Hat 4.8.5-4) (GCC)

        linux-vdso.so.1 =>  (0x00007ffc32bf6000)
        libcurl.so.4 => /lib64/libcurl.so.4 (0x00007f5dc0371000)
        libintbox.so.1 => /home/toddmtz/production/build/lib/libintbox.so.1 (0x00007f5da72a4000)
        libthcbox.so.1 => /home/toddmtz/production/build/lib/libthcbox.so.1 (0x00007f5da5f16000)
        libcublas.so.7.5 => /opt/CUDA/cuda-7.5/lib64/libcublas.so.7.5 (0x00007f5da4637000)
        libcufft.so.7.5 => /opt/CUDA/cuda-7.5/lib64/libcufft.so.7.5 (0x00007f5d9d9fb000)
        libcuda.so.1 => /lib64/libcuda.so.1 (0x00007f5d9c76b000)
        libcudart.so.7.5 => /opt/CUDA/cuda-7.5/lib64/libcudart.so.7.5 (0x00007f5d9c50d000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f5d9c20a000)
        libcilkrts.so.5 => /opt/intel/lib/intel64/libcilkrts.so.5 (0x00007f5d9bfcb000)
        libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f5d9bcc3000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f5d9baac000)
```



```
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f5d9b890000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f5d9b68c000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f5d9b2c9000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f5dc05f0000)
        libidn.so.11 => /lib64/libidn.so.11 (0x00007f5d9b096000)
        libssh2.so.1 => /lib64/libssh2.so.1 (0x00007f5d9ae6c000)
        libssl3.so => /lib64/libssl3.so (0x00007f5d9ac28000)
        libsmime3.so => /lib64/libsmime3.so (0x00007f5d9aa01000)
        libnss3.so => /lib64/libnss3.so (0x00007f5d9a6db000)
        libnssutil3.so => /lib64/libnssutil3.so (0x00007f5d9a4ae000)
        libplds4.so => /lib64/libplds4.so (0x00007f5d9a2aa000)
        libplc4.so => /lib64/libplc4.so (0x00007f5d9a0a5000)
        libnspr4.so => /lib64/libnspr4.so (0x00007f5d99e66000)
        libgssapi_krb5.so.2 => /lib64/libgssapi_krb5.so.2 (0x00007f5d99c1a000)
        libkrb5.so.3 => /lib64/libkrb5.so.3 (0x00007f5d99935000)
        libk5crypto.so.3 => /lib64/libk5crypto.so.3 (0x00007f5d99702000)
        libcom_err.so.2 => /lib64/libcom_err.so.2 (0x00007f5d994fe000)
        liblber-2.4.so.2 => /lib64/liblber-2.4.so.2 (0x00007f5d992ef000)
        libldap-2.4.so.2 => /lib64/libldap-2.4.so.2 (0x00007f5d9909b000)
        libz.so.1 => /lib64/libz.so.1 (0x00007f5d98e85000)
        librt.so.1 => /lib64/librt.so.1 (0x00007f5d98c7c000)
        libssl.so.10 => /lib64/libssl.so.10 (0x00007f5d98a0f000)
        libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007f5d98626000)
        libkrb5support.so.0 => /lib64/libkrb5support.so.0 (0x00007f5d98417000)
        libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007f5d98212000)
        libresolv.so.2 => /lib64/libresolv.so.2 (0x00007f5d97ff8000)
        libsasl2.so.3 => /lib64/libsasl2.so.3 (0x00007f5d97dda000)
        libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f5d97bb5000)
        libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007f5d9797d000)
        libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f5d9771c000)
        liblzma.so.5 => /lib64/liblzma.so.5 (0x00007f5d974f7000)
        libfreebl3.so => /lib64/libfreebl3.so (0x00007f5d972f3000)
###############################################################################################

Cannot find license.dat file in the TeraChem installation directory /home/toddmtz/production/build


TRYING THE NETWORK LICENSE...
Connecting to license server '54.208.252.40' port '8877'...
Connected!
Checking your license...

************************************************************
  Greetings, Martinez Group! You have 1000 licenses in total
  IN USE: 0
  AVAILABLE: 1000
************************************************************

Scratch directory: ./scr
Random number seed: 1464569163

XYZ coordinates caffeine.xyz
Molden File Output: ./scr/caffeine.molden
```

```
Using basis set: 6-31g
Spin multiplicity: 1
DIIS will use up to 10 vectors.
WF convergence threshold: 3.00e-05
Using DIIS algorithm to converge WF
Maximum number of SCF iterations: 100
Incremental fock with rebuild ever 8 iterations
Will switch to conventional Fock if diffuse functions are detected
X-matrix tolerance: 1.00e-04
PRECISION: DYNAMIC
DFT Functional requested: hf
Method: Hartree-Fock
Wavefunction: RESTRICTED
Initial guess generated by maximum overlap

****************************************
***** SINGLE POINT ENERGY CALCULATIONS *****
****************************************

using 1 out of 1 CUDA devices
   Device 0:  GeForce GTX TITAN X, 12206MB, CC 5.2 -- CPU THREAD 0
-----------------------------------------------------------------
Compiled with MAGMA support.  MAGMA parameters:
    Matrices larger than 10 square will be treated with MAGMA
       (Change by setting the MagmaMinN environment variable)
    Magma will use 1 out of 1 GPUs
       (Change by setting the MagmaNGPUs environment variable)
-----------------------------------------------------------------
 CPU Memory Available: 3995.12 MegaWords
 GPU Memory Available: 1269.83 MegaWords
 Maximum recommended basis set size: 13300 basis functions
 (limited by GPU memory)
-----------------------------------------------------------------
Not using d-functions. Configuring GPUs accordingly.
0: CUBLAS initialized, available GPU memory: 10792MB


Basis set:       6-31g
Total atoms:     24
Total charge:    0
Total electrons: 102 (51-alpha, 51-beta)
Number electrons modeled by ECPs: 0
Total orbitals:  146
Total AO shells: 90 (62 S-shells; 28 P-shells; 0 D-shells; 0 F-shells; 0 G-shells)
Spin multiplicity: 1
Nuclear repulsion energy (QM atoms): 933.356431723821 a.u.


                     *** Start SCF Iterations ***

 Iter     DIIS Error       Energy change         Energy        Time(s)
------------------------------------------------------------------------
         >>> Purifying P... IDMP = 2.51e-14 <<<
```

```
THRESPDP set to 1.00e+00
   1     0.4043648490    -672.1012472141    -672.1012472141     0.10
   2     0.2909010200      -3.0669389439    -675.1681861580     0.06
   3     0.1283248644      -0.7029524073    -675.8711385653     0.06
   4     0.0292821138      -0.1303876634    -676.0015262286     0.06
   5     0.0093170672      -0.0119835193    -676.0135097480     0.05
   6     0.0064218627      -0.0016419531    -676.0151517011     0.05
   7     0.0018119189      -0.0004989767    -676.0156506777     0.05
   8     0.0011542627      -0.0000636296    -676.0157143074     0.05
THRESPDP set to 3.59e-03
   9     0.0002755390      +0.0000288961    -676.0156854113     0.07
  10     0.0001410432      -0.0000017814    -676.0156871927     0.04
  11     0.0000428022      -0.0000003023    -676.0156874950     0.04
  12     0.0000193138      +0.0000000081    -676.0156874869     0.03
-----------------------------------------------------------------------
FINAL ENERGY: -676.0156874869 a.u.
CENTER OF MASS: {0.008017, 0.006143, 0.000066} ANGS
DIPOLE MOMENT: {4.454334, -0.892845, 0.000021} (|D| = 4.542936) DEBYE
Writing out molden info

Running Mulliken population analysis...

Total processing time: 0.68 sec

 Job finished: Sun May 29 17:46:05 2016
```

3 TeraChem I/O file formats

3.1 Input files

In addition to the start file containing job parameters, TeraChem requires at least two other files to start a job: coordinates and basis set.

3.2 Atomic coordinates

The coordinates file fed to the `coordinates` parameter should be either in XMol or PDB format. In an XMol coordinates file, the first line specifies the number of atoms, and the second line provides a description of the system (it can be left blank). Atomic coordinates are listed starting from the third line. All coordinates are either in Angstroms (default) or Bohrs (this can be specified in the start file). Here is an example coordinates file for a hydrogen molecule:

h2.xyz

```
2
Hydrogen Molecule -- Xmol format
H 0.0 0.0 0.0
H 0.7 0.0 0.0
```

Some jobs (for example, transition state search using NEB method) require several sets of coordinates (frames). In this case all frames should be listed in the coordinates file one by one, i.e.

h2-2frames.xyz

```
2
Hydrogen Molecule -- Xmol format frame 1
H 0.0 0.0 0.0
H 0.7 0.0 0.0
2
Hydrogen Molecule -- Xmol format frame 2
H 0.0 0.0 0.0
H 0.8 0.0 0.0
```

Note: There should be no blank lines between individual frames.

The PDB format often used for protein molecules is also supported and will be automatically assumed if the filename for the coordinates ends in '`.pdb`'. More details on PDB file format are available at http://www.wwpdb.org/docs.html.

3.3 Basis set file format

TeraChem supports standard atom-centered Gaussian type basis sets such as those available at EMSL website http://www.emsl.pnl.gov/forms/basisform.html. The basis set information used by TeraChem is provided by a set of files in the `basis` directory. This directory should be located at `$TeraChem/basis`, where the environment variable `TeraChem` is set to the TeraChem installation directory by default. All basis sets are listed in individual files and corresponding information for constructing initial guesses is in a separate file, e.g. `6-31g` and `6-31g.ao`.

For every atom in a basis set, the first line specifies the atom type and the following lines specify the basis functions and contractions. Consider a hydrogen atom in the 6-31G basis set (see the file `basis/6-31g`):

basis/6-31g

```
ATOM H
S 3
     18.7311370              0.03349460
      2.8253937              0.23472695
      0.6401217              0.81375733
S 1
      0.1612778              1.0000000

```

Line 1 specifies that the element symbol is H. Line 2 specifies the angular momentum of the first contraction (which can be `S`, `P`, or `D`) and the number of primitives in this contraction, N_{prim}. The next N_{prim} lines (Lines 3-5) specify the exponent and contraction coefficient for one of the primitives. This is repeated until all contracted basis functions and their primitives have been listed. Note that TeraChem normalizes the contracted functions. A blank line signifies the end of the basis set for this atom. If the atom also has an effective core potential, this is specified immediately after the basis set information. For an example here, we turn to the Na atom in the LANL2DZ basis set (see the file `basis/lanl2dz_ecp`):

basis/lanl2dz_ecp

```
ATOM Na
S 2
      0.4972000             -0.2753574
      0.0560000              1.0989969
S 1
      0.0221000              1.0000000
P 2
      0.6697000             -0.0683845
      0.0636000              1.0140550
P 1
      0.0204000              1.0000000

ECP  NCORE= 10 MAXL= 2
D-UL  5
    1      175.5502590    -10.0000000
    2       35.0516791    -47.4902024
    2        7.9060270    -17.2283007
    2        2.3365719     -6.0637782
    2        0.7799867     -0.7299393
S-UL  5
    0      243.3605846      3.0000000
    1       41.5764759     36.2847626
    2       13.2649167     72.9304880
    2        3.6797165     23.8401151
    2        0.9764209      6.0123861
P-UL  6
    0     1257.2650682      5.0000000
    1      189.6248810    117.4495683
    2       54.5247759    423.3986704
    2       13.7449955    109.3247297
```

```
31      2         3.6813579      31.3701656
32      2         0.9461106       7.1241813
```

The first line specifies this entry is for the Na atom. Lines 2-11 specify the contracted basis functions, as in the previous example. A blank line (Line 12) signals the end of the contracted basis functions. Line 13 specifies that this atom has an effective core potential which removes 10 electrons (`NCORE=10`) and includes projection terms with angular momentum up to d (`MAXL=2`). Note that `MAXL` is normally one greater than the angular momentum of the electrons occupied in the core. The next line (Line 14) begins specification of the Gaussian fits to the local potentials. The first set should correspond to the "local" potential, i.e. $U(L_{Max}+1)$. The number N_{Gauss} (5 on this line) specifies the number of Gaussians in the fit for this potential. The next N_{Gauss} lines contain the information for each of these Gaussians. The first column is the power of r for this term (should be in the range 0-2). Note that, following usual practice, this *includes* the powers of r coming from the radial volume element. The second column is the exponent for the Gaussian in this term and the third column is the coefficient of this Gaussian. The next blocks should correspond to *difference* potentials, i.e. $U(L) - U(L_{Max})$, starting from the lowest angular momentum.

The initial guess information is provided in the `basis.ao` files. These are normally machine generated, but we explain the format here. An example for hydrogen atom in the 6-31G** basis set is:

6-31gss.ao

```
1  H 0 1
2  5 5
3  4.274303e-01
4  6.654494e-01
5  0.000000e+00
6  2.465190e-32
7  -1.387779e-17
8  1.257798e+00
9  -1.149752e+00
10 0.000000e+00
11 -7.888609e-31
12 6.661338e-16
13 8.307465e-16
14 -7.808226e-16
15 0.000000e+00
16 5.257311e-01
17 -8.506508e-01
18 -4.183196e-16
19 3.874649e-16
20 0.000000e+00
21 8.506508e-01
22 5.257311e-01
23 0.000000e+00
24 0.000000e+00
25 1.000000e+00
26 0.000000e+00
27 0.000000e+00
```

The first line specifies the element symbol, the number of core orbitals (0 in this example) and the number of valence orbitals (1 in this example). The second line specifies the number of MOs and AOs which will be

constructed from this guess. Normally, both of these numbers will be identical (and equal to the number of contracted basis functions on this atom in this basis set). Subsequent lines (Lines 3-27 above) give the AO coefficients of the guess MOs. These are ordered so the most negative orbital energy is first, i.e. the lowest energy guess MO corresponds to lines 3-7 above. The second lowest energy guess MO corresponds to lines 8-12 above. The order of the AOs follows that specified in the basis set file. Finally, a blank line (Line 28) signifies the end of the guess information for this element.

The information in the `basis.ao` file is used to construct an initial guess as described in

- J.-M. Langlois, T. Yamasaki, R.P. Muller, and W.A. Goddard III, "Rule-Based Wave Function for Generalized Valence Bond Theory", *J. Phys. Chem.*, **98**, 13498 (1994).

It should be noted that the ordering of basis functions in the AO expansion used to construct the initial guess must be the same as in the GTO block above. Within P-type shells, the basis functions are ordered as X, Y, Z, and within D-type shells as XY, XZ, YZ, XX, YY, ZZ.

3.3.1 Mixing basis sets - Custom Basis File Method

It is straightforward to mix different basis sets for the same molecule. To do that, one needs to create a separate basis file which will contain all required basis functions. Here is an example calculation of a hydrogen molecule with STO-3G and 3-21G basis set used for the first and the second hydrogen atom, respectively:

mixedbasis

```
ATOM    Hx
S 3
             3.4252509    0.154328967295
             0.6239137    0.535328142282
             0.1688554    0.444634542185


ATOM    Hy
S 2
             5.4471780    0.156284978695
             0.8245472    0.904690876670
S 1
             0.1831916    1.000000000000
```

and likewise a concatenated `.ao` file must be created. In this case:

mixedbasis.ao

```
Hx 0 1
1 1
1.000000e+00

Hy 0 1
2 2
-3.734069e-01
-7.173244e-01
1.255539e+00
-1.096019e+00
```

Note that the atomic number and mass is inferred from the first letter(s) of the element (H in this case). With the following files, we will get a mixed basis set calculation for H_2:

h2mix.xyz

```
2

Hx      0.0000000047    0.0000000098   -0.3735490915
Hy     -0.0000017747   -0.0000051727    0.3650468754
```

start.h2mix

```
basis           mixedbasis
coordinates     h2mix.xyz
method          b3lyp
run             energy
end
```

Note that a user can use a local copy of the basis directory if desired by setting the `TeraChem` environment variable appropriately. In that case, one should also ensure that a copy of the `license.dat` or `license.key` file exists in `$TeraChem`.

3.3.2 Mixing Basis Sets - Multibasis Method

A second way to use multiple basis sets is possible when one desires to use the same basis set for all atoms corresponding to the same element. For example, if one wishes to use the LANL2DZ basis set for Fe and to use the 3-21G basis set for C and O in $Fe(CO)_5$. In this case, one can use the `$multibasis` input as exemplified below:

FeCO5.inp

```
1       basis 6-31g
2       $multibasis
3       Fe lanl2dz_ecp
4       C 3-21g
5       O 3-21g
6       $end
7       method b3lyp
8       coordinates feco5.xyz
9       run energy
10      end
```

In this case, any atoms other than Fe, C, O will be treated with a 6-31G basis set, as specified on line 1. The `$multibasis` keyword signals the beginning of the list of atoms and basis sets that will override the default 6-31g basis specified in line 1. The `$multibasis` section **must** be terminated with `$end` (line 6 above). Within the `$multibasis` section, simply list the elements and the basis to be used for that element, as is done in lines 3-5 above. When the Multibasis method is used, TeraChem will create two files in the scratch directory: `JobName.multibasis` and `JobName.multibasis.ao`. These are concatenated basis set and guess files, respectively (similar to what one would create using the Custom Basis File Method described above). Note that JobName is replaced by the jobname of the current job.

3.4 Fixing Atoms in Molecular Dynamics

It is possible to fix a number of atoms in molecular dynamics by listing the atom numbers in a file called `fixed_atoms`, located in the directory containing the input file. The format is simply the number of fixed

atoms followed by the atom number (counting starts at zero) of each fixed atom, with one atom per line. The force on fixed atoms is simply zeroed out, with no modification to the forces on other atoms. Only QM atoms can be fixed in QM/MM simulations. This is equivalent to setting the masses for the fixed atoms to infinity. It does not have any effect on optimizations.

3.5 Ab Initio Steered Molecular Dynamics (AISMD)

Specific atoms can be subject to constant pulling forces to generate force-modified potential energy surfaces (or molecular dynamics on these FMPESs).[2] There are two different ways to do this: fixed or adaptive steering.

3.5.1 Fixed Steering

In fixed steering, a set of atoms are pulled towards lab-frame fixed pulling points. Because the pulling points are specified in the lab frame, the resulting potential energy surface depends on the orientation of the molecule (the pulling points do not rotate with the molecular frame). This is the best way to proceed when one wants to use molecular dynamics in the steering context. The potential energy surface is modified as:

$$V_{total}(r) = V_{ab-initio}(r) + \sum_{i}^{N_{attach}} F_0 * (||r_i^{fix} - r_i|| - ||r_i^{fix} - r_i^0)$$

where F_0 is the steering force, r_i^{fix} is the position of the ith fixed pulling point, r_i^0 is the initial position of the ith atom, and N_{attach} is the number of atoms being pulled.

Fixed steering requires one to specify the number of atoms subject to steering forces (N_{attach}), the identity of the atoms which are being steered, the fixed (in Cartesian space) points they are being pulled toward (r_i^{fix}), and the magnitude of the force (F_0). This is accomplished with the `steering` file (located in the same directory as the input file) which has the following format:

```
Number_Steered_Atoms
Steered_Atom_Index Fixed_Point_X Fixed_Point_Y Fixed_Point_Z Steering_Force
```

The last line occurs `Number_Steered_Atoms` times. Coordinates of the fixed points should be given in bohr and the steering force should be given in atomic units (1nN=0.012138 a.u.). Note: the code supports up to 16 steering atoms. Also, note that the "`Steered_Atom_Index`" starts counting from zero.

Alternatively: use `steering` keyword in input file to point to custom `filename/path` for above file.

3.5.2 Adaptive Steering

Adaptive steering specifies the direction of the force to be along the separation between two atoms. In contrast to fixed steering, the resulting energy is invariant to rotations of the molecule. Effectively, the pulling forces are being defined in the molecular frame. The potential energy surface is modified as:

$$V_{total}(r) = V_{ab-initio}(r) + \sum_{i}^{N_{pairs}} F_0^i * (||r_i^{left} - r_i^{right}||$$

where N_{pairs} is the number of atom pairs subject to pulling forces, F_0^i is the steering force for the ith pair, and r_i^{left} and r_i^{right} are the positions of the atoms being pulled away from each other (or pushed towards each other if the sign of the steering force is negative) in the ith pair.

[2] M. T. Ong, J. Leiding, H. Tao, A. M. Virshup and T. J. Martinez, J. Amer. Chem. Soc. **131** 6377 (2009).

Adaptive steering is not recommended for molecular dynamics simulations, but is most appropriate for geometry optimizations and minimal energy path calculations. An example input file with two pairs of atoms subject to adaptive steering forces is:

adaptive_steer.inp

```
basis 6-31g
method b3lyp
coordinates c6h6.xyz
run minimize
steering adaptive
steeratom1 1,4
steeratom2 6,5
steerforce 0.012,-0.012
end
```

In this input file, there is a pulling force of 1nN (0.012 atomic units of force) between atoms 1 and 6 (tending to expand the distance between atoms 1 and 6) and also a pushing force of 1nN between atoms 4 and 5 (tending to compress the distance between atoms 4 and 5). Note that there must be *no whitespace* between the atom numbers in the `steeratom` lines. Also, note that the atom indices start counting from one. Finally, the maximum number of pairs of atoms subject to steering forces is currently five.

3.6 Specifying Values of the Hubbard Parameter for DFT+U

Values of U can be assigned in one of two ways. The default is to read the values, specified in eV, from a fifth column on the right in the xyz file. For each atom, the specified U value will be applied to the default shell for that element. Atoms for which no value is specified will either have no U correction applied (default), or will be set to a built in default value if the `dft+u_default` parameter is set to `yes`.

Alternatively, custom shell assignments can be set in a text file specified by the `hubbard_input` parameter. Each atom type to be specified is listed sequentially, followed by the shell(s) to which a U value should be applied, and the corresponding value(s) in eV. If different values are desired for different atoms of the same element, element names can be numbered (e.g. "C1") in the hubbard file, and correspondingly in the xyz file. The following xyz and hubbard file are for a water dimer in which U values are assigned to all hydrogen 1s shells, and to the 2s and 2p shells of one of the two oxygen atoms.

hubfile.txt

```
O1
2S 2P
2 3
H
1S
1
```

water_dimer.xyz

```
6
  water dimer
O1   -0.702196054  -0.056060256   0.009942262
H    -1.022193224   0.846775782  -0.011488714
H     0.257521062   0.042121496   0.005218999
O     2.268880784   0.026340101   0.000508029
H     2.645502399  -0.412039965   0.766632411
H     2.641145101  -0.449872874  -0.744894473
```

3.7 Specifying Values of linear potential terms in DFT+U

A linear potential, α, may be applied to a subshell in lieu of or in addition to U on each subshell by setting `dft+u_alpha` to `yes`. The linear potential must be specified in a text file specified by the `alpha_input` parameter. Each atom type to be specified is listed sequentially, followed by the shell(s) to which an α value should be applied, and the corresponding value(s) in eV.

3.8 Specifying user defined atomic radii for PCM cavity

Other than the built in atomic radii, users can specify the radii for each type of element present in the geometry file. To activate this feature, the parameter `pcm_radii` needs to be set to `read`, and `pcm_radii_file` should be set to the **absolute** path to the atomic radii file. The file should be formatted based on the following rules.

1. Each line stands for one atom type. The first column is atomic number, second column is radius in Å
2. Separate the two columns by whitespace (TABs or SPACEs)
3. It does not matter if more atom types than needed are provided. E.g. If the atomic radii file contains information for all elements, but the geometry file contains only a few types, the program will run successfully. However, if the radii are not specified for some elements present in the geometry file, the program will exit with an error.

Following is an example radii file.

radii.txt

```
1 1.301
8 1.7221
```

This file contains the radii of Hydrogen and Oxygen atoms. It can only be used for systems with only Hydrogen and/or Oxygen atoms.

3.9 Output files

In addition to the information displayed on the screen, TeraChem creates several output files.

All calculations:

- `scr/c0` (`scr/ca0` and `scr/cb0` in UHF and UKS jobs) – the converged WF binary file containing the MO coefficients `C[i][j]` where `i` (row) is the MO and `j` (column) is the AO basis function index. This file(s) can be used as an initial WF guess in subsequent calculations. For UHF/UKS jobs, the files `ca0` and `cb0` are the coefficients for the alpha and beta spin orbitals.
- `scr/charge_mull.xls` – a tab-separated file containing Mulliken atomic charges.
- `scr/mullpop` – Detailed Mulliken population analysis including atomic spin densities
- `scr/results.dat` – Summary of results including final energy, center of mass coordinates, and dipole moment
- `scr/jobname.basis` – Normalized basis set information for the current run (`jobname` may be specified in the input file, otherwise it is automatically determined from the name of the `coordinates` file – see above)

- `scr/jobname.geometry` – Geometry information for the current run
- `scr/jobname.molden` – Geometry and MO information for current run
- `scr/bond_order.mat` – Bond order matrix if requested with `bond_order_mat` keyword
- `scr/bond_order.list` – Bond order matrix in list format if requested with `bond_order_list` keyword

Project Jobs

- `scr/prjct` (`scr/prjcta` and `scr/prjctb` in UHF and UKS jobs) – the converged WF projected onto another (usually, larger) basis set. These files are generated by `project` jobs and used as an efficient initial guess.

Geometry Optimization/Transition State Search:

- `scr/charge.xls` Atomic charges for each optimization step. Atoms are ordered by columns and rows correspond to different optimization steps
- `scr/optlog.xls` – a tab-separated file containing 7 columns of which only one (the first one) is currently used. The first column contains the SCF energy during geometry optimization or transition state search.
- `scr/optim.xyz|scr/optim.pdb` – geometry optimization or transition state search (depending on the job type) trajectory file in the XMol or PDB format. The trajectory can be visualized by VMD. If it is in XYZ format, it can be visualized with MolDen. In cases where an NEB calculation is being performed, this file contains a single frame which is the putative transition state (corresponding to the climbing image).
- `scr/spin.xls` - Excess spin on each atom in the same format as the `scr/charge.xls` file. This is only produced if an unrestricted (UHF/UKS) calculation is performed.

Transition State Search:

- `scr/neb_n.xyz` – an XMol file containing trajectory of the n^{th} NEB image. The last image (i.e. the `neb_10.xyz` if `min_image` equals `10`) is the actual transition state that that is also stored in `optim.xyz` file.
- `scr/nebinfo` – contains energies of all (`min_image-1`) NEB images along the converged NEB path.
- `scr/nebpath.xyz` – contains XYZ coordinates of all (`min_image-1`) NEB images along the converged NEB path.
- `scr/img_n.molden` – MolDen-readable file with coordinates and mooecular orbital coefficients for the n^{th} NEB image.

Frequency Calculation:

- `scr/Hessian.restart` – a binary file which can be used for restarting frequency calculations

- `scr/Hessian.bin` – a binary file containing the Hessian, i.e. force constant matrix. If this is present, it will be reused for subsequent frequency calculations. For example, this means you can run the thermochemical analysis for different temperatures (or different atomic masses) without recomputing the Hessian matrix.
- `scr/Frequencies.dat` – Formatted output containing the normal modes (in Cartesian coordinates) and their frequencies. These are after removal of rotation and translation, so there are normally $3N_{atoms}$-6 normal mode vectors. This file is also usually used as input for **FMS90**.

Initial Condition Generation:

- `scr/Hessian.restart` – see Frequency Calculation above
- `scr/Hessian.bin` – see Frequency Calculation above
- `scr/CentralGeometry.initcond.xyz` – XMol format file with the reference geometry used for the initial condition generation
- `scr/Geometry.initcond.xyz` – XMol format file with a geometry selected by initial condition sampling. Use this as an initial geometry in a susbsequent MD run.
- `scr/Velocities.initcond.xyz` – XMol format file with velocities selected by initial condition sampling. Use this along with `Geometry.initcond.xyz` in a subsequent MD run.
- `scr/Frequencies.initcond.dat` – This file contains the reference geometry and atomic masses. It is not of interest to most users, but serves as input for a quantum dynamics code (**FMS90**).

MD simulation:

- `scr/log.xls` – a tab separated file containing 8 columns: 1) time, 2) SCF energy, 3) currently not in use, 4) Kinetic energy, 5) Temperature, 6) Total energy (SCF + Kinetic), 7) HOMO energy, 8) LUMO energy. All times/energies are in Hartree and the temperature is in degrees Kelvin. In NVT dynamics, the Total energy does not include the contribution from the damping force, and thus should not be conserved.
- `scr/coors.xyz` – the MD trajectory geometry file in the XMol format. The trajectory can be visualized by VMD. Units are Angstroms. This is not created if `MDBinaryOutput` is set to true.
- `scr/vel.log` – contains atomic velocities along the MD trajectory. The format is similar to that of `scr/coors.xyz`except there is a blank line separating each set of velocities. Units are the same as those used by AMBER – Angstroms/(1/20.455) ps. This is not created if `MDBinaryOutput` is set to true.
- `scr/coors.dcd` – the MD trajectory geometry file in binary DCD format. The trajectory can be visualized by VMD. This is only created if `MDBinaryOutput` is set to true.
- `scr/vel.dcd` – contains atomic velocities along the MD trajectory in the binary DCD format. This is only created if `MDBinaryOutput` is set to true.
- `scr/restart.md` – a binary MD restart file containing all information (wavefunction, coordinates, velocities, etc) required to restart an MD job. By default, this information is stored at every 100^{th} MD iteration. A user can change it by specifying `restartmdfreq` parameter in the start file.

- `scr/restart.mdRnd` – a binary file which allows one to exactly restart the random number generator when an MD run is restarted. This ensures that the results of a set of shorter runs concatenated together will exactly follow the results of a single longer run with the same initial random seed. Such agreement is automatic for methods that do not involve any random numbers, but is otherwise not guaranteed (for example in Langevin dynamics). This file is automatically used on restart, i.e. *both* the `restart.md` and `restart.mdRnd` files are needed for restarting MD runs.

PCM:

- `scr/ratom.txt` – a text file listing the actual radii used for each PCM cavity sphere around each atom, i.e. the values reflects the cavity radii scaling factor. Each row stands for one atom, and has the same order as the input geometry file.
- `scr/sas.xyz` – a .xyz file containing all the PCM grid points. The element name in the first column is not meaningful and is only filled in to allow visualization with any program that can display molecular structures in XMol format. Only created if `print_ms` is set to true.

ESP/RESP:

- `scr/esp.xyz` – The ESP grid points in Å, together with the ESP on that point. Each row stands for one grid point.

  Column 1: The element type of the atom that the grid point originates from.

  Column 2-4: coordinates of the grid point

  Column 5: Electrostatic potential (ESP) on that grid point.

  Column 6: The index of the the atom that the grid point originates from. Order of the index is the same as the molecule in the input deck.
  When we use software to visualize this xyz file, only data in the first 4 columns is read by the software, though sometimes the 5th column can also be recognized and presents in labels (Molden).

4 More details on some TeraChem capabilities

4.1 Wavefunction projection

Sometimes, especially when transition metals or diffuse basis functions are present in a system, the SCF procedure does not converge due to an insufficiently accurate initial guess. In such cases, it often helps to converge the wavefunction using a smaller basis set (mini, sto-3g, etc) with no diffuse functions, then project the solution onto the desired basis set and start another SCF procedure using the projected wavefunction as initial guess. Below is an example of such a job. Here, the calculations are first performed in a smaller (`sto-3g`) basis set, and the converged wavefunction is projected onto the `6-31g**` basis set.

start.sto.prjct

```
run             energy
method          rhf
basis           6-31g**
projectbasis    sto-3g
charge          0
guess           project
coordinates     water.xyz
end
```

4.2 Frequencies and Thermochemical Corrections

TeraChem can calculate the Hessian, i.e. force constant, matrix using finite differences of analytic gradients. This provides the vibrational frequencies for a molecule. Simply use **run frequencies** to do this. The output will contain thermochemical analysis at the desired temperature (by default this is 300.00 K, but that can be changed with the `T0` keyword). If the frequencies calculation is interrupted, it can be restarted from the checkpoint information stored in `scr/Hessian.restart`. This restart will be automatic – if you ask for a frequency calculation, TeraChem will check for the existence of a `scr/Hessian.restart` file. If it is found, and the file corresponds to the same number and types of atoms, then TeraChem will carry on from where it left off. The final Hessian matrix is stored in the file `scr/Hessian.bin`. If this file exists when a frequency calculation is requested, TeraChem will simply read the Hessian matrix from the file and carry out normal mode analysis. This is an easy way to repeat the thermochemical analysis at a different temperature.

The thermochemical analysis is performed at the specified temerature (at 300.00 K, if no temperature is specified) and a pressure of 1 atm, and free energy corrections are computed. The list of corrections computed is:
a) non-thermal corrections, i.e., zero point energy (ZPE)
b) thermal corrections, i.e., vibrational, rotational, and translational energies
c) enthalpic correction
d) entropic correction

The inner energy, U is computed as

$$\mathrm{U} = E_{\mathrm{elec}} + \mathrm{ZPE} + E_{\mathrm{vib}} + E_{\mathrm{rot}} + E_{\mathrm{trans}} \,,$$

where E_{elec}, E_{vib}, E_{rot}, and E_{trans} are electronic, vibrational, rotational, and translational energies. E_{rot} and E_{trans} are computed using the number of rotational and translational degrees of freedom, f, of the molecule, respectively. $f = 2$ for rotaional degrees of freedom if the molecule is linear, otherwise, $f = 3$. For translational degrees of freedom, $f = 3$ for all systems.

$$E_{\text{rot}} = \frac{f}{2} k_b \text{T}$$

$$E_{\text{trans}} = \frac{3}{2} k_b \text{T}$$

The enthalpy, H, is computed as

$$\text{H} = \text{U} + k_b \text{T}$$

The entropic corrections are multiplied by temperature to have the units of energy. They are computed as:

$$\text{TS}_{\text{total}} = \text{TS}_{\text{elec}} + \text{TS}_{\text{vib}} + \text{TS}_{\text{rot}} + \text{TS}_{\text{trans}} \,,$$

where S_{elec}, S_{vib}, S_{rot}, and S_{trans} are electronic, vibrational, rotational, and translational entropies.

The rotational entropy, S_{rot}, is computed as:

$$\text{S}_{\text{rot}} = R * [\ln(\frac{8\pi^2}{\sigma}) + \frac{3}{2} \ln(\frac{2\pi k_b T}{h^2}) + \frac{1}{2} \ln(I_A I_B I_C) + \frac{3}{2}]$$

where I_A, I_B, and I_C are the principal moments of inertia of the molecule, and σ is the symmetry number of the molecule.

The translational entropy, S_{trans}, is computed as:

$$\text{S}_{\text{trans}} = R * [\frac{3}{2} \ln(\frac{2\pi m k_b T}{h^2}) + \ln(V) + \frac{5}{2}]$$

where V is the volume of the system.

The Gibbs free energy is computed as

$$\text{G} = \text{H} - \text{TS}_{\text{total}}$$

$$\text{G} = \text{U} + k_b \text{T} - (\text{TS}_{\text{elec}} + \text{TS}_{\text{vib}} + \text{TS}_{\text{rot}} + \text{TS}_{\text{trans}})$$

$$\text{G} = E_{\text{elec}} + \text{ZPE} + E_{\text{vib}} + E_{\text{rot}} + E_{\text{trans}} + k_b \text{T} - \text{TS}_{\text{elec}} - \text{TS}_{\text{vib}} - \text{TS}_{\text{rot}} - \text{TS}_{\text{trans}}$$

The free energy correction to electronic energy ($\text{G} - E_{\text{elec}}$) is given by:

$$\text{G} - E_{\text{elec}} = \text{ZPE} + E_{\text{vib}} + E_{\text{rot}} + E_{\text{trans}} + k_b \text{T} - \text{TS}_{\text{elec}} - \text{TS}_{\text{vib}} - \text{TS}_{\text{rot}} - \text{TS}_{\text{trans}}$$

4.3 MD with spherical boundary conditions

In an MD simulation, it is possible to impose spherical boundary conditions to prevent "evaporation" events or constrain a system to a given density. The spherical boundary conditions are provided in the form of a sum of two harmonic terms,

$$U_{constr}(r) = k_1 \left((r - R_{center}) - R_1\right)^2 + k_2 \left((r - R_{center}) - R_2\right)^2 \,,$$

where R_{center} is the center of mass of the system calculated for the first MD frame and then fixed. By default, k_1 is set to 10.0 kcal/(mol Å^2) and k_2 is set to zero. The simplest way to impose spherical boundary conditions is to set


```
mdbc            spherical
md_density      1.0
```

in the start file. `md_density` specifies the density of the system in g/mL used to automatically adjust R_1. If `md_density` is not provided, R_1 needs to be set explicitly using the `md_r1` keyword. Note that constraining forces are not applied to hydrogen atoms by default (but this behavior can be changed with the `mdbc_hydrogen` keyword).

4.4 Nanoreactor

A key part of the *ab initio* nanoreactor[3] is the use of TeraChem for fast molecular dynamics. Nanoreactor dynamics simulations involve the use of spherical boundary conditions as described above, but also the use of a virtual piston to accelerate reactions. This "piston" is accomplished through the use of time-dependent spherical boundary conditions. In TeraChem, this is possible by using the `mdbc_t1` and `mdbc_t2` input parameters which specify the number of time steps that the first and second set of spherical boundary conditions should be active, respectively. For example:

nanoreactor.inp

```
coordinates nanoreactor.xyz
basis       3-21g
tinit       1200
dispersion  no
thermostat  langevin
t0          1500
lnvtime     200
convthre    0.005
levelshift  yes
levelshiftvala 0.3
levelshiftvalb 0.1

run         md
method      uhf
scf         diis+a

timings     yes
nstep       30000
maxit       300

mdbc                spherical
md_r1                     6.0
md_k1                     3.0
md_r2                     4.0
md_k2                     5.0
mdbc_hydrogen             yes
mdbc_mass_scaled          yes
mdbc_t1                   750
mdbc_t2                   250
end
```

[3] L.-P. Wang, A. Titov, R. McGibbon, F. Liu, V. S. Pande and T. J. Martinez, Nature Chem. **6 1044 (2014)**

The initial temperature is set on line 3 to 1200K. A Langevin thermostat (line 5) is used to keep the temperature at 1500K (line 6). The thermostat response time is set in line 7. In order to promote convergence to UHF solutions, a level shift is used with different values for the alpha and beta electrons (lines 9-11). Spherical boundary conditions are employed (line 21), with two different sizes (lines 22-23 for the larger sphere and lines 24-25 for the smaller sphere). The larger sphere boundary conditions are operative for the first 750 time steps (line 28) and then the sphere is compressed for the next 250 time steps (line 29). Then the larger sphere will again be used for the next 750 time steps, followed by the smaller sphere for the next 250 time steps, and so on. Note that the spherical boundary conditions are applied to hydrogen atoms (line 26). Note also that the boundary conditions are applied in a mass-scaled fashion (line 27), which avoids stripping of hydrogen atoms when the sphere is compressed (see the original article for details). Some experimentation will be needed to determine appropriate values for the spherical boundary parameters: `md_r1`, `md_k1`, `md_r2`, and `md_k2`. Analysis of nanoreactor dynamics and refinement of minimal energy pathways associated with discovered reactions is part of a separate package which is not yet distributed with TeraChem (but ultimately will be made available).

4.5 QM/MM Functionality

There are various forms of QM/MM functionality, including:

1. Built in water MM models
2. Built in OpenMM interface with Amber formatted input files
3. Via the Amber-Terachem client/server protocol bufffers interface with TCPB
4. Via the Amber-Terachem file-based interface

In the first two options, the MD simulation is driven by TeraChem. In the last two options, the MD is driven by Amber. Please refer to the Amber manual for more information on how to do QM/MM simulations with TeraChem.[4]

4.5.1 Built in water MM models

For this functionality, only water molecules are available for MM treatment and they can be modeled using: TIP3P, SPC, SPCE, TIP4P, SWM4, SWM4-DP, or SWM4-NDP. TIP3P is used by default. If a flexible water model is chosen, the user should define this with the `rigidbonds` keyword set to `false`. One signals that a QM/MM calculation is desired by including the `qmmm` keyword in `inputfile`. The keyword specifies the location of a file with coordinates for the desired MM water molecules. Note that the coordinates of the MM waters are *not* in the `coordinates` file. The water molecule positions must appear in a specific order – grouped by water molecule and in the order O,H,H. We provide an example for one QM water molecule surrounded by one MM water molecule.

inputfile:

```
basis           6-31g
qmmm            mmwater.xyz
coordinates     qmwater.xyz
run             minimize
end
```

[4] http://ambermd.org

qmwater.xyz:

```
3

 O           6.144353       -0.788526       -6.483525
 H           6.378922       -1.564817       -5.989856
 H           6.475062       -0.043630       -5.995619
```

mmwater.xyz:

```
3

 O           6.450776        0.962914       -3.536427
 H           6.338156        1.514355       -2.771897
 H           6.519847        1.524502       -4.303300
```

4.5.2 Built in OpenMM interface with Amber formatted input files

In order to run QM/MM with the built-in OpenMM interface, the Terachem input file must contain the following keywords:

```
prmtop          waters.prmtop
coordinates     waters.rst7
velocities      waters.rst7
qmindices       qmregion.dat
```

The `velocities` keyword should only be specified if the rst7 file contains velocities and if the user wants to use them in TeraChem.

Atomic masses used in the QM region are read from the prmtop file.

The prmtop file is a typical Amber parm7 file (see Amber manual on tleap for how to generate an Amber parameter file), with the following constraints:

- The prmtop file must not have definitions for periodic boundary conditions, which are currently not supported by TeraChem.
- The prmtop file must have element types compatible with TeraChem. Alternatively, it is possible to specify the atom type of each atom in the QM region by adding a second column with this information in the file passed to the qmindices flag (qmregion.dat, in the example above)
- If spherical boundary conditions are desired, the prmtop and rst7 should be cut such that the system is spherical and the biasing forces to ensure spherical boundary conditions should be specific inside the TeraChem input file, and will be controled by TeraChem

The `qmindices` flag contains the path to a separate text file where each atom in the QM region has its own line. The first column specify which atoms in the prmtop file should be in the QM region (atom numbering starts from 0). The second column is optional and specifies the atom type to be used for a given atom in the TeraChem calculation. If the second column is not provided, TeraChem will assign atom types in the quantum calculation based on the element types specified in the prmtop file. In the example below, we pass a file to `qmindices` that specifies two water molecules in the QM region (the first two in the prmtop file) and use custom atom types for the first molecule (Oa and Ha, that allows us to use custom basis sets for these atoms) while using regular atom types for the second molecule:


```
0 Oa
1 Ha
2 Ha
3 O
4 H
5 H
```

Note: if atomic constraints are used in TeraChem, for example freezing particular atoms, the atomic indices start from 1 (even though the QM region file indices start from 0).

The coordinates file should be in Amber inpcrd or rst7 format. If no velocities are present in this file or if the user does not want to use velocities from this file, the user shall not specify the `velocities` keyword in the input file and TeraChem will randomly generate velocities from a random seed. This number can be modified with the `seed` keyword. For calculation types othen than `run md`, the velocities are ignored.

For minimizations, unless a substantial portion of the system is frozen, cartesian coordinates should be used (`min_coordinates cartesian`).

For NEB calculations, the `coordinates` keyword must be filled by `neb1.rst7`, with the other endpoint named `neb2.rst7` and contained in the input folder.

4.5.3 TeraChem/Amber Protobuf interface

In addition to the file interface, TeraChem has been expanded to include a client/based Protocol buffers interface to Amber.[5] This interface allows access to multiple Amber features while taking advange of TeraChem's GPU-accelerated electronic structure calculations. See the Amber manual for more details.[6]

4.5.4 TeraChem/Amber file-based interface

It is also possible to use TeraChem with Amber 12 or higher for QM/MM calculations using a file based interface. However, the user is discoraged to use this interface since the Protobuf interface just mentioned above is faster. See the Amber manual for more details.[7]

4.5.5 Constrained QM/MM

QM/MM is typically applied to systems in which the QM and MM regions encompass distinct chemical identities, e.g. QM solute, MM solvent; QM active site+cofactor, MM protein. However, there are many situations where one wishes to include chemically identical particles in both the QM and MM regions, e.g. QM solute+solvation shell, MM solvent. This introduces a problem in the context of *ab initio* molecular dynamics (AIMD), since without additional constraints, QM and MM separation will break down over the course of the dynamics, leading to a mixing of particles treated at two levels of theory.

To preserve QM/MM separation during AIMD, TeraChem includes an implementation of the Flexible Boundary Layer using Exchange (FlexiBLE) method.[8] FlexiBLE belongs to a class of constrained QM/MM methods that add a bias potential in order to maintain QM/MM separation during AIMD. What is unique about FlexiBLE is that it rigorously preserves canonical ensemble averages and even dynamical properties are well reproduced, at least on sub-diffusional timescales for particles in the inner QM region. FlexiBLE is enabled with the keyword `qmmmbp flexible`.

For full details of FlexiBLE, the reader is referred to Ref. [8] . Briefly, a (classical) bias potential is added

[5] J. Chem. Phys. 158, 044801 (2023)
[6] http://ambermd.org
[7] http://ambermd.org
[8] Z. Shen, W. J. Glover, *J. Chem. Phys.*, 2021, **155**, p224112

to the system Hamiltonian of the form:

$$V^{\text{bias}}_{ijk\cdots} = -k_{\text{B}}T \log f_{ijk\cdots},$$

where k_{B} is the Boltzmann constant, T is the temperature, and $f_{ijk\cdots}$ is a normalized bias function of the nuclear degrees of freedom, $f_{ijk\cdots} \equiv f(\cdots, \mathbf{R}_i, \cdots, \mathbf{R}_j, \cdots, \mathbf{R}_k, \cdots) \geq 0$.

To ensure normalization with respect to all possible exchanges between identical particles, the bias function is written in terms of an unnormalized penalty function, $h_{ijk\cdots}$:

$$f_{ijk\cdots} = \frac{h_{ijk\cdots}}{\sum_L \hat{P}_L(h_{ijk\cdots})},$$

where $\hat{P}_L$ permutes the indices $ijk\cdots$. Formally, this would involve enumerating $(N_{\text{QM}}+N_{\text{MM}})!/(N_{\text{QM}}!N_{\text{MM}}!)$ unique terms in the denominator; however, with a careful choice of $h_{ijk\cdots}$ (described below), most permutations result in a negligible contribution and can be efficiently screened out with a tree algorithm.

Finally, the unnormalized bias function is constructed as a product of pair functions, g_{ik}, between each QM and MM particle:

$$h_{ij\cdots,kl\cdots} = \prod_{m=i,j,\cdots}^{N_{QM}} \prod_{n=k,l,\cdots}^{N_{MM}} g_{mn}$$

$$g_{ik} = \begin{cases} 1, & x_i < x_k \\ \exp(-\dfrac{\alpha^3(x_i - x_k)^3}{1+\alpha(x_i - x_k)}), & x_i \geq x_k, \end{cases}$$

where x_i and x_k are radial distances from the system's origin of the QM and MM particles respectively, and α controls the rate of decay of bias with QM-MM distance: larger values of α achieve better QM/MM separation. This parameter is set with the keyword `flexible_alpha` in units of Å^{-1}. The default of `flexible_alpha = 15` is recommended for most applications. The form of the bias and pair functions ensures that the resulting bias potential penalizes configurations in which any QM particle is found further from the system origin than any MM particle. Specifics of defining the system origin are discussed below.

FlexiBLE is implemented using the interface to OpenMM and a custom OpenMM plugin (https://github.com/wjg3/FlexiBLEPlugin). As in other QM/MM jobs, the user must construct the total system's topology and coordinate files and then provide a file with indices of the atoms to be included in the QM region with the keyword `qmindices`. It is the user's responsibility to ensure the initial coordinates and/or indices are chosen to have QM/MM separation. If there is too much initial QM/MM mixing, FlexiBLE will fail with an error message. The user also has to provide the information of their system in a separate file specified by the keyword `molecule_info`. This file describes the number of each kind of molecule and the atoms that make them up. Molecules of the same type must be grouped consecutively in both the topology and coordinate files.

FlexiBLE supports a wide variety of systems; however, its formalism ensures correct ensemble averages only if the bias potential is applied between identical molecules. As a result, for a single solvated solute with explicit QM solvent, the `molecule_info` file should only list the solvent molecules, otherwise the bias potential may be incorrectly applied also to the solute as it approaches the QM/MM boundary. As a result, FlexiBLE is not designed to prevent the solute from crossing the QM/MM boundary. It is therefore recommended that the user also fix an atom of the solute to the system origin (the center of the QM region).

Spherical boundary

The simplest (and default) QM/MM boundary geometry is that of a sphere, in which QM particles are biased to be within a sphere centered at the system origin, with the MM particles biased to be outside the

sphere. Note: FlexiBLE does not assume a fixed radius of the sphere, i.e., the volume of the QM region can fluctuate. Instead, the number of QM particles is assumed to be fixed (as defined in the file set by the keyword `qmindices`). This is in contrast to adaptive QM/MM methods that generally assume the QM region has a fixed volume, but allow the number of QM particles to fluctuate.

Two choices of system origin are possible. By default, the origin is taken to be the total system's center of mass. Gradient contributions from the bias potential's dependence on the center of mass are included so that the total classical energy and momentum are rigorously conserved. A second choice is to set the origin to a fixed point in space (default is 0,0,0). This is enabled with the keywords `flexible_usecom no` and `flexible_center`. When FlexiBLE is combined with a spherical droplet boundary condition (see Section 4.3), it is recommended that the user set the same origin for both methods.

The formalism of FlexiBLE means that the bias potential is applied to a single atom (or virtual site) of each molecule, specified by the input option `flexible_target`. It is recommended that an atom central to the molecule is chosen (e.g. the oxygen atom of water). Another choice (the default) is to apply the bias potential to the center of mass of each molecule, by setting `flexible_target` to -1. This is especially useful for molecules which do not have a central atom, such as benzene.

The user can define unique FlexiBLE parameters for each type of molecule. This allows, for example, a different QM region center for each molecule type. This is achieved by appending multiple values together for each input parameter - one for each molecule type. If only one value is provided, the code will apply that value to all molecule types. A mixed style input, with one value provided for some parameters and multiple values provided for others, is also supported for all boundary types except capsule (see below).

An example input is:

flexiblesphere.inp

```
prmtop flexiblesphere.prmtop
coordinates flexiblesphere.rst7
qmindices flexiblesphere.txt
basis 6-31g
method b3lyp
run md
timestep 0.5
thermostat nhc
t0 300
mdbc spherical
md_r1 25
md_k1 10
qmmmbp flexible
flexible_alpha 15
flexible_thresh 0.1
flexible_scale 0.5
flexible_maxit 10
flexible_target -1
flexible_type spherical
flexible_usecom yes
end
```

In this example, `flexible_alpha` refers to the exponent parameter of the pair function (defined above) in units of $Å^{-1}$. For most purposes, the default value of 15 maintains QM/MM separation to a high degree. Increasing the value will further improve QM/MM separation, but might require a smaller timestep to conserve total classical energy. `flexible_thresh` is a threshold for discarding negligible penalty functions following a particular permutation of QM-MM indices, $\hat{P}_L(h_{ijk\ldots})$. The default threshold of 0.1 is sufficiently small to

preserve ensemble averages while conserving total classical energy for most systems. A smaller threshold will include more QM-MM exchanges in the definition of the bias potential, at increased computational expense. Note: the FlexiBLE tree is swept through in an iterative fashion, with each iteration adaptively tightening the threshold by a factor of `flexible_scale`, until the denominator of the bias function, $\sum_L \hat{P}_L(h_{ijk\ldots})$ evaluated for the retained permutations, converges to within `flexible_thresh` between two sequential sweeps of the tree. For most systems, the algorithm converges within two iterations. The algorithm will fail with an error if convergence is not reached within `flexible_maxit` iterations.

Capsule boundary

For QM systems that are significantly extended in one spatial dimension compared to the other two (e.g. a conjugated oligomer), a spherical QM region is not particularly atom economical, and the cost of the QM calculation could be dominated by a large number of solvent molecules far from the solute. In these circumstances, it is preferable to use a QM/MM capsule boundary condition, with a geometry shown in Fig. 1.

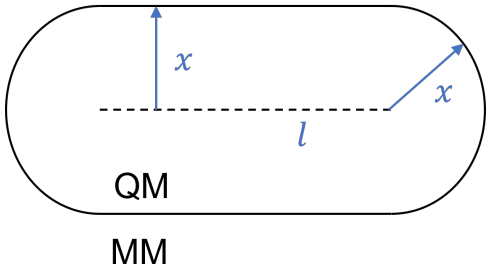

Figure 1: FlexiBLE QM/MM capsule boundary condition geometry

An example input for FlexiBLE capsule boundary conditions is:

flexiblecapsule.inp

```
prmtop flexiblecapsule.prmtop
coordinates flexiblecapsule.rst7
qmindices flexiblecapsule.txt
basis 6-31g
method b3lyp
run md
timestep 0.5
thermostat nhc
t0 300
mdbc spherical
mdbc_no_centering yes
md_r1 25
md_k1 10
qmmmbp flexible
flexible_alpha 15
flexible_thresh 0.1
flexible_scale 0.5
flexible_maxit 10
flexible_type capsule
flexible_usecom no
```

```
21      flexible_center 0.0 0.0 0.0
22      capsule_vector 0.0 1.0 0.0
23      capsule_length 7.0
24      end
```

Here the keyword `qmmmbp_boundary capsule` enables the capsule boundary condition, and the length of the capsule (variable l in Fig. 1) is set by keyword `capsule_length` in units of Å. In this example, we have also set the system origin to (0,0,0) with the keyword `flexible_usecom no`. Something worth noting is that `flexible_center` will be ignored if `flexible_usecom` is not turned off. Here, the capsule is centered at (0,0,0) and its long axis is defined by the `capsule_vector`, which is in the y direction. It is the user's responsibility to ensure the initial configuration places the QM region centered at `flexible_center` with the long-axis of the QM region aligned according to `capsule_vector`.

4.6 Overriding Atomic Masses

The atomic masses for elements in TeraChem are set according to standard values for the most abundant isotope. These can be changed if desired, which will affect frequency and/or molecular dynamics calculations. In order to override the default values, one uses the `$masses` keyword in the input file. An example is:

caffeine.inp

```
1       basis 6-31g
2       method b3lyp
3       coordinates caffeine.xyz
4       run md
5       end
6       $masses
7       C 13.0
8       15 H 2.0
9       $end
```

In this example, the first line `C 13.0` sets the masses for all C atoms to 13.0 amu. The second line `15 H 2.0` sets the mass for the 15th atom (counting from 1) to 2.0 and asserts that this atom is expected to be a hydrogen. If the 15th atom is not H, the code will assume that a counting error has been made and will abort.

4.7 Initial Condition Generation

TeraChem automates the construction of initial conditions for dynamics sampling from either Wigner or Husimi distributions (in the harmonic approximation). The `initcond` run mode initiates a calculation of the gradient at the chosen geometry. If the gradient is close to zero, the geometry is deemed a stationary point and TeraChem will compute the Hessian matrix. The geometry is shifted such that the center of mass is at the origin – the new geometry (and the masses used for the atoms) is written to `Geometry.initcond.dat` in the scratch directory. Translation and rotation are removed from the Hessian and the resulting frequencies and normal mode vectors are written to `Frequencies.initcond.dat` in the scratch directory. The central geometry used for the initial conditions (after translation so the center of mass is at the origin) is written to the file `CentralGeometry.initcond.xyz`. The positions and velocities for the chosen initial condition are written to the files `Geometry.initcond.xyz` and `Velocities.initcond.xyz`. These files are in a suitable format for reading with a subsequent dynamics run by using

```
coordinates Geometry.initcond.xyz
velocities Velocities.initcond.xyz
```

in the TeraChem input file. Finally, TeraChem writes the file `Hessian.bin` to the scratch directory – if this file is present in the scratch directory, TeraChem will use it instead of recomputing the Hessian matrix when generating initial conditions.

This mode can also be used for transition state sampling dynamics runs. In this case, special treatment of the imaginary frequency mode is required. This is controlled by the `initcondtssign` parameter. If `initcondtssign=0`, the chosen initial conditions will not be displaced along the imaginary mode at all and will have zero velocity along the imaginary mode. Otherwise, the position and velocity along the imaginary mode are chosen by sampling from the Wigner/Husimi distribution corresponding to a harmonic oscillator with the a frequency of the same magnitude as the imaginary mode, but purely real. The sign of `initcondtssign` ensures that all sampled initial conditions are on the same side of the transition state and with velocity that points away from the transition state. Which value of the sign corresponds to motion towards reactants and which corresponds to motion towards products can only be discerned by some test calculations (following the molecule away from the transition state long enough to determine if it is going towards reactants or products).

4.8 Ab Initio Molecular Dynamics Integrators

By default, molecular dynamics is carried out in the microcanonical (NVE) ensemble. In this case, the total energy should be conserved when a sufficiently small timestep is used (values of 0.5fs are normal for molecular systems). When computing dynamical properties, the NVE ensemble is the most appropriate (and some would argue this is the **ONLY** choice). However, if one is not concerned with dynamical properties but rather with sampling, it may be appropriate to carry out simulations in an alternative ensemble such as the canonical (NVT) one. Example uses for NVT dynamics include 1) equilibration, where initial conditions for NVE runs are obtained by using NVT dynamics and 2) calculation of free energy surfaces using umbrella sampling or related techniques. Several thermostats are available in `TeraChem`, and we provide details here. In this section, we are only discussing integrators for Newton's equations of motion for the nuclei. See (some other section) for information about the propagation of the electronic wavefunction along different molecular geometries in a trajectory.

4.8.1 Velocity Rescaling

This is the default integrator for molecular dynamics in TeraChem (`thermostat=rescale`). In this case, Hamilton's equations are solved (i.e. one is in the NVE ensemble):

$$\frac{dp_i}{dt} = -\frac{\partial V(q)}{\partial q_i}$$

$$\frac{dq_i}{dt} = p_i/m$$

where $V(q)$ in TeraChem is normally taken from an *ab initio* calculation (although this could also be QM/MM). However (at variance with the usual NVE ensemble), the velocities are periodically rescaled to give a kinetic energy corresponding to the desired temperature:

$$K_{actual} = \frac{1}{2}\sum_i m_i \vec{v}_i \cdot \vec{v}_i$$

$$K_{desired} = \frac{1}{2} N_{DOF} k_b T$$

$$\alpha = \sqrt{\frac{K_{desired}}{K_{actual}}}$$

$$v_i^{rescaled} = \alpha v_i$$

where k_b is Boltzmann's constant, T is the desired temperature according to the `T0` parameter in the input file, and N_{DOF} is the number of degrees of freedom in the system (presently calculated as $3*N_{atom}$ minus the number of constraints such as rigid bonds). The parameter `rescalefreq` controls the frequency of velocity rescaling. Velocity rescaling does not exactly reproduce the canonical ensemble and this has been shown to lead to some pathological behaviors for very long time simulations.[9] It is therefore not recommended in general. One usually runs NVE dynamics by simply setting `rescalefreq` to a time which is much longer than the simulation time desired.

4.8.2 Langevin Dynamics

There are two implementations of Langevin dynamics in TeraChem.

4.8.3 Bussi-Parrinello Stochastic Velocity Rescaling

This is the recommended thermostat for NVT simulations.

4.9 Integration with NBO6

This release of TeraChem is interfaced with the NBO6 package of Weinhold and coworkers.[10] NBO deletions are not currently supported, nor are analyses which require the matrix elements of the dipole operator. However, many of the usual natural bond orbital (NBO) and natural population analyses (NPA) are supported. Setting the `nbo` keyword in the `startfile` to `npa` or `full` will give natural population analysis or full NBO/NPA analysis. You may also use advanced keywords (documented in the NBO manual) if desired. In this case, the `nbo` keyword should be set to `advanced` and the `startfile` should contain a `$nbo` group as documented in the NBO manual. An example of this usage of advanced keywords for CH_3NH_2 is provided in `TeraChem/tests/nbo`. Note that the `NBOEXE` environment variable needs to point to the NBO6 executable file in order for successful integration with TeraChem. This is currently set in the `SetTCVars.sh` script. The NBO6 executable is included with the TeraChem distribution and resides in the TeraChem directory.

4.10 Ghost Atoms

Sometimes it is desirable to do calculations including the basis functions of an atom (or atoms) without including the nucleus or electrons corresponding to the atom. This is most often used to calculate basis set superposition error (BSSE). For example, the Boys-Bernardi counterpoise correction[11] for the interaction energy of fragments A and B is:

$$E_{AB}^{counterpoise} = E_{AB}\left(H_{AB}, R_A, R_B; A \cup B\right) - E_A\left(H_A, R_A; A \cup B\right) - E_B\left(H_B, R_B; A \cup B\right) ,$$

where the first arguments in the parentheses are the Hamiltonian and coordinates it depends on and the second arguments (after the semicolon) are the basis sets used. Thus, in the second term on the right, the calculation has the atoms of A (and the corresponding basis functions), but also includes the basis functions for fragment B (but *not* the electrons or nuclei corresponding to the atoms of B). In TeraChem, one can include ghost atoms (atoms which contribute basis functions to the calculation, but *not* electrons or nuclear charges) by prefacing the element name with the letter X. For example, the following is a coordinates (XYZ) file used in calculating the counterpoise-corrected interaction energy for water dimer:

[9] S. C. Harvey, R. K.-Z. Tan, and T. E. Cheatham III, J. Comp. Chem. **19** 726 (1998)
[10] http://www.chem.wisc.edu/~nbo6
[11] S. F. Boys and F. Bernardi, Mol. Phys. **19** 552-566 (1970)

waterdimer.xyz

```
6

O          -0.029481       -0.086991       -0.533710
H          -0.660044       -0.772698       -0.712077
H           0.086741        0.049151        0.406478
XO          0.306708        0.307377        2.262155
XH          1.109626        0.019913        2.679671
XH         -0.049150        1.088298        2.668550
```

The last three atoms will be "ghost" or "dummy" atoms, i.e. the system has only 10 electrons, one oxygen atom nucleus and two hydrogen atom nuclei, but basis functions will be used on all six listed atoms (according to the basis set listed in the `start` file).

4.11 Geometry Optimization and Transition State Search - DL-Find

The `instdir/tests/sp` directory contains a simple configuration file (`start.opt`) used for geometry optimization of a spiropyran molecule:

start.opt

```
# basis set
basis           6-31g
# coordinates file
coordinates     sp.xyz
# molecule charge
charge          0
# SCF method (rhf/blyp/b3lyp/etc...): RHF
method          rhf
# type of the job (energy/gradient/md/minimize/ts): geometry
# optimization
run             minimize
timings         yes
end
```

The optimization is triggered by the '`minimize`' keyword. Note that the `sp.xyz` file in fact contains two sets of coordinates (frames) one by one. In geometry optimization jobs, only the first frame is taken into account while the others (if any) are ignored. The second frame, however, is required by transition state search jobs and represents the second NEB endpoint in NEB calculations. The TS search is triggered by the '`ts`' keyword, i.e. file (`start.ts`)

start.ts

```
# basis set
basis           6-31g
# coordinates file
coordinates     sp.xyz
# molecule charge
charge          0
# SCF method (rhf/blyp/b3lyp/etc...): RHF
method          rhf
# type of the job (energy/gradient/md/minimize/ts): TS search
run             ts
nstep           200
timings         yes
end
```

Spring constants for the NEB path can be set with `min_nebk`. If not set in the input file, `min_nebk` defaults to a value of `0.01`.

Variable spring constants can be used in NEB in order to improve resolution of the path near the barrier. To use this feature, set `max_nebk` and `min_nebk` where `max_nebk` is the highest spring constant applied around the highest energy image and `min_nebk` is the lowest. If `max_nebk` $\leq$ `min_nebk`, the standard approach is employed.

If a `steering` file is present or the steering keyword with custom filename is used in the input file, force-modified nudged elastic band (FMNEB) will be run. The FMNEB currently only supports 2 attachment points/pulling points as defined earlier in the *ab initio* steered molecular dynamics section.

`$constraints ... $end` group in the configuration file allows one to impose geometrical constraints during optimization (frozen atoms, fixed bond lengths, angles, and dihedrals). `instdir/tests/constraints` directory contains two example start files for constrained geometry optimization jobs.

Constrained geometry optimization can be performed in any type of coordinates with two exceptions: 1) if DLC coordinates are used for optimization (the default), no atoms may be frozen and 2) if Cartesian coordinates are used for optimization (`min_coordinates cartesian`), only frozen atom constraints are allowed. Below is an example of a job configuration start file. Note that all constraints should be listed in separate lines, and enumeration of atoms begins from 1. Constrained coordinates will be constrained to the value they have in the starting geometry.

start.b-a-d

```
# basis set
basis           6-31g
# coordinates file
coordinates     C10H22.inp
# molecular charge
charge          0
# optimize geometry
run             minimize
end

$constraints
# bond connecting atoms 32 and 29
bond            32      29
# 32-29-26 angle
angle           32      29      26
# 32-29-26-27 dihedral
dihedral        32      29      26      27
$end
```

Finally, it is straightforward to freeze all hydrogen or non-hydrogen atoms by `atom hydrogens` or `atom heavy`, respectively. These last two options as well as the above format are only valid when using DL-FIND (i.e., `new_minimizer==no`).

4.12 Geometry Optimization - geomeTRIC

Early versions of TeraChem used the DL-FIND program for geometry optimization and minimum energy path searches. Starting in TeraChem 1.93P, a new optimizer is available which was developed specifically for TeraChem.[12] To access this optimizer, set `new_minimizer` to `yes` in the input file. Constraints are handled differently with the new minimizer. Specifically, you may choose to freeze certain degrees of freedom to their initial values by specifying these in the `$constraint_freeze` section:

start.min

```
basis           6-31g
coordinates     C10H22.inp
charge          0
run             minimize
new_minimizer   yes
end
$constraint_freeze
bond 5_6         # Freeze the bond between atoms 5 and 6 (atoms are
                 # numbered starting from 1)
bond 5_6,7_8     # Freeze bond lengths for atoms 5/6 and 7/8
angle 1_2_3      # Freeze bond angle between atoms 1-2-3
dihedral 1_2_3_4 # Freeze dihedral for atoms 1-2-3-4
xyz 5            # Freeze xyz coordinates of atom 5
xy 5-11,13,35    # Freeze xy coordinates of atoms 5-11, 13 and 35
$end
```

[12] L.-P. Wang and C. Song, J. Chem. Phys. **144** 214108 (2016).

You may also choose to set certain degrees of freedom to specific values (as above, but the initial values could be different from the desired constrained value):

start.min

```
basis           6-31g
coordinates     C10H22.inp
charge          0
run             minimize
new_minimizer   yes
end
$constraint_set
bond 1.5 5_6         # Set bond length b/t atoms 5 and 6 to 1.5 Angstrom
angle 30 1_2_3       # Set 1-2-3 bond angle to 30 degrees
angle 45 2_3_4,5_6_7 # Set 2-3-4 and 5-6-7 bond angles to 45 degrees
$end
```

Finally, you may direct the code to scan a particular set of variables, optimizing all other degrees of freedom for each set of scanned variables:

start.scan

```
basis           6-31g
coordinates     C10H22.inp
charge          0
run             minimize
new_minimizer   yes
end
$constraint_scan
bond 1.3 1.5 3 1_2  # Scan bond length b/t atoms 1 and 2 from 1.3A
                    # to 1.5A, with 3 equally spaced data points
$end
```

When using geomeTRIC (i.e., `new_minimizer=yes`), the name of the `$constraints ... $end` group is replaced by `$constraint_freeze ... $end` and the two additional groups `$constraint_set ... $end` and `$constraint_scan ... $end` become available. The format of these three groups is almost the same as shown in the above example (differences explained below). However, `atom` is replaced by `trans`. The keyword `trans` can be followed directly by arbitrary combinations of the letters `x`, `y` and `z`, which indicates that the specified constraint is only valid in the declared direction(s). The group `$constraint_set ... $end` can be used to set an internal degree of freedom to a certain value. This value has to be specified in front of the atom indices, i.e., behind the type of the constraint. For example `angle 12.5 1_2_3` would fix the angle formed by the first three atoms to 12.5 degrees. The group `$constraint_scan ... $end` can be used to scan an internal degree of freedom from a certain value to another value. These values have to be specified in front of the atom indices, i.e., behind the type of the constraint. For example `angle 5.0 12.5 16 1_2_3` would scan the angle formed by the first three atoms from 5.0 to 12.5 degrees, with 16 equally spaced data points, i.e., in steps of 0.5 degrees at 5.0, 5.5, 6.0, ..., 12.5 degrees.

4.13 Polarizable Continuum Model

4.13.1 Ground State PCM

TeraChem has the Polarizable Continuum Model (PCM) efficiently implemented on GPUs,[13] and the overall runtime for a PCM calculation is accelerated by more than 10X compared to CPU programs. All the imple-

[13] Liu, F.; Luehr, N; Kulik J. K.; Martinez, T. J., *J. Chem. Theo. Comp.* **2015**, *11*, 3131-3144

mented PCM variants belong to the conductor-like solvation model (COSMO), including the C-PCM[14] (also known as G-COSMO [15]) with smooth first order energy derivatives (ISWIG[16]/SWIG[17] cavity discretization), and the original COSMO by Klamt with a polyhedron grid.[18]

To enable PCM for ground state HF/DFT calculations, the minimal input parameters to specified are `pcm` and `epsilon`, as shown in the following example:

pcm.inp

```
basis        6-31g
method       rhf
charge         0
spinmult       1
maxit         100
gpus        1
# run type can also be gradient/minimize/md
run         energy
coordinates  c2h4.xyz
#here starts the pcm parameters
pcm          cosmo
epsilon         78.39
end
```

Here `pcm cosmo` turns on PCM calculation, and `epsilon` specifies the solvent dielectric. Other PCM related parameters take their default value, so this calculation will be a C-PCM ISWIG calculation with Lebedev Grid points at discretization level 17 (110 points/atom). To customize the PCM calculation, there are mainly 4 groups of parameters to specify:

1. **PCM Cavity Definition**:
   - If `pcm_grid=lebedev/ISWIG/SWIG`:
     `pcm_grid, pcmgrid_h, pcmgrid_heavy, pcm_radii, pcm_rad_scale, solvent_radius, pcm_radii_file`
   - If `pcm_grid=polyhedron`:
     `nspa, nppa, pcm_radii, pcm_rad_scale, solvent_radius, pcm_radii_file, cosmodsc`
   - If `pcm_grid=sphere`:
     `pcm_global_radius, pcmgrid_heavy`
2. **PCM related one-electron integrals:** `pcm_pair_driven, pcm_threoe`
3. **Conjugate Gradient solver:** `pcm_matrix, dynamiccg, cgtoltight, cgtolscale, dynamiccgtol, cgprecond, cgblocksize`

Details about these parameters are listed on page 81.

Among the parameters above, `pcm_grid` is the most important as it specifies the type of COSMO variant to use.

- `pcm_grid=iswig/swig` is always recommended for general-purpose PCM calculations, as these have smooth analytical first order energy derivatives which match perfectly with the corresponding numerical gradients.

[14] Barone, V.; Cossi, M., *J. Phys. Chem. A* **1998**, *102*, 1995-2001
[15] Truong, T. N.; Stefanovich, E. V., *Chem. Phys. Lett.* **1995**, 240,253-260.
[16] Lange, A. W.; Herbert, J. M., *J. Chem. Phys.* **2010**, *133*, 244111.
[17] York, D. M.; Karplus, M., *J. Phys. Chem. A* **1999**, *103*,11060-11079.
[18] Klamt, A.; Schuurmann, G.,*J. Chem. Soc. Perkins. Trans. 2* **1993**, 799-805.

- `sphere` uses a single spherical cavity to enclose the whole system, though the polarization charges are still presented as spherical gaussians as in ISWIG/SWIG models. This is different from the Onsager model for two reasons: 1) the spherical cavity surface is discretized and 2) the treatment includes solute contributions to the reaction field beyond the dipole term. The center of the spherical Cavity does not move with the COM of the solute molecule during MD simulations. This model can be activated only when spherical boundary condition is turned on (`mdbc=1`).
- `polyhedron` should not be used for production simulations as its gradients are not smooth. Visualization of its cavity, however, can give the users a good sense of why this type of grid results in non-smooth gradients and why ISWIG/SWIG should be used instead.
- `lebedev` uses Lebedev grids for building the cavity as ISWIG/SWIG, but does not have switching functions or spherical gaussians to present the polarization charges. It should not be used for production simulations, either. It give the users a sense of why ISWIG/SWIG should be used.

It is very helpful to visualize the PCM cavity. This can be easily done by generating the grid point coordinate file with `print_ms=1`. Please refer to page 28 for details.

4.13.2 Extreme Pressure PCM (XP-PCM)

The extreme pressure PCM (XP-PCM) [19] is an implicit pressure model to simulate high pressure condition at the order of 1-300 GPa. In TeraChem, we implemented [20] the version of XP-PCM theory developed by Cammi *et al.*. Currently, only single point XP-PCM energy is working, whereas analytical gradient implementation is still under development. To enable XP-PCM for ground state HF/DFT calculations, the minimal input parameters are shown in the following example:

xppcm.inp

```
method          b3lyp
basis           aug-cc-pvdz
scf             diis
charge          0
spinmult        1
maxit           100
convthre        1e-6
gpus            1
run             energy

pcm             xppcm
pcm_grid        iswig
pcmgrid_heavy   35
pcmgrid_h       35
epsilon         2.0165
pcm_rad_scale   1.1
xppcm_nb        36
xppcm_mb        84.16
xppcm_rhob      0.779
coordinates     Ar.xyz
jobname         energy_f1.1_rhob0.779_g35
```

[19] R. Cammi, V. Verdolino, B. Mennucci, and J. Tomasi *Chem. Phys.* **2008**, *344*, 135
[20] A. Gale, E. Hruska, F. Liu *J. Chem. Phys.* **2021**, *154*, 244103

Here, `pcm xppcm` turns on XP-PCM calculation, and `epsilon` specifies the solvent dielectric. Other important XP-PCM related parameters are related to the solvent, including `xppcm_nb` (solvent valence electron number), `xppcm_mb` (solvent molecular weight), and `xppcm_rhob` (solvent number density). Notice that XP-PCM ISWIG calculations need higher density of grid points (at least at Lebedev level 35). To reflect the compression of molecular volume at high pressure, we shrink the radii scaling factor `f` to be no more than 1.2. Details about these parameters are listed on page 81.

The XP-PCM calculation only generates the free energy of the system at different volume. In order to obtain the corresponding pressure, one should calculate the free energy of the same system at different radii scaling factors (volumes), and then perform numerical fitting according to real gas equation. Please refer to reference [21] for details about the fitting.

4.14 ESP/RESP Charge

The Restrained Electrostatic Potential (RESP) method [22] is a widely used atomic charge derivation method in developing force field (AMBER). In TeraChem this method is efficiently implemented on GPU and the derived charges can be easily visualized with VMD.
To enable RESP charge derivation for ground state HF/DFT calculations, the minimal input parameters to specified is `resp`, as shown in the following example:

resp.inp

```
basis        6-31g
method       rhf
charge         0
spinmult       1
maxit        100
gpus        1
# run type can also be gradient/minimize/md
run         energy
coordinates  mymol.xyz
#here starts the resp parameters
resp yes
end
```

Here `resp yes` turns on RESP calculation. Other RESP related parameters take their default value, so this calculation will be a RESP calculation with grid points located on 4 layers of Connolly surface, whose radii are {1.2, 1,4, 1,6, 1,8 }× vdw_radii, and the density of the grid points on the Connolly Surface is 1.0 point/Å^2. When deriving the atomic charges, the quadratic restraint $a\sum^k((q_k^2+b^2)^{1/2}-b)$ is used, where `a` Controls the strength of restraint and takes the default value 0.0005 a.u., and `b` controls the tightness near the bottom of the restraint and takes the default value of $0.1e^-$. Users can customize the above parameters with the keywords listed in page 85
It is very helpful to visualize the RESP charge cavity. This can be easily done by visualizing the RESP generated file `scr/esp.xyz` together with the input molecule `mymol.xyz` with VMD, using representation "point" for `esp.xyz`, and other types of representation, say "CPK", "licorice" etc, for `mymol.xyz`. Please refer to page 28 for details of the format of `resp.xyz`

[21] R. Cammi *J. Comp. Chem.* **2015**, *36*, 2246-2259
[22] C. Bayly, P. Cieplak, W. Cornell, and P. Kollman *J. Phys. Chem.* **1993**, *97*, 10269-10280

4.15 ESP Calculation

The electrostatic potential can be written to a file by setting `epotential yes` in the input deck. When ESP computation is requested, the default is to construct a cubic grid surrounding the molecule and evaluate the ESP on these points. The default cubic grid begins 10Å before/after the atom centers defining the bounding box of the molecule along each coordinate and has 64 points along each of the x,y,z directions. The default output file is `scr/epotential.dx`. Users can provide their own list of gridpoints on which to evaluate the ESP by setting `ep_points` in the input deck. The format for a user-defined grid is:

ep_points.xyz

```
2 Number of gridpoints
Title Line which is ignored
anytext_or_number 10.0  0.0 0.0
anytext_or_number 0.0  10.0 0.0
```

The output format is either APBS[23] readable by Chimera[24] (when a TeraChem-generated cubic grid is used) or a simpler format shown below (when a user-defined grid is used by setting `ep_coordinates`):

epotential.dx

```
ESPValue(x_1,y_1,z_1) x_1 y_1 z_1
...
ESPValue(x_npoints,y_npoints,z_npoints) x_npoints y_npoints z_npoints
```

4.16 Hole-hole Tamm-Dancoff approximated (hh-TDA) density functional theory

The hole-hole Tamm-Dancoff approximated density functional theory (hh-TDA-DFT) method is an electronic structure method that allows efficient treatment of dynamic and static correlation effects. The method formally starts from a reference state with N+2 electrons. By annihilating two electrons, the N-electron ground aand excited states are obtained as solutions to the same eigenvalue problem. Hence, these states are handled on equal footing and the method is suited for photochemical simulations of systems with low-lying $\pi\pi^*$ and $n\pi^*$ states. The restriction is that the excited states of interest can well-described as excitations to the LUMO. Two orbital generation models are implemented, N+2 and fractionally occupied electron model. [25] [26] In the current implementation, this has some implications on the input structure as seen below (see orbital choice-related keywords): The hh-TDA method uses the same technical settings as for restricted CIS/TDA-DFT.

An example inputs are given below. For orbital generation for a double anion (i.e., (N+2)-electron) reference, we recommend the use of a range seprated hybrid functional. For example:

[23] http://www.apbs.org
[24] https://www.cgl.ucsf.edu/chimera
[25] C. Bannwarth, J. K. Yu, E. G. Hohenstein, T. J. Martínez, J. Chem. Phys., (2020), accepted; ChemRxiv DOI: 10.26434/chemrxiv.11828256.
[26] J. K. Yu, C. Bannwarth, E. G. Hohenstein, T. J. Martínez, in preparation.

start.md

```
method        wb97x
basis         def2-svp
guess         sad
precision     mixed
dftgrid       1
threall       1.0e-14
threspdp      1.0e-4
convthre      1.0e-6
xtol          1.0e-6
coordinates   coord.xyz

# hhtda settings
hhtda         yes
cisnumstates  10
hhtdasinglets 10
cisalgorithm  davidson
cisguessvecs  30
cismax        100
cismaxiter    1000
cistarget     1
cisconvtol    1.0e-6

cphftol       1.0e-6
cphfiter      100

# in N+2 reference run, charge has to be the actual charge - 2
charge       -2
spinmult      1
run           gradient
```

This calculation will lead to N-electron hh-TDA states that correspond to a charge-neutral molecule.

For a fractional occupation SCF for orbital generation with an electron number different from $N+2$, the input could look like:

start.md

```
method         bhandhlyp
basis          def2-tzvp
max_l_in_basis 2   # neglects any function higher than d
sphericalbasis yes # switch on spherical basis representation
guess          sad
precision      mixed
dftgrid        2
threall        1.0e-14
threspdp       1.0e-4
convthre       1.0e-8
coordinates    coord.xyz

# switch on FON
fon            yes
fon_method     constant
# the total number of electrons closed+active = N/2+1 to obtain N-electron hh-TDA states
closed         0
active         30

# hh-TDA related options
hhtda          yes
cisnumstates   10
hhtdasinglets  10
cisalgorithm   davidson
cisguessvecs   30
cismax         100
cismaxiter     1000
cistarget      1
cisconvtol     1.0e-6
cphftol        1.0e-8
cphfiter       100

# this determines the number of electrons in SCF
charge         0
spinmult       1
run            coupling
# coupling between ground and 1st excited state
nacstate1     0
nacstate2     1
```

This input implies that in the SCF, all N electrons will be smeared in the 30 ($=N/2+1$) active orbitals. In the hh-TDA calculation, the 30 orbitals will first be doubly occupied, generating an intermediate $N+2$ intermediate reference state. Then the ground and excited states are generated by annihilating two electrons within the entire active space. The electron number in the final hh-TDA states is 56 that are distributed among the 30 active orbitals.

4.17 Coupled-cluster theory

Coupled-cluster singles and doubles (CCSD) and equation-of-motion CCSD (EOM-CCSD) are available in TeraChem. These methods are implemented using a Cholesky decomposition of the electron repulsion integrals. A restricted, closed-shell reference function is required, but it need not be canonical (i.e. the Fock operator does not need to be diagonal in this basis). A simple configuration file to perform a CCSD calculation follows:

start.ccsd

```
basis           cc-pvdz
sphericalbasis  true
coordinates     geom.xyz
method          rhf
run             energy

threall         1.0e-14
convthre        1.0e-8
precision       double
guess           sad

charge          0
spinmult        1

ccbox                   yes
ccbox_ccsd              yes
ccbox_scratch_dir       ./scr.geom
ccbox_ccsd_t_convthre   1.0e-6
ccbox_cd_thresh         1.0e-4
end
```

This will compute the CCSD correlation energy following a Hartree-Fock calculation. The CC code freezes core orbitals by default. Following the solution of the CCSD amplitude equations, the correlation energy and D1 diagnostic are output:

out.ccsd

```
CCSD converged

  ------------------------------------------------------
  ====================> CCSD Energies <====================
  ------------------------------------------------------
  Reference  Energy         =    -227.8305188340249288 [H]
  Correlation Energy        =      -0.6665110167832837 [H]
  Total Energy              =    -228.4970298508082180 [H]
  ------------------------------------------------------

  D1 Diagnostic             =                 0.059025
```

An important consideration for CC calculations is where the temporary files are written. This is specified by the `ccbox_scratch_dir` option. In the example above, the temporary files are written the the scratch directory automatically created by TeraChem. This may not always be ideal and so the CC scratch directory can be specified separately. Note that the default scratch directory is `/tmp`.

To perform an EOM-CCSD calculation a few extra options are required:

start.eomccsd

```
basis                       cc-pvdz
coordinates                 geom.xyz
method                      rhf
run                         energy

threall                     1.0e-14
convthre                    1.0e-8
precision                   double

charge                      0
spinmult                    1

cis                         yes
cisnumstates                8
cismax                      50
cismaxiter                  100
cisguessvecs                20

ccbox                       yes
ccbox_ccsd                  yes
ccbox_scratch_dir           ./scr.geom
ccbox_cd_thresh             1.0e-8
ccbox_ccsd_r_convthre       1.0e-8

ccbox_eomccsd_states        4
ccbox_eomccsd_maxiter       100
ccbox_eomccsd_r_convthre    1.0e-8
ccbox_eomccsd_guessvecs     8
ccbox_eomccsd_maxsubspace   28
end
```

In this example, 4 EOM-CCSD singlet states are requested (triplet states are not available). Setting `ccbox_eomccsd_states` to a positive, nonzero value indicates that an EOM-CCSD calculation should be performed. As a guess, 8 CIS states are calculated and used to initialize the Davidson diagonalization. In lieu of a CIS calculation, singly and doubly excited CSFs can be specified through the keywords `ccbox_eomccsd_guess_singles` and `ccbox_eomccsd_guess_doubles`.

Calculation of one-electron properties – currently, the ground state dipole moment and transition dipole moments – can be requested by setting `ccbox_properties` to `yes`. If transition properties are calculated, natural transition orbitals will be output if `ccbox_ntos` is set to `yes`.

5 Trajectory visualization

To create a simple MD trajectory, you can use the example in `instdir/tests/caffeine`. The configuration file `start.md` is:

start.md

```
# basis set
basis                   6-31g
# coordinates file
coordinates             caffeine.xyz
# molecule charge
charge          0
# SCF method (rhf/blyp/b3lyp/etc...): DFT-BLYP
method          blyp
# type of the job (energy/gradient/md/minimize/ts): MD
run                             md
# number of MD steps
nstep                   10
# dump orbitals every MD step
orbitalswrtfrq  1
end
```

The only difference between this file and the file used for the single point calculations are the `nstep` and `orbitalswrtfrq` parameters, which are now set to 10 and 1, respectively. The keyword `orbitalswrtfrq` ensures that the molecular orbitals will be written at every MD step (out of 10, total).

The TeraChem output coordinates file format is compatible with VMD.[27] To open the trajectory file, go to VMD → File → New Molecule. Browse to the output `coors.xyz` file and make sure that the file type is correctly determined. Otherwise, select XYZ in the [Determine file type] menu. After the trajectory is loaded, you can adjust the representation settings so that the final molecule looks as shown in Figure 2.

The molecular orbitals can be visualized with VMD in similar fashion. Again, go to VMD → File → New Molecule. Browse to the output `caffeine.molden.1` file and make sure that the determined file type is Molden. Otherwise, select Molden in the [Determine file type] menu. To display both positive and negative isosurfaces you will need to create two representations for each orbital (one for the positive isosurface and one for the negative isosurface). After adjusting the atomic representation, create a new one and select "orbital" in the [Drawing Method] menu. All MO's will be listed in the [Orbital] dropdown along with the orbital energy. The [Isovalue] scroll bar controls the isosurface value (in a.u.). It is also possible to visualize the geometry and orbitals in the `caffeine.molden.n` files using the MolDen program.[28]

[27] http://www.ks.uiuc.edu/Research/vmd
[28] http://www.cmbi.ru.nl/molden

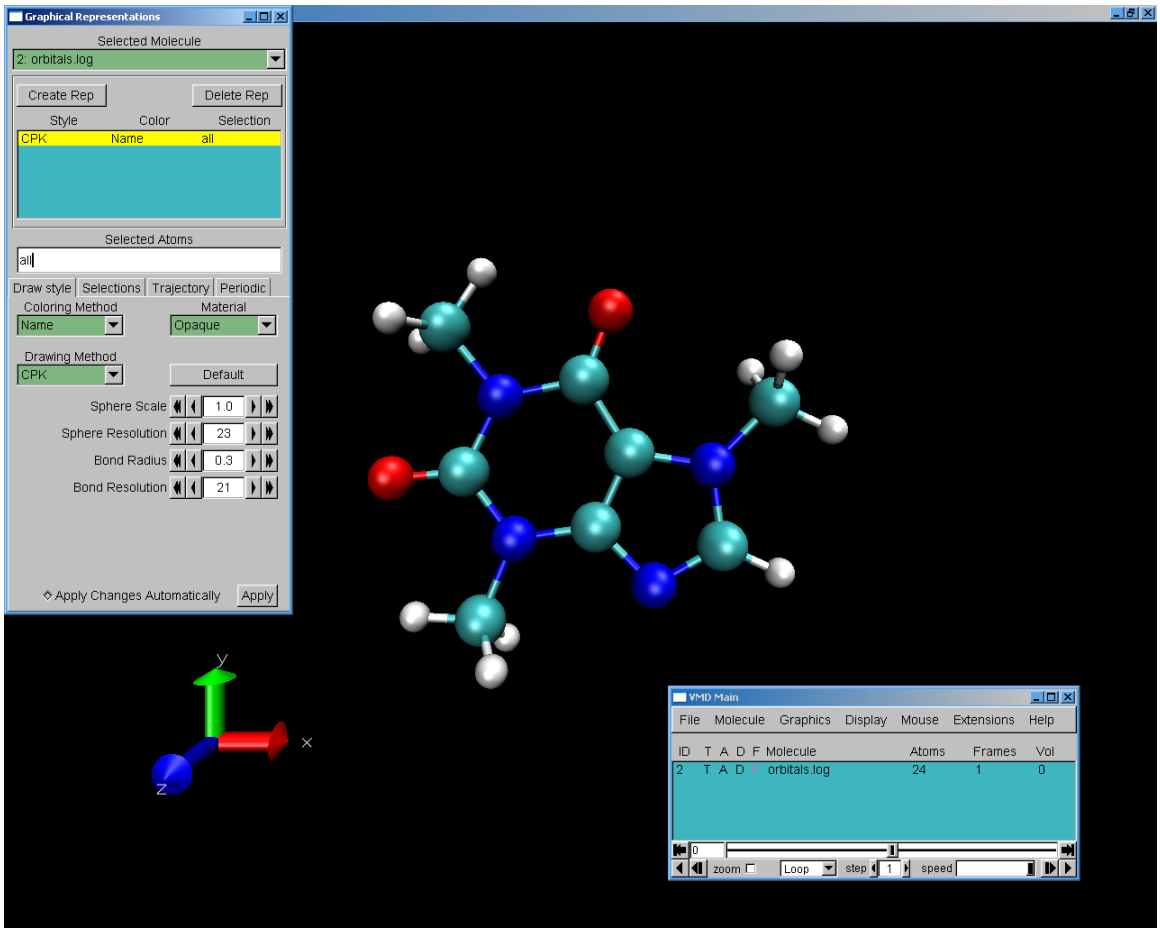

Figure 2: Caffeine molecule geometry output (`coors.xyz`) in VMD.

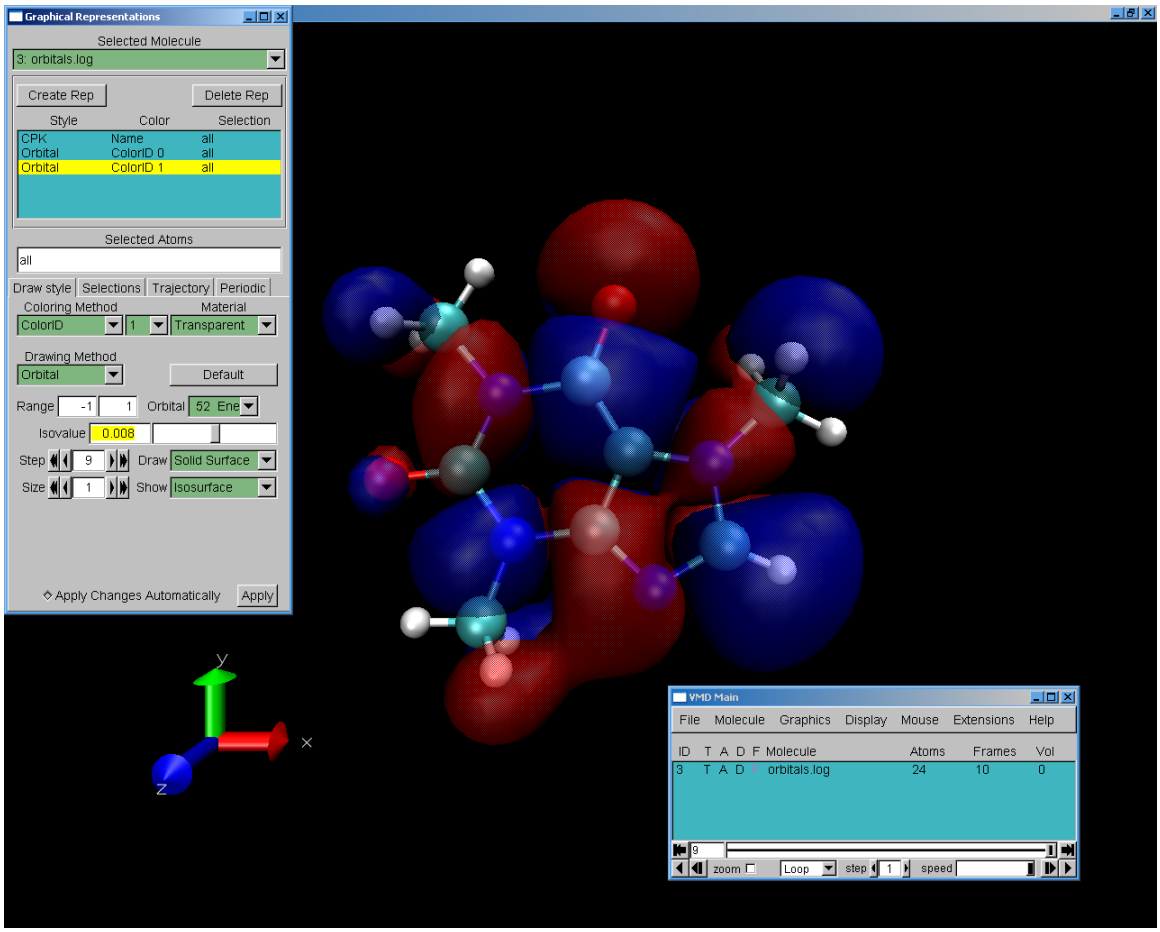

Figure 3: Caffeine molecule orbitals output (`orbitals.log`) in VMD.

6 Interactive calculations

Interactive calculations are especially suitable for remote jobs when TeraChem is running on a remote machine (cluster) and the trajectory visualization is performed on a local desktop. TeraChem can visualize the geometry optimization, TS search, and molecular dynamics trajectories in real time, i.e. interactively as the calculations run. These interactive molecular dynamics (IMD) runs are practical for molecules with up to approximately 50 atoms using a hardware solution with eight GPUs.[29] Future versions will also allow for user manipulation of the molecule, i.e. imposing external forces on atoms. In the `instdir/tests/benzene` directory, you will find the files needed to try IMD for benzene. The configuration file `start.imd` reads:

start.imd

```
# basis set
basis           sto-3g
# coordinates file
coordinates     C6H6.pdb
# molecule charge
charge          0
# SCF method (rhf/blyp/b3lyp/etc...): Restricted Hartree-Fock
method          rhf
# type of the job (energy/gradient/md/minimize/ts): MD
run             md
# initial temperature in K
tinit           1000
# this triggers interactive molecular dynamics
# imdport specifies the port VMD should connect to
imdport         54321
# Amount of time in milliseconds which VMD assumes each MD
# step will take.  If an MD step is completed faster than this,
# TeraChem will wait to provide the new coordinates. This is
# in order to ensure smooth display, even when the SCF takes
# a variable amount of time to complete.
imdtime         100
# number of MD steps
nstep           1000
end
```

Open two terminal windows. In the first window, launch VMD and load in the coordinates of the benzene molecule from C6H6.pdb In the second terminal window, launch TeraChem, which will initialize the simulation and pause for connection to be made with VMD. Now, in VMD, select Extensions → Simulation → IMD Connect (NAMD). Type `localhost` (or the IP address of the remote machine on which TeraChem is currently running) in the `Hostname` field and `54321` (the port specified by the imdport keyword in the TeraChem input file) in the `Port` field. Click the `Connect` radio button and you should see TeraChem executing in its window while the benzene molecule vibrates in the VMD display window.

[29] Luehr, N.; Jin, A. G. B.; Martínez, T. J., *J. Chem. Theo. Comp.* **2015**, *11*, 4536-4544.

7 Single Point Engine Mode through MPI Interface

When executed with command line option `--UseMPI=3`, TeraChem enters single point engine mode which can be used to perform single point energy or gradient calculations. MPI interface is used to establish server/client communications through paired `MPI_Comm_accept`/`MPI_Comm_connect` calls. The client should use the MPI_Comm_connect() method after the server process is started. Lookup service name specified by `--MPIPort` parameter to connect to the correct port. Once connected TeraChem expect to receive the initial set of settings in the form of a text buffer, identical with the format of TeraChem input file. For consistency with other MPI interfaces, each keyword and value in the input file must be padded to length of 128 characters, and the file must be terminated with the line **end**, with all lowercase and no indentation. Once the settings are processed, TeraChem will enter standby mode and respond to requests which can be any one of the three following options.

All requests starts with an array of 8 C integers and the tag identifies the type of the request.

- **Request single point calculation** (`tag=1`) The 8 integers in the transmitted array contains the following information:

  1. Number of atoms
  2. Molecular charge
  3. Spin multiplicity
  4. The wavefunction is closed shell if non-zero
  5. The wavefunction is restricted if non-zero
  6. Gradient will be calculated and reported if non-zero. Otherwise, only energy will be calculated.
  7. Meyer bond order matrix will be calculated and reported if non-zero.
  8. MO coefficients from the last run will be used as starting guess if set to 0. Guess will be regenerated with the specified method if set to 1. If set to 2, MO coefficients will be transmitted through MPI. If the previous MO coefficients or data received from MPI is inconsistent with the current molecule (for example, different molecules or basis set), TeraChem will fall back to regenerate guess orbitals with a GenerateGuess() call.

  A character buffer of size 2*natoms should then be passed to TeraChem through MPI, which should contain the atom symbols of each atom, each padded to width of 2 characters. A double precision floating point array of size 3*natoms which contains the Cartesian coordinates should then be sent.

  After all the information is received, TeraChem will perform single point calculation, and send one single double precision number which contains the energy. If the calculation is not successful (SCF did not converge), this message will have a non-zero tag and no other results will be transmitted. Otherwise, the tag should be zero and TeraChem will send the following results in order:

  - Atomic charge. Double * natoms
  - Spin density. Double * natoms
  - Dipole moments. Double * 4, first 3 components contains the vector and the 4th component the norm.
  - If gradient calculation is requested, a 3*natoms double array which contains gradient vector
  - If bond order is requested, a natoms*natoms double matrix which contains the Meyer bond order between each pair of atoms.

- **Changing settings** (`tag=2`) TeraChem expects to receive character buffer which contains the new TeraChem input file, with the same format as the initial settings buffer. Note that no memory of previous settings will be retained, and any keywords that are not mentioned in the new buffer will be set to their default values. This works the same way as performing a standalone TeraChem calculation with the content of this string buffer as the input file. TeraChem does not send any response to this request.
- **Disconnecting** (`tag=0`) The MPI connection will be terminated and the server will shutdown. **NOTE:** After sending the termination signal, the client must also call `MPI_Comm_disconnect()` for the server to finalize MPI and successfully shutdown.

8 TeraChem job parameters

Available job parameters are listed in the following Table. Note the use of the character "|" which should be interpreted as "or". For example, `x|y` means that *one* of `x` or `y` should be entered.

Parameter	*Description*	*Default value*
General parameters		
`jobname`	Name of job - used for naming output files. If this is not specified, the program will attempt to deduce a reasonable value from the basename of the coordinates file. For example, if the coordinates file is named `caffeine.xyz`, then `jobname` will be set to `caffeine`	`See description`
`scrdir`	Scratch directory	`./scr`
`keep_scr`	Keeps old scr directories, from a previous execution, when needed `yes|no`	`no`
`genscrdata`	Controls if the data in the scratch folder, such as the molden and c0 files, will be generated `yes|no` or `true|false`	`true`
`save_sao`	Controls if the atomic orbital overlap matrix will be generated and saved in $scrdir/sao `yes|no` or `true|false`	`false`
`run`	Sets calculation type `energy` – Single point energy `gradient` – Energy and gradient `project` – Wavefunction projection for better initial guess `minimize` – Optimize geometry `ts` – Transition state search `md` – Born-Oppenheimer Molecular Dynamics `initcond` – Generate initial conditions for MD `plumed` – MetaDynamics with PLUMED `numgradient` – Numerical gradients `frequencies` – Numerical frequencies `tdci` – Time-dependent CI `exciton` – Exciton model `ci_vec_overlap` – CI vector overlaps `ciopt` – Conical intersection search	`energy`

`gpus`	Number of GPUs to use in parallel. Sometimes one needs to change the default order of the devices. For example when device zero is used solely for display purposes. In such cases, the following line specifies which GPUs should be used (the enumeration starts from 0):
	`gpus 3 1 3 2`
	or
	`gpus 3`
	to use the default device order `0 1 2`
		`all`

`coordinates`	Name of file containing atomic coordinates	not set
`qmmm`	Name of file containing MM water coordinates. The presence of this keyword triggers a QM/MM calculation	not set
`water`	In QM/MM calculation, type of water molecules `tip3p|spc|spc/e|tip4p`	`tip3p`
`rigidbonds`	Rigid O-H bonds for MM water in QM/MM? `yes|no` or `true|false`	`true`

`pointcharges`	Add point charges from file?
	`filename|no`
	If point charges are to be used, the value of this keyword is the file name of the file containing the point charge locations and charges. Format of file is:
	`q1 x1 y1 z1`
	`q2 x2 y2 z2`
	`...`
	with charges in atomic units and coordinates in Angstroms
		`no`

`pointcharges_self_interaction`	Include self interaction of MM point charges in energy and forces? `yes|no` or `true|false`	`true`

`basis`	All available basis sets are located in the `instdir/basis` directory. In addition, users can construct customized basis sets. Here are examples of correct entries:
	`sto-3g, 6-31++G**,`
	`6-311++G(3df,3pd),aug-cc-pvdz,`
	`mini(s), midi!`
	When the string is parsed, all '`*`' symbols are replaced by '`s`' and all parenthesis are replaced by brackets to comply with Unix file naming rules. Thus, 6-31G** and 6-31Gss are equivalent.
		not set

auxbasis	Auxiliary basis set. If specified, the two-electron repulsion integrals will be calculated with density fitting approximation. N.B., The current version needs future revision to improve its performance.	not set
sphericalbasis	Spherical basis representation: yes|no. If witched off, Cartesian representation is used.	no
max_l_in_basis	Specify the highest angular momentum to be considered in the basis set: specify 0|1|2|... corresponding to s,p,d,.. angular momenta. The purpose is to neglect high angular momentum polarization functions.	not set
projectbasis	In project jobs, the converged wave function is projected onto projectbasis and stored in scr/prjct file (scr/prjcta scr/prjctb in unrestricted methods).	not set
charge	The total charge of the molecule (integer). May instead specify protein to determine protein charge based on protonation states and information in PDB file. charge protein requires use of standard amino acid 3-letter codes.	not set
spinmult	Spin multiplicity, 2S+1, of wavefunction (integer).	1
method	Restricted, unrestricted, and restricted open shell HF and KS. Restricted wavefunction: rhf | svwn | bhandhlyp | blyp | b3lyp |b3lyp1 | b3lyp5 | b3p86 | b3pw91 | pbe | pw91 | revpbe | pbe0 | revpbe0 | wpbe | wpbeh | bop | mubop | camb3lyp | b97 | wb97 | wb97x | wb97xd3 b3lyp or b3lyp1: VWN1 correlation (B3LYP in Gaussian) b3lyp5: VWN5 correlation (B3LYP in GAMESS) Unrestricted calculations are invoked by adding 'u' prefix, i.e. uhf | ublyp | ub3lyp, etc. Restricted open shell calculations are invoked by adding 'ro' prefix, i.e. rohf | roblyp | rob3lyp, etc.	rhf

`hfx`	Parameter that specifies percentage of HF exchange used in global hybrid functionals (e.g. `hfx 0.10`)	can be used with `b3lyp`, `b3lyp1`, `b3lyp5`, `bhandhlyp`, `revpbe0`, `pbe0`, `b3p86`, and `b3pw91` functionals
`rc_w`	Range correction scaling parameter ω required by DFT functionals with a fraction of exact long range exchange operator	can be used with `wpbe`, `wpbeh`, `mubop`, `wb97`, `wb97x`, `wb97xd3`, `camb3lyp`, and `rcamb3lyp` functionals
`c_ex`	Long range exact exchange operator contribution	can be used with `wpbeh` functional
`libxc`	A comma-separated list of functional names from a library of exchange-correlation functionals, libxc, e.g. `libxc xc_gga_x_b88,xc_gga_c_lyp`. If specified, DFT calculation will be performed with libxc functional(s) and the `method` keyword will be ignored.	not set
`libxc_fractions`	A comma-separated list of weighting factors for each specified libxc functionals, assigned with the same order; assumed to be 1 if unspecified.	not set
`dftgrid`	Integer value within [0-5] range, inclusive. Larger numbers are denser grids (and hence provide more accurate results). Grid 0 contains ˜800 grid points/atom and grid 5 contains ˜80,000 points per atom. The default grid (type 1) contains about 3,000 points per atom.	`1`
`dynamicgrid`	This parameter enables use of dynamical DFT grids. When it is on, grid 0 is used to converge the wavefunction until the DIIS error reaches the `gridthre` value (default: 0.01). Grid `dftgrid` is then used to finally converge the wavefunction. (`yes|no`)	`yes`
`gridthre`	Threshold for switching dynamical DFT grids.	`0.01`
`threall`	Two-electron integral threshold (float). Two electron integrals less than this threshold are neglected.	10^{-11}

`xtol`	Basis set linear dependence threshold (float). When diffuse basis functions are used, `xtol` and `convthre` may need to be raised up to ∼`1.0e-3`	`1.0e-4`
`dispersion` `dftd` (alias)	Should empirical dispersion corrections be used? `yes` | `no` | `d3` | `d2` D2[a] and D3[b] are two different dispersion parameterizations developed by S. Grimme and coworkers.	`no`
`units`	Units used for coordinates (`angstrom`|`bohr`)	`angstrom`
`nbo`	NBO analysis (`no` | `yes` | `npa` | `full` | `advanced` | `$nbo`) `no` – no NBO analysis `yes`\|`npa` – Natural population analysis only `full` – Full NPA/NBO analysis `advanced`\|`$nbo` – Analysis with $NBO input TeraChem uses v6.0 of the NBO package. See www.chem.wisc.edu/~nbo6 for documentation.	`no`
`polarizability`	Calculate polarizability tensor? (`yes`|`no`) This only works for restricted wavefunctions.	`no`
`poptype`	Population analysis method `mulliken` | `vdd` | `vanalsenoy` | `camm_mulliken` | `none` Output will appear in `scr/charge_xxx.xls` where `xxx=mull`|`vdd`|`vanalsenoy` or in `scr/camm_mulliken` for cumulative atomic mulliken charges	`mulliken`
`chkfile`	Checkfile name. No checkfile created if not specified.	`name_of_the_file`
`print_post_tc`	Print wavefunction info to `post.tcfchk` `yes`|`no`	`no`
`memcheck`	Memory check `yes`|`no`	`yes`
`epotential`	Calculations of the electrostatic potential map? `yes`|`no`	`no`
`ep_coordinates`	File with gridpoints for evaluation of ESP	

[a] S. Grimme, J. Comp. Chem. **27**, 1787 (2006).
[b] S. Grimme, J. Antony, S. Ehrlich and H. Krieg, J. Chem. Phys. **132**, 154104 (2010)

ep_file	Output file with values of ESP	scr/epotential.dx
ep_gridpoints	Number of gridpoints along x,y,z nx ny nz	
ep_gridcenter	Center of cubic grid (Angstrom) x y z	
ep_griddims	Distance of grid (Angstrom) along x,y,z x y z	
edensity	Calculations of the electronic density map? yes|no Output in scr/density.dx	no
bond_order_mat	Print bond order matrix[a] as a matrix in scr/bond_order.mat?	no
bond_order_list	Print bond order matrix as a list in scr/bond_order.list?	no
bond_order_thresh	Only bond orders exceeding this threshold will be printed in the list	0.001
ml_prop	Should machine learning properties be generated? (yes|no) If yes, properties such as mulliken populations would be printed out for after each optimization cycle to scr/mullpop file. If no, mulliken populations would only be printed for the optimized structure to scr/mullpop file.	no

Hardware Parameters

safemode	Allow to pre-configure GPUs for basis sets with d-functions. Setting this parameter to 'no' bypasses pre-configuration step but allows to have more GPU memory (e.g. GeForce). Use at your own risk.	yes
reservegpumwords	Number of GPU MWords that TeraChem will never use	0
timebomb	Toggle kernel timer mode off|warn|on off – Do nothing (timelimit not enforced). warn – Print error message only on – Print error and terminate execution	warn
timebombinit	Set initial kernel timelimit. If not set, first SCF iteration runs without limit and its time is used to initialize the limit for subsequent thread_pool calls.	not set

[a] I. Mayer, *J. Comp. Chem.* **2007**, *28*, 204-221.

`gpumem`	GPU memory (in MWords) reserved for host and device buffers. 128MWords=1Gb	`128`
`timings`	Provide timings `verbose|yes|no`	`no`
`simd`	All cards run every K kernel `yes|no`	`yes`
`cudamemoryflag`	Merge cudaMallocs for optimization `yes|no`	`no`
Numerical Thresholds		
`precision`	`single|mixed|dynamic|double` By default, TeraChem uses a dynamic precision scheme where two-electron integrals larger than some threshold value are computed with double precision and the rest are computed with single precision. The threshold which determines the splitting between single precision and double precision evaluation of the integrals is determined automatically during the SCF calculation. The `mixed` precision scheme uses a static threshold to determine which integrals should be evaluated in double precision (those larger than `threspdp`, which can be set in `inputfile`). Users can also request to use full single or double precision depending on the job purposes. Note that accumulation of all quantities on GPU is *always* performed with double precision accuracy to avoid floating point summation errors.	`dynamic`

`threspdp`	Threshold used to split single and double precision work in mixed precision calculations (atomic units).	`0.001`
`threcl`	Coulomb integral threshold	`1.0e-11`
`pqthre`	PQ threshold	`1.0e-15`
`threoe`	1-e integral threshold	`1.0e-12`
`thregr`	Grad integral threshold	`1.0e-11`
`threex`	Exchange integral threshold	`1.0e-11`
`kguard`	Kguard	`1.0e-3`

SCF Guess

`guess`	`generate`|`path/to/the/WFfile`|`sad`|`hcore`|`sadlp`|`frag`
	`generate` means the initial WF guess is generated from scratch using maximum orbital overlap;[a] it can also be loaded from the WF file. The WF is dumped in the end of each calculation to the `scr/c0` file (`scr/ca scr/cb` in unrestricted methods). To load WF in unrestricted calculations, `guess` requires two parameters, for example:
	`guess scr/ca scr/cb`
	`hcore` – Core Hamiltonian guess
	`sad` – Superposition of atomic density (SAD) guess
	`sadlp` – SAD guess with UHF
	`frag` – Guess from densities of converged UHF calculations on fragments
	See other keywords below for `sad` and `frag` guesses
		`generate`

[a] J.-M. Langlois et al, J. Phys. Chem. **98**, 13498 (1994).

`guess sad path/to/file`	Optional file specifies net charge, spin multiplicity, and spin up/down (`a|b`) of atoms, in the order of the coordinate file. If no file is provided, atoms are assumed uncharged, spin up (if necessary) with default ground state spins from NIST (up to $Z = 56$). For H_2 with two uncharged atoms:
	`0 2 a`
	`0 2 b`
		`No default`

`guess frag path/to/file`	File specifies number of fragments in the first line then: atoms per fragment, net charge, spin multiplicity, and spin up/down (`a|b`), with atoms designated to fragments in the order of the coordinate file. File must be provided . For ethane split into two methyl groups:
	`2`
	`4 0 2 a`
	`4 0 2 b`
		`No default`

`sad_spherical`	SAD densities are spherically symmetrized. `yes|no` or `true|false`	`true`
`sad_fxnl | frag_fxnl`	Densities generated for SAD or fragments from HF or BLYP `hf|blyp`	`hf`
`sad_avgspin | frag_avgspin`	Average alpha and beta spins of SAD or fragment guess `yes|no` or `true|false`	`true`
`frag_guess`	Initial guess to each fragment `hcore|generate`	`generate`
`frag_dblprec`	Use double precision in fragment convergence `yes|no` or `true|false`	`false`
`frag_threshscale`	Scale the convergence threshold for fragments (float)	`1.0`
`frag_maxit`	Maximum number of SCF iterations for fragment calculations (integer)	`50`
`frag_minspin`	Disregard the spin multiplicity in the fragment input file, instead minimize multiplicities automatically for each fragment, based on NIST ground state spins. `yes|no` or `true|false`	`false`

SCF Convergence		
purify	Density matrix purification guess|scf|no. guess – purifies the initial guess before the SCF scf – use purification[a] instead of matrix diagonalization in SCF (only without fon)	guess
scf	diis – Use Pulay's DIIS[b] for SCF convergence diis+a – Hybrid DIIS/A-DIIS[c] scheme	diis
diismaxvecs	Maximum number of DIIS vectors for diis+a	10
fock	incremental – Force incremental Fock matrix formation conventional – Force conventional Fock matrix formation auto – Use conventional algorithm if diffuse basis functions are detected or incremental otherwise	auto
maxit	Maximum number of SCF iterations (integer).	100
convthre	WF convergence threshold (float), the largest component of the DIIS error vector.	3.0e-5
levelshift	Use level shifting? (yes|no)	no
levelshiftvala	If levelshift=yes, the shift value (in hartree) for the virtual orbitals.	No default
levelshiftvalb	If levelshift=yes, and this is a UHF/UKS or ROHF/ROKS calculation, the shift value (in hartree) for the beta-spin or open-shell orbitals, respectively.	No default
watcheindiis	Switch from DIIS to ADIIS if DIIS predicts energy increase for two consecutive iterations (yes|no)	no
fon	Use FOMO-SCF (thermal smearing)? (yes|no)	no
fon_method	Type of smearing (gaussian|fermi|constant)	gaussian
fon_temperature	kT for FON (in atomic units, max set to 1.0)	0.25

[a] A. M. N. Niklasson, Phys. Rev. B **66**, 155115 (2002).
[b] P. Pulay, J. Comp. Chem. **3**, 556 (1982).
[c] X. Hu and W. Yang, J. Chem. Phys. **132**, 054109 (2010).

`fon_print`	Print level for FON calculations `(0|1)` 1 gets more printing	`not set`
`fon_mix`	Alpha and beta electrons will be mixed in UHF calculations `yes|no` or `true|false`	`no`
`fon_guess`	Provide a file name to read FON occupation numbers.	`none`
`fon_anneal`	Anneal FOMO temperature from a higher value to a lower one. This is useful in both normal DFT (where the final temperature is 0), and FOMO-SCF when it does not converge. During failsafe mode, the temperature will be lowered to as close to the target temperature as possible. If the number of prescribed iterations is used up, it will be allowed to converge at a temperature between the target and initial temperature. The 'marzari' `fon_method` should be used for failsafe mode so that the FOMO free energy equals to energy at zero temperature perturbatively. (`yes|no|failsafe`)	`no`
`fon_anneal_iter`	Number of SCF iterations during which that the temperature annealing is completed. This should be smaller than the total number of SCF iterations.	`100`
`fon_target`	Target temperature for annealing. The default is to anneal to zero temperature.	`0`
`fon_converger`	FON will be used to accelerate convergence when this option is 'yes', and the SCF is considered finished once the DIIS procedure converges. The temperature will be raised and re-annealed if SCF fails to converge after `fon_anneal_iter` iterations. Otherwise, FON is used to explore SCF solution space, and each time after convergence, the density matrix and the associated energy will be saved, then the temperature will be raised again to look for more solutions. After a fixed number of attemps, set by 'fon_tests', the lowest energy solution will be picked as the final solution.	`yes`

fon_tests	Number of annealing attemps to try.	3
fon_steer	Magnitude of steering term in FON calculation. Only active when FON is used in exploration (instead of converger) mode, set by 'fon_converge' parameter. This adds an additional component to the DIIS vector, which is the total overlap of the current density matrix with any of the converged values from previous annealings. This term also scales with temperature, making it vanish when temperature is annealed to the target temperature. This is used to push the wave function away from previous values.	6
fon_coldstart	When this option is yes, the FON temperature will start directly at the target temperature. It will only be raised to the high temperature and annealed back again if SCF fails to converge at the low temperature. This usually result in faster convergence in systems that are normally easy to converge and only occasionally experience difficulties that requires annealing.	The value of fon_converger
closed	Number of doubly occupied orbitals for FOMO	none
active	Number of orbitals with noninteger occupation for FOMO	none

Geometry Optimization and Transition State Search - DL-FIND

new_minimizer	Use the geomeTRIC minimizer (yes) or DL-FIND (no)?[a] (no|yes).	no
min_print	How much output is desired (something|verbose|debug)	verbose
nstep	Maximum number of optimization/TS search steps	100
min_tolerance	Termination criterion based on the maximum energy gradient component	$4.5*10^{-4}$
min_tolerance_e	Termination criterion based on the SCF energy change	10^{-6}
min_coordinates	Type of coordinates in which optimization/TS search is performed (cartesian | dlc | dlc_tc | hdlc|hdlc_tc)	dlc

[a] Wang, L.-P.; Song, C.C. (2016), *J. Chem. Phys.* **2016** 144, 214108.

min_method	Optimization/TS search method (sd | cg1 | cg2 | lbfgs) sd – steepest descent cg1 and cg2 – conjugate gradient lbfgs – L-BFGS bfgs – BFGS, requires initial Hessian set according to min_init_hess	lbfgs
gp_c3, gp_c4	Weighting parameters given to gradient terms in the gradient-projection method for conical intersection searches [a]	1.0, 0.9
mdci_coordinates	Reference coordinate file of the geometry from which to minimize the distance for an MDCI	Value of coordinates
min_hess_update	Hessian update algorithm (never | powell | bofill | bfgs) If never, the Hessian is recalculated using finite differences at each step	bfgs
min_init_hess	Initial Hessian (used only when min_method=BFGS) (fischer-almlof | one-point | two-point | diagonal | identity) one-/two-point – exact Hessian from finite differences diagonal – only diagonal elements are calculated using final differences, off-diagonal elements equal zero identity – initial Hessian is an identity matrix fischer-almlof – chemically-motivated guess Hessian, usually the best choice	fischer-almlof
min_delta	Atomic displacement in finite difference calculations	0.003
min_max_step	Maximum step size in internal coordinates	0.5
min_restart	Whether the optimization/TS search job is started from scratch (no) or loaded from the checkpoint files (yes)	no
ts_method	Transition state search method (neb_free | neb_restricted | neb_frozen | neb_free_cart | neb_restricted_cart| neb_frozen_cart) neb_free – Nudged Elastic Band (NEB) with free endpoints neb_restricted – NEB with endpoints allowed to move perpendicularly to their tangent direction neb_frozen – NEB with frozen endpoints neb_x_cart – only initialization is performed in min_coordinates coordinates, while the TS search is done in Cartesians	neb_free

[a] T. W. Keal, A. Koslowski, and W. Thiel, Theor. Chem. Acc. **118**, 837 (2007)

min_image	Number of NEB images in the TS search calculations. Should be greater than one. The images are listed in the input coordinates file (specified by coordinates). If the number of images found is smaller than min_image, the program will automatically generate missing images by interpolation. At least two images (endpoints) should be listed in the coordinates file. The last one (i.e. the min_imageth) is the climbing image.	10
min_nebk	Minimum/standard spring constant between NEB images.	0.01
max_nebk	Maximum spring constant between NEB images for variable springs. If max_nebk $\leq$ min_nebk, standard approach is employed.	0.01
min_maxstephess	Update Hessian every min_maxstephess steps.	50
min_dump	Optimization checkpoint file will be dumped every min_dump cycles.	10
min_scale_step	Each step will be scaled by min_scale_step.	1.0
num_deriv_formula	Formula applied for the numerical derivatives: central|forward|backward|	central
num_deriv_points	Number of points for numerical derivatives.	3
num_deriv_displacement	Value for the displacement in numerical derivatives.	0.01
lbfgs_mem	Number of steps kept in the memory of the L-BFGS minimizer.	#dof
lbfgs_reset	Number of steps after which to reset the L-BFGS minimizer.	Never

Geometry Optimization - geomeTRIC

new_minimizer	Use geomeTRIC? (yes|no)	no
min_coordinates	Coordinates for minimization (prim|dlc|hdlc|tric|cart)	tric
min_maxiter	Maximum number of optimization/TS search steps	500
min_converge_gmax	Termination criterion based on the maximum energy gradient component	4.5*10^{-4}

min_converge_grms	Termination criterion based on the root mean square of the energy gradient vector	3.0*10^{-4}
min_converge_dmax	Termination criterion based on the maximum component of the atom displacement vector.	1.8*10^{-3}
min_converge_drms	Termination criterion based on the root mean square of the atom displacement vector.	1.2*10^{-3}
min_converge_e	Termination criterion based on the SCF energy change	10^{-6}
check_coor_freq	Rebuild frequency for internal coordinates	1000

Initial Condition Generation

nderivpoints	Number of points in numerical derivative (3|5|7)	3
displacement	Step size in finite difference (bohr)	0.005
initcondtemp	Temperature (K) for initial conditions	0
husimi	Sample from Husimi distribution? (true|false) Default is to sample from Wigner distribution	false
maxgradnorm	Calculation will stop if maximum component of gradient is larger than this value. Initial condition code assumes starting geometry is a local minimum or transition state. Override this check with mincheck, see below.	0.0001
mincheck	Check that initial geometry is a stationary point, i.e. norm of gradient is less than maxgrad (true|false). Default is to check	true
initcondtssign	Used to force displacement along imaginary modes to be in specific direction (-1|0|1). One needs in general to run with both signs and then determine which is desired (i.e., which goes back to reactants and which goes on to products). Setting this to zero (the default) will not displace along imaginary modes at all.	0

cross_sections	Calculation of Raman intensities (no) or Raman cross-sections (yes) or calculation of only the temperature-independent part of the cross-sections (ti). (yes|no|ti)	no
thermo_temp	Temperature for evaluation of vibrational analysis in Kelvin	300

Molecular Dynamics Parameters

nstep	Total number of MD steps. Set nstep to 0 for single-point energy calculations (integer).	10^6
integrator	The way the initial WF guess is generated in subsequent MD iterations. In the first several iterations the initial guess is constructed from scratch. (reversible_d|reversible|regular|reset) reversible_d: time-reversible integrator with dissipation[a] reversible: time-reversible integrator without dissipation[b] regular: initial WF guess is taken from converged WF at previous MD step reset: initial WF guess is generated from scratch at each MD step (Work also with scfintegrator)	reversible_d
cisintegrator	Same as integrator, but for excited state.	
zintegrator	Same as integrator, but for Z-vec.	
md_binaryoutput	Binary output for MD. (yes|no)	no
md_outputfreq	Frequency for MD output?	1
seed	Seed for random number generator. Set this to a different integer for different MD runs (or set it to a known seed in order to reproduce an earlier run).	1351351
timestep	MD integration time step in femtoseconds (float)	1.0
thermostat	Temperature control – velocity rescaling, Langevin dynamics, or Bussi-Parrinello Langevin dynamics (rescale, langevin, or bp)	rescale

[a] A.M.N. Niklasson et al, J. Chem. Phys. **130**, 214109 (2009).
[b] A.M.N. Niklasson et al, Phys. Rev. Lett. **97**, 123001 (2006).

`rescalefreq`	When velocity rescaling is used, determines how often the velocities are rescaled. For instance, setting `rescale` to `1000` will force rescaling at every 1000^{th} MD step. To obtain NVE dynamics, set `rescalefreq` to a value larger than `nstep`.	$2{\cdot}10^9$
`tinit`	Initial temperature (K) sampled from Boltzmann distribution of velocities at T = `tinit` (float)	`300.0`
`t0`	Thermostat temperature (K) (float)	`300.0`
`lnvtime`	The Langevin damping time (fs), only used when thermostat is set to `langevin` or `bp`.	`1000.0`
`molden`	Path to an output file containing the canonical molecular orbitals in Molden format	`molden.molf`
`orbitalswrtfrq`	Determines how often the orbitals are written to the output file. Due to large size (sometimes the orbitals require 100MB and even more of disk space) it does not make sense to write orbitals at every MD iteration. (integer)	$2{\cdot}10^9$
`velocities`	Filename containing initial velocities (or `random` to specify that initial velocities should be drawn from Boltzmann distribution)	`random`
`initialtime`	Initial time for simulation.	`0.0`
`restartmd`	Path to the MD restart binary file. If set, the data in this file (rather than coordinates, velocities, temperature, etc) is used to start MD trajectory.	`not set`
`restartmdfreq`	Specifies how often the restart data is dumped to `scr/restart.md`	`100`
`mdbc`	`spherical` – Enables spherical boundary conditions `spherical+rf` – Onsager reaction field enabled	`not set`
`epsilon`	Solvent dielectric constant, if `spherical+rf` or for `pcm`	`80.0`
`md_density`	If set, R_1 is automatically adjusted to the specified density in g/mL	`1.0`
`md_r1`, `md_r2`	R_1 and R_2 parameters for spherical boundary conditions in Å	`0.0`, `0.0`
`md_k1`, `md_k2`	k_1 and k_2 parameters for spherical boundary conditions in kcal/(mol Å^2)	`10.0`, `0.0`

mdbc_hydrogen	Inclusion of the hydrogen atoms in the spherical boundary condition. yes|no	no
mdbc_mass_scaled	Mass weighting in the spherical boundary condition yes|no	no
mdbc_no_recentering	Fix the origin of the spherical boundary to (0,0,0)? If no, the origin follows the total system's center of mass. yes|no	no
mdbc_t1 mdbc_t2	For nanoreactor simulations. Spherical boundary condition md_r1 will be operative for mdbc_t1 time steps, followed by spherical boundary condition md_r2 for mdbc_t2 time steps and then the cycle will repeat.	not set
imdport	Invokes interactive molecular dynamics by specifying the corresponding VMD port	not set
imdtime	Number of milliseconds for each MD step. Notice this is a lower bound, i.e. if the SCF takes longer than this, the display will appear to freeze. But if SCF takes less time than this, the MD will wait before starting the next step	not set
imdsubsteps	Number of substeps per MD step for integrating the haptic force	imdtime/50 + 1
imddrawfreq	Frames per second drawing rate for "display system"	30
steering	Use Steering? adaptive|name_of_ file|no If adaptive: need steeratom1, steeratom2, and steerforce If name_of_file: Steering switched on and will use name_of_file If no file name: Steering switched on from file steering	no

CIS, RPA, TDDFT, TDDFT/TDA

cis	Request a CIS (HF) or a TDDFT/TDA (DFT) calculation (yes or no)	no
rpa	Request a RPA (HF) or a TDDFT (DFT) calculation (yes or no)	no
cisnumstates	Number of excited states	1
cismult	Multiplicity of the excited states	Same as ground state

cistarget	Target state for properties or gradient calculation	0
cisalgorithm	Algorithm for diagonalization davidson|diis|debug	davidson
ciss2	Compute $< S^2 >$ true or false	true
cistransdipole	Compute transition dipole moments (yes or no)	yes
cistransdipolederiv	Compute transition dipole moments derivatives (yes or no)	no
cisunrelaxdipole	Compute unrelaxed dipole moments (yes or no)	yes
cisrelaxdipole	Compute relaxed dipole moments (yes or no)	no
cisdipolederiv	Compute dipole moments derivatives (yes or no)	no
cischarges	Compute excited state charges (yes or no)	no
cisprintcivec	Print CI vectors (yes or no)	yes
cisprintthresh	Threshold to print contribution to CI vector (threshold compared with the absolute value of the CI coefficients)	0.1
cisnos	Print natural orbitals (yes or no)	no
cisattachdetach	Print attachement/detachement densities (yes or no)	no
cisntos	Print natural transition orbitals (yes or no)	no
cistransdensity	Print transition density (yes or no)	no
cisdiffdensity	Print difference of densities (yes or no)	no
cisguessvecs	Number of guess vectors	=cisnumstates
cissubspace	Number of vectors to store	=cisnumstates
cismaxiter	Maximum number of Davidson iterations	20
cismax	Maximum size of the subspace	=cisnumstates*10
ciscollapse	Collapse vectors per root	2
cisconvtol	Convergence tolerance on the residue	1.0e-4
cisspdpconvtol	Convergence tolerance before switching from single to double precision	1.0e-4
cisstol	Orthogonality tolerance	1.0e-12
cisuserguess	Name of user guess file for CIS. Guess file contains a list of guess vectors in the format 'label occ virt coeff', where 'occ' and 'virt' are orbital numbers specifiying the excitation, and 'coeff' is the CI coefficient. 'label' indicates the type of excitation: for restricted calculations this should be left blank; for unrestricted calculations it should be 'alpha' or 'beta'; for RPA calculations it should be 'x' (excitation) or 'y' (deexcitation). Each guess should be preceded by 'vector' and closed with 'end'.	not set

cisrestart	Name of a restart file for CIS	not set
cpcisalgorithm	CP-CIS algorithm diis|inc_diis	diis
cpcisiter	Maximum number of iterations for CP-CIS	50
cpcistol	CP-CIS convergence threshold	1.0e-4

hole-hole Tamm-Dancoff approximation (hh-TDA)

hhtda	Do an hh-TDA calculation (yes | no). This starts from a $(N+2)$-electron reference, and annihilates two electrons to obtain the N-electron ground and excited states. Two orbital generation models are implemented, N+2 and fractionally occupied electron model. [a] [b] In the current implementation, this has some implications on the input structure as seen below (see orbital choice-related keywords): The hh-TDA method uses the same technical settings as for restricted CIS/TDA-DFT. Specific settings are given below. Only restricted wavefunctions are currently implemented.	no
hhtdasinglets	Number of singlet states to be computed. Note, that the ground state is part of this set.	0
hhtdatriplets	Number of triplet states to be computed.	0
cisnumstates	Needs to be set to the sum of hhtdasinglets and hhtdatriplets.	0
cistarget	Target state for gradient calculation	0
cismult	Target state multiplicity	1
hhtdaexactexchange	Use functional-independent, Hartree-Fock-like response kernel suggested by W. Yang and coworkers (yes | no). By default, the functional-dependent linear response-type kernel is used.	no
nacstate1	State 1 in nonadiabatic coupling calculations.	_
nacstate2	State 2 in nonadiabatic coupling calculations.	_

[a] C. Bannwarth, J. K. Yu, E. G. Hohenstein, T. J. Martínez, J. Chem. Phys., (2020), accepted; ChemRxiv DOI: 10.26434/chemrxiv.11828256.

[b] J. K. Yu, C. Bannwarth, E. G. Hohenstein, T. J. Martínez, in preparation.

	orbital choice-related keywords	
`charge`	If `fon false` (see below), the charge is the actual charge of the system reduced by 2. This generates orbitals for the (N+2)-electron system. The final states obtained with hh-TDA will thus correspond to a charge of `charge + 2`. If `fon true`, then the `charge` keyword defines the number of electrons in the SCF part only, where `charge` must be smaller than twice the number of `active` orbitals (restricted case). The final charge of the system will be defined via the `closed` and `active` keywords in that case (see below). A typical setting is described at the very bottom of this block.	–
`fon`	Enable fractional orbital occupation in the SCF (`yes|no`). This requires specification of a `fon_method` and an active space. The total charge of the final system is then $\sum_A Z_A - 2 \cdot (n_{\text{active}} + n_{\text{closed}})$. As written above: in the fon mode, the `charge` value then solely refers to the charge used in SCF.	
`closed`	Number of closed orbitals.	`0`
`active`	Number of active orbitals. This defines the number of populated electrons from which the double annihilation in hh-TDA takes place. Set `closed`+`active` such that – if multiplied by two and summed with the nuclear charge – the net charge of the system is obtained.	`0`
	A recommended setting based on benchmark data (for vertical excitation energies) is:	

	`method bhandhlyp`
	`charge 0`
	`hhtda yes`
	`fon yes`
	`fon_method constant`
	`closed 0`
	`active` $N_{\text{el}}/2 + 1$
	Here N_{el} is the number of electrons in the system. Combine with remaining settings. A sample input is found in	

CPSCF

cphfdiismin	DIIS min vectors for CPSCF equations.	1
cphfdiismax	DIIS max vectors for CPSCF equations.	10
cphfdiisstart	DIIS start for CPSCF equations.	0
cphfiter	Maximum number of iterations for CPSCF equations.	20
cphftol	Convergence threshold for CPSCF equations.	1.0e-4
cphfalgorithm	Solver for CPSCF equations diis|inc_diis|cg|precon_cg|jacobi|inc_jacobi	diis

PCM/COSMO

pcm	Polarizable Continuum Model?cosmo|xppcm cosmo - C-PCM solvent model xppcm - XP-PCM extreme pressure model[a b]	no pcm
epsilon	Solvent dielectric constant. (default=bulk water)	80.0
pcm_scale	PCM charge scaling 0|1 0 - Charge scaling factor f=$(\epsilon-1)/\epsilon$ as suggested by G-COSMO[c] and C-PCM[d] 1 - Charge scaling factor f=$(\epsilon-1)/(\epsilon+0.5)$ as suggested by original COSMO.[e]	0
solvent_radius	The radius of the solvent molecule in Angstroms.	0.0
pcm_grid	Type of pcm sphere grid polyhedron|lebedev|iswig|swig polyhedron - original COSMO cavity by Klamt. lebedev - Lebdev grid with point charge on each point iswig - Switch Gaussian by Lange & Herbert swig - Switch Gaussian grid by York & Karplus sphere - Single fixed Spherical cavity for mdbc=1	iswig

[a] R. Cammi, V. Verdolino, B. Mennucci, and J. Tomasi *Chem. Phys.* **2008**, *344*, 135
[b] A. Gale, E. Hruska, F. Liu *J. Chem. Phys.* **2021**, *154*, 244103
[c] Truong, T. N.; Stefanovich, E. V., *Chem. Phys. Lett.* **1995**, 240,253-260.
[d] Barone, V.; Cossi, M., J. Phys. Chem. A **1998**, *102*, 1995-2001
[e] Klamt, A.; Schuurmann, G.,*J. Chem. Soc. Perkins. Trans. 2* **1993**, 799-805.

nspa	Gathered tesselation order (NSPA)0|1|2|3|4|5 Only used when pcm_grid=polyhedron, must have nspa ≤nppa	2
nppa	Basic tesselation order (NPPA) 0|1|2|3|4|5 Only used when pcm_grid=polyhedron, must have nppa ≥ nppa	3
pcmgrid_h	Lebedev grid order ℓ for Hydrogen atoms. 5|7|11|17|23|29|35|41|47|53|59 5 - 14 points/atom 7 - 26 points/atom 11 - 50 points/atom 17 - 110 points/atom 23 - 194 points/atom 29 - 302 points/atom 35 - 434 points/atom 41 - 590 points/atom 47 - 770 points/atom 53 - 974 points/atom 59 - 1202 points/atom	17
pcmgrid_heavy	Lebedev grid order ℓ for Heavy atoms. 5|7|11|17|23|29|35|41|47|53|59 meaning of each option is the same as pcmgrid_h	17
pcmgrid_h_mm	Lebedev grid order ℓ for MM Hydrogen atoms. 5|7|11|17|23|29|35|41|47|53|59 meaning of each option is the same as pcmgrid_h	17
pcmgrid_heavy_mm	Lebedev grid order ℓ for MM Heavy atoms. 5|7|11|17|23|29|35|41|47|53|59 meaning of each option is the same as pcmgrid_h	17
pcmgrid_ptchrg	Lebedev grid order ℓ for point charges in point charge QM/MM calculation. 5|7|11|17|23|29|35|41|47|53|59 meaning of each option is the same as pcmgrid_h	17
pcm_radii	PCM radii method klamt|ff|read|bondi klamt - Radii recommended by original Klamt paper.[a] read - Read radii from a input file specified with pcm_radii_file bondi - Radii by bondi,[b][c] with new definition for hydrogen atoms.[d] Relevant only for pcm_grid≠sphere.	bondi

[a] Klamt, A.; Schuurmann, G.,*J. Chem. Soc. Perkins. Trans. 2* **1993**, 799-805.
[b] Bondi, A.,*J. Phys. Chem.* **1964**, *68*, 441-451
[c] Rowland, R. S.; Taylor, R.,*J. Phys. Chem.* 1996, *100*, 7384-7391
[d] Mantina, M.; Chamberlin, A. C.; Valero, R.; Cramer, C. J.; Truhlar, D. G.,*J. Phys. Chem. A* 2009, *113*, 5806-5812

`pcm_rad_scale`	Scaling factor for solute atom radius in cosmo.	`1.2`
`pcm_ptchrg_radius`	Universal atom radius for all point charges in QM/MM calculation in cosmo.	`1.5`
`solvent_radius`	Solvent probe radius in Å	`0.0`
`cosmodsc`	dsc (δ^{SC}) for the cavity of original COSMO method by Klamt. Relevant only if `pcm_grid=polyhedron`	`0.0`
`pcm_radii_file`	File containing PCM radii if `pcm_radii = read`. File format please refer to section 3.8.	`not specified`
`pcm_global_radius`	If `pcm_grid = sphere` and `mdbc`=1, radius of the single spherical cavity. Otherwise this option has no effect on cavity radii	`10`
`pcm_matrix`	Different ways of storing matrix **A** `full` The full matrix is stored. `packed` - Lower triangular of A is stored. `no` - No matrix is stored. The matrix-vector product Ax is built on the fly on GPUs with double precision. `no_sp` - The matrix -vector product Ax is built on the fly on GPUs with single precision.	`full`
`pcm_pair_driven`	Use primitive-pair driven algorithm for the evaluation of electrostatic potential on the grid points? `yes|no`	`no`
`pcm_threoe`	Screening threshold for 1-electron integrals related to PCM	10^{-12}
`dynamiccg`	Which mode of Dynamic CG[a] is used `0|1|2` `0` - No dynamic CG used. Use the same CG convergence threshold `cgtoltight` throughout all SCF iterations. `1` - Use the 2-δ switching scheme. `2` - Use the dynamic CG threshold based on empirical formula.	`2`
`cgtoltight`	If `dynamiccg`=0: dynamic CG is not used, CG is considered to be converged when $\|Ax-b\| <$ `cgtoltight` for the linear equation Ax=b. If `dynamiccg`=1: 2-δ switching scheme of dynamic CG is turned on, `cgtoltight` is the tight CG convergence threshold δ_2	10^{-6}

[a] Liu, F.; Luehr, N; Kulik J. K.; Martinez, T. J. DOI: 10.1021/acs.jctc.5b00370

cgtolscale	Loose CG convergence threshold scaling factor for dynamic CG 2-δ switching scheme δ_1=δ_2*cgtolscale	10000
dynamiccgtol	When dynamiccg=1, use 2-δ switching scheme to choose CG threshold current DIIS error < dynamiccgtol: start to use tight threshold δ_2 to converge CG	10^{-3}
cgprecond	CG preconditioner jacobi – Diagonal preconditioner blockjacobi – Random-block-jacobi. Inverse of the block diagonal stored If blockjacobi: set cgblocksize	jacobi
cgblocksize	If cgprecond=blockjacobi, CG blocksize.	100
print_ms	Print molecular surface coordinates sas.xyz in scratch folder 1=yes|0=no	0
ss_pcm_solvation	State-specific[a] solvation model for CIS-type calculation eq|neq|ground_neq eq – Equilibrium solvation neq – Nonequilibrium solvation ground_neq – Ground state nonequilibrium solvation All options work only if run=energy.	no ss_cis calculation set
sspcm_maxit	Maximum SSPCM iterations	20
sspcm_convthre	SSPCM Convergence threshold	10^{-5}
lr_pcm_solvation	Linear-response[a] solvation model for CIS-type calculation eq|neq eq – Equilibirum solvation neq – Nonequilibrium solvation Must have cis=yes and specify the Restricted CIS related parameters.	no lr_pcm calculation set
pcm_write	File name of the PCM field file to be written. If specified, PCM field of the final result will be written to this file.	no default value
pcm_read	File name of the PCM field file to be read. Can be used with state-specific nonequilibrium calculations. ss_pcm_solvation=neq|ground_neq	no default value

[a] Cammi, R.; Corni, S.; Mennucci, B.; Tomasi, J.,*J. Chem. Phys.* **2005**, *122*,104513

`fast_epsilon`	Fast dielectric constant for non-equilibrium solvation if `ss_pcm_solvation=neq|ground_neq` or `lr_pcm_solvation=neq` or `split_pcm_energy=1`	`2.0`
`split_pcm_energy`	Split ground state PCM solvation energy into fast and slow part and print out. Final energy is the same as regular ground state PCM. `0` - Do not split PCM energy `1` - Split PCM energy	`0`
`xppcm_nb`	For XP-PCM calculation only. Number of valence electrons for the solvent.	`0`
`xppcm_mb`	For XP-PCM calculation only. Molecular weight for the solvent.	`18.0`
`xppcm_rhob`	For XP-PCM calculation only. Number density for the solvent.	`0`

ESP/RESP

`resp`	Calculate ESP/RESP charge? `yes | no`	no
`eps_grid_scale`	Cavity radius of the first layer of Connolly surface (float, >1.0)	`1.4`
`esp_grid_incr`	Increment of radii scale between different layers of Connolly surface (float, >0). Radius of i^{th} layer: `eps_grid_scale`+i×`esp_grid_incr`*vdw_radii	`0.2`
`esp_grid_layers`	Number of Connolly surface layers (integer)	4
`probe_radius`	Radius of the rolling ball (probe) for Connnolly surface. Unit: Angstrom (float)	0
`esp_grid_dens`	Number of grid points/Å^2 (float) Recommendation: Should increase density until the resulting charge is converged	1.0
`esp_restraint_a`	Parameter a for the restraint $a\sum^k((q_k^2 + b^2)^{1/2} - b)$. Controls the strength of restraint . Unit: a.u. (float)	0.0005

`esp_restraint_b`	Parameter a for the restraint $a\sum^{k}((q_k^2 + b^2)^{1/2} - b)$. Controls the tightness near the bottom of the restraint . Unit: e^- (float)	0.1

Uniform Electric Field (UEF)

`finite_field`	Toggles the presence of a uniform electric field for electronic structure calculations `yes | no`	no
`efx`	Magnitude of the electric field in the lab frame x-direction in units of e	`0.0`
`efy`	Magnitude of the electric field in the lab frame y-direction in units of e	`0.0`
`efz`	Magnitude of the electric field in the lab frame z-direction in units of e	`0.0`

9 Contact information

We will be pleased to receive any feedback on your experience with TeraChem. Should you have any suggestions, concerns, or bug reports, please email them to help@petachem.com Also, please feel free to participate in the TeraChem User Forum at http://www.petachem.com/forum. TeraChem developers monitor this forum frequently and are happy to answer questions about the software.

10 Copyright

11 End user license agreement

IMPORTANT: PLEASE READ THIS END USER LICENSE AGREEMENT ("AGREEMENT") CAREFULLY BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE.

BY DOWNLOADING, INSTALLING, UPDATING, COPYING, OR USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, DO NOT DOWNLOAD, INSTALL, UPDATE, COPY, OR USE THE SOFTWARE.

1. Grant of License. Subject to the conditions and limitations set forth herein, PetaChem, LLC (PetaChem) hereby grants you a limited, nonexclusive, and nontransferable license to use in binary form one (1) copy of the Software and any updates, patches, enhancements, technical support, documentation, and any related materials provided by PetaChem (collectively, "Software"). You may install and use the Software on one computer with up to eight graphical processing unit cards (the "Licensed Computer"). The Licensed Computer may be a storage device, such as a network server, used only to install or access the Software from other computers over an internal network, provided that each separate computer which accesses or uses the Software is a Licensed Computer.

2. License Termination. This license and Agreement is effective until terminated. This Agreement will terminate with or without notice on any breach by you of this Agreement. You may terminate this Agreement at any time. Promptly upon termination, you will cease using and destroy all copies of the Software in your possession or under your control. Sections 3-9 of this Agreement will survive any termination of this Agreement.

3. Consent to Collection and Use of Data. You agree that PetaChem may collect technical data about the Licensed Computer and Software performance. PetaChem will not collect information about the identity or geometry of molecules being studied with Software. You further agree that PetaChem may use this information, so long as it is in a form that is not personally identifiable to you, to improve its Software or services to you.

4. License Limitations. The Software is licensed and not sold. All rights not expressly granted herein are reserved by PetaChem. Except as expressly authorized in this Agreement, you will not:

   (a) Sell, rent, lease, distribute, sublicense, assign, publish, or otherwise transfer (including by loan or gift) the Software, or any full or partial copies thereof;

   (b) Copy, modify, or create derivative works of the Software in any way;

   (c) Remove any copyright notice or licensing information contained in the Software;

   (d) Use the Software for the benefit of third parties or as part of its own commercially licensed products or services;

   (e) Disassemble or otherwise reverse compile or reverse engineer the Software; or

   (f) Install the Software on any computer which is not a Licensed Computer.

5. Copyright. All content included in the Software, including, without limitation, text, graphics, images, logos, audio or video clips, digital downloads, data compilations, is the property of PetaChem or licensors and protected by the laws of the United States and other countries and international treaties. All trademarks that are not owned by PetaChem that appear in the Software are the property of their respective owners, which may or may not be affiliated with or connected to PetaChem.

6. Any reports or published results obtained with the Software will acknowledge its use by appropriate citation as follows:

TeraChem v 1.9, PetaChem, LLC (2009,2015). See http://www.petachem.com.

Any published work which utilizes TeraChem shall include the following reference:

"I. S. Ufimtsev and T. J. Martinez. Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients and First Principles Molecular Dynamics, Journal of Chemical Theory and Computation, 5:2619-2628, 2009."

Electronic documents will include a direct link to the official TeraChem page at http://www.petachem.com.

1. Feedback. All suggestions, comments, or other feedback concerning your experience with or use of the Software that may be given to PetaChem ("Feedback") will be given voluntarily and without obligation or restriction of any kind. PetaChem may use such feedback for any purpose. Due to the nature of development work, PetaChem will not commit to correcting any reported errors or discrepancies. Feedback will not create any confidentiality obligation for PetaChem. You will not give Feedback that is subject to license terms that seek to require any product, technology, or service that incorporates or is derived from Feedback, or any intellectual property, to be licensed to or otherwise shared with you or any third party.

2. Disclaimers of Warranty and Liability.

   (a) TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, PETACHEM PROVIDES THE SOFTWARE "AS IS" AND WITH ALL FAULTS. PETACHEM HEREBY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY REPRESENTATIONS, WARRANTIES AND CONDITIONS REGARDING THE SOFTWARE, INCLUDING BUT NOT LIMITED TO MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, QUIET ENJOYMENT, OR NONINFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY OF THE SOFTWARE, ITS USE OR PERFORMANCE, REMAINS WITH YOU.

   (b) IN NO EVENT WILL PETACHEM BE LIABLE FOR ANY LOSS OR DAMAGE, INCLUDING WITHOUT LIMITATION ANY SPECIAL, INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL LOSS OR DAMAGE WHATSOEVER, WHETHER FOR BREACH OF CONTRACT, IN NEGLIGENCE OR ON ANY OTHER THEORY OF LIABILITY, ARISING OUT OF OR IN ANY WAY RELATED TO THE SOFTWARE, THE PROVISION OF OR FAILURE TO PROVIDE TECHNICAL SUPPORT, OR OTHERWISE UNDER OR IN CONNECTION WITH THIS AGREEMENT, EVEN IF PETACHEM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. YOU ACKNOWLEDGE THAT THIS ALLOCATION OF RISKS IS A PART OF THE BARGAIN OF THIS AGREEMENT. Notwithstanding, PetaChem's total liability under or in connection with this Agreement and your exclusive remedy for all of the foregoing, however arising, is limited to direct damages up to the cost of the Software to you. The foregoing limitations, exclusions, and disclaimers will apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose. Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations may be inapplicable to you.

3. Export. You may not use or otherwise export or reexport the Software except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software may not be exported or reexported (a) into any U.S. embargoed countries or (b) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce Denied Person's List or Entity List. By using the Software, you represent and warrant that you are not located in any such country or on any such list. You also agree

that you will not use the Software for any purpose prohibited by United States law, including, without limitation, the development, design, manufacture or production of nuclear, missiles, or chemical or biological weapons.

4. Miscellaneous.

   (a) This Agreement will be governed by and construed in accordance with the laws of the State of California as if entered into and performed wholly within the state and without regard to the principles of conflicts of law. You consent to exclusive jurisdiction and venue in the courts within the Northern District of California.

   (b) This Agreement constitutes the entire agreement between parties with respect to the Software and merges all prior and contemporaneous communications. If any provision of this Agreement is held to be void or unenforceable for any reason, that provision will be enforced to the maximum extent permissible so as to effect the economic benefits and intent of the parties, and the remaining provisions of this Agreement shall remain in full force and effect.

   (c) Neither party's failure or delay in exercising any of its rights will constitute a waiver of such rights. Any waiver or amendment of any provision of this Agreement will be effective only if in writing and signed by authorized representatives of both parties.

   (d) Neither party may represent or bind the other in any way and nothing in this Agreement shall be construed as creating of the relationships of joint venturers, partners, employer and employee, franchisor and franchisee, master and servant, or principal and agent.

12 Release Notes

v1.9:

- Effective core potentials are now supported
- TeraChem runs on Maxwell GPUs (e.g., GTX980, Titan X)
- Equilibrium PCM solvation (including both energies and gradients)
- Frequency analysis was revamped and now allows restarting
- Initial condition generation using Wigner/Husimi sampling
- Tamm-Dancoff Time-dependent Density Functional Theory (TDDFT) and Configuration Interaction Singles (CIS). Both energies and gradients are available for CIS. Only energies are available for TDDFT.
- Basis set and initial guess file formats changed to be more clear. Old basis set and initial guess files are no longer compatible.

v1.50K:

- Fixed memory allocation problem which caused CUDA errors when running on multiple cards with shells that had few basis functions, e.g. when running with a single set of d functions over four cards.
- Implemented workaround for NVIDIA's texture bug (which caused Tesla C2075 and GeForce 580 GTX to hang randomly). Although we still do not recommend these cards, all tests so far suggest they can run TeraChem without errors.
- Document the ability to use ghost atoms (atoms that have basis functions but no electrons or nucleus – used in computing basis set superposition error).
- Document the ability to use matrix purification instead of diagonalization.
- Added A-DIIS/DIIS convergence scheme
- TeraChem runs on Kepler GPUs (e.g. Titan, K20)

v1.93P

- Allow use of multiple basis sets for different elements `$multibasis`
- Polarizable continuum methods for ground and excited states
- TeraChem runs on Maxwell and Pascal GPUs (e.g. Titan X-Pascal, P100)