Massachusetts Institute of Technology

User's Manual

`molSimplify` version 1.0

by

E. Ioannidis

KULIK GROUP

March, 2016

Contents

1 General information

molSimplify is an open source utility that incorporates geometric manipulation routines necessary for the generation of transition metal complexes,automated setup and completion of electronic structure calculations, post-processing and data analysis. The software generates a variety of coordination complexes with any number of metals coordinated by ligands in a single or multidentate (chelating) fashion. The code can both build the coordination complex starting from a single metal atom or work to functionalize a more complex structure (e.g. a porphyrin or other metal-ligand complex) by including additional ligands or replacing existing ones. molSimplify builds intermolecular complexes for evaluating binding interactions and generating candidate reactants and intermediates for catalyst reaction mechanism screening and also supports interaction with chemical databases. Furthermore, it provides a Graphical User Interface (GUI) and is thus accessible to a wider audience since it does not require a lot of prior computational chemistry experience.

2 Installation

2.1 From source

2.2 Binaries

3 Structure generation module

test

3.1 test

daf

3.2 Building simple structures

test

3.3 Force field optimization

test

3.4 Custom geometries

test

3.5 Chelating ligands

test

3.6 Building custom structures

test

3.7 Replacing existing ligands

test

3.8 Additional molecule placement

test

4 Random generation module

test

5 Input files & Jobscripts

test

5.1 Input file generation

test

5.2 Jobscript generation

test

6 Interaction with chemical databases

test

6.1 Similarity search

test

6.2 Database screening

test

7 Post processing

test

7.1 Summary

test

7.2 Charges

test

7.3 Wavefunction

test

7.4 Cubefiles

test

7.5 Molecular orbitals

test

7.6 Natural Bonding Orbitals

test

7.7 Delocalization indices

test

8 Extras

8.1 Updating the database

8.2 Jobscript generation

In this tutorial we will learn some basics about working with the command line and how to run a simple electronic structure calculation on our supercomputer `gibraltar`. The first step is logging in the machine using the provided username and password. Once the user is logged in the cluster, the required input files for the calculation need to be prepared. However in order to do all that, you need to get somewhat familiar with the basic mechanics of the command line and that's what we are going to review now. The following discussing is by any means very brief and I suggest you look online for more resources regarding UNIX (Linux, OSX), the command line, bash shell etc.

In the beginning you should get familiar with using the command line or terminal. In OSX you can find the terminal using Spotlight or under /Applications/Utilities/Terminal. On linux systems it should be on your homepage or you can search for it using the super key (windows logo on most keyboards). Once you open terminal you will see a command prompt where you can type in various commands. The program that interprets these commands and tells the computer what to do is called shell. The most widely used shell is `bash` (Bourne Again SHell). Other popular ones are ksh, tcsh, csh etc. In our machines (both Linux and OSX), we use the bash shell, so we will focus on this one. The most important commands in bash that you should know are the following:

Listing 1: Basic bash commands

```
~        # home directory
.        # current directory
..       # up one directory in the tree
ls       # lists files in current directory (ex: ls .)
cd       # change directory (ex: cd ~/myruns)
mkdir    # create new directory (ex: mkdir newdir)
mv       # move or rename file/folder (ex: mv oldf newf)
rm       # remove file (ex: rm file)
rm -rf   # remove folder (ex: rm -rf fold)
cp       # copy file (ex: cp file file2)
cp -R    # copy folder (ex: cp -R fold fold2)
pwd      # lists current directory
cat      # outputs contents of file (ex: cat file)
touch    # creates new file (ex: touch newfile.txt)
```

```
grep    # searches for text within a file (ex:grep 'test' file)
find    # finds files/folders in directory (ex: find . -name file)
```

In order to be able to run programs on our supercomputer you need to log in remotely. In order to remotely control a machine we will use the `ssh` command. To log into gibraltar you will type:

```
ssh username@gibraltar.mit.edu
```

The first time the program will ask you if you want to add gibraltar in your known keys (type YES) and then you will be asked to type in your password. If you type it correctly then you will see another command prompt but this time it's on gibraltar and not your local machine and you can perform tasks remotely on gibraltar using the same commands you learnt before.

Once you've performed some calculations you may want to copy back some files to your local machine (or the other way around). In order to do that there three options, `scp`, `sftp` and mounting the filesystem on your local machine. In the beginning I would recommend scp or sftp. For scp from your local machine if you want to transfer file test.txt to a local folder named localfold in your current directory you should type:

```
scp username@gibraltar.mit.edu:test.txt localfold/
```

If you want to transfer a file from your local machine to the remote one type:

```
scp localfile username@gibraltar.mit.edu:remotefile
```

To transfer folders instead of type scp you should type "scp -r".

The second alternative, sftp, is an interactive file transfer program. To use it open a new window on the terminal on your local machine and type:

```
sftp username@gibraltar.mit.edu
```

You will again type the same password and another command prompt will show up. From there you can navigate using cd, ls etc as in a normal window. In order to copy a file from the remote machine to your local machine you should type "`get remotefile`" or "`get -r remotefolder`" if it is a folder. In order to move a file from your local machine to the remote machine type "`put localfile`" or "`put -r localfolder`".

If you need to edit files within gibraltar, you should use a text editor. The most widely used ones are vi, nano and emacs. I personally like vi the most, but it's usually a matter of preference. In order to edit a file using vi you should type "`vi filename`". Then the contents of the file will show up, however you won't be able to just type and delete whatever you want as in a normal text editor. In vi there are 2 modes, the input and the command mode. By typing i you enter into the input mode where you can type in whatever you want like a normal editor. Typing Esc brings you back to the command mode where you can perform various tasks such as copying/deleting multiple lines, replacing text and a lot more. In order to save the file type Esc and then :w and to quit the program Esc and :q. For more information read online about vi.

Calculations on the cluster can be run either interactively or through a queuing system. The queuing system is a program that distributes resources such as memory, gpus, cpus etc among the various users in the group. Our queuing system is called SGE (System Grid Engine) and along with SLURM are the two most widely used. The default way to run calculations is through the queuing system, however under certain circumstances someone can request specific resources to be reserved in order for the user to run interactive calculations. If you think you need to reserve a number of nodes for your exclusive use, please contact HJK or EI. As of December 2015, `gibraltar` consists of 28 GPU nodes with 224 GPUs and 4 CPU nodes with 64 CPUs.

Jobscript

For each job the user needs to submit a jobfile that specifies what resources will be required for his/her job to run. These could be amount of memory, number of cpus, number of gpus, execution time etc. Furthermore, inside the same file the user needs to load any environmental variables that will be used by the job and specify the program that will be run. In our example we will run a `terachem` calculation using 1 GPU, 8GB of memory and 24 hours of maximum execution time. The corresponding jobscript file is shown in Listing 2.

The first thing that needs to be specified is the interpreting shell, which by default is bash. Then a short name (10 characters or less) is given to make it easier for the user to identify the job once submitted. The `-cwd` option tells the queuing system to start the job from the current working directory. The next two lines request 24 hours of computing time and 8G of memory. With the `-q` option, the user specifies to which queue the job must be submitted to. In our cluster we have several queues that serve different purposes with the `gpus` queue being the general queue for GPU jobs shorter than 1 week, the `gpusbig` queue serving longer GPU jobs and the `cpus` queue accommodating cpu jobs. Because `terachem` is a GPU-based code and because we expect our job to be fairly short, we will submit it to the gpus queue. The next 2 lines specify on how many GPUs we want our job to run. The line `-l gpus=1` should remain unchanged and the number of GPUs should be specified using the `smp` option followed by the number of GPUs. This flag tells the queuing system to spawn multiple processes each of them controlling one gpu. This number should be the same as the `OMP_NUM_THREADS` value later in the jobscript. One important thing to notice is that your jobs will not run in the same folder that you submit the jobs from. Therefore you need to specify which files are necessary for your job to run in order for the queue to copy them to the running directory and then also specify which output files to copy back when the job is done. This is done using the `-fin` and `-fout` flags. In our case we want the queue to copy the terachem input file and the coordinates file and then copy back the folder with all the results. The `module load` directive loads all necessary environmental variables for cuda (GPU drivers) and terachem before the program is run. The next flag tells the queuing system the number of threads to be spawned (should be the same as the smp value specified before). Finally the last line runs the actual terachem job and directs the output to an output file in the current directory.

Listing 2: Example jobscript file

```
#$ -S /bin/bash       # specifies interpreting shell for the job
#$ -N cafrun          # identifier for the job
#$ -cwd               # run the job from the cwd
#$ -l h_rt=24:00:00 # specify maximum runtime
#$ -l h_rss=8G        # specify requested memory
#$ -q gpus            # specify which queue to submit the job to
#$ -l gpus=1
#$ -pe smp 1          # number of parallel processes/GPUs to run
# -fin caffeine.in    # copy input file in the workdir
# -fin *.xyz          # copy geometry file in the workdir
# -fout scr/          # copy back results in the current directory

module load cuda      # loads drivers etc for CUDA
module load terachem # loads the appropriate terachem variables

# number of threads to be spawned (needs to be the same as smp)
export OMP_NUM_THREADS=1

# run the actual calculation and pipe output to caffeine.out
terachem caffeine.in > $SGE_O_WORKDIR/caffeine.out
```

In order to actually submit a job, you will use the command `qsub <jobfile>`. Once your job is submitted, you can track its progress using the `qstat` command. The status of your job will turn from `qw` (queued waiting) to `r` (running) when your job starts running. If you ever want to delete a running or queued job you could do so using the `qdel` command followed by the ID of the job you want to kill.

Terachem input file

Besides the jobscript which provides instructions for the queueing system, we need to tell terachem which calculation to perform and using what methods. In the following example terachem input file we specify the parameters for running a simple geometry minimization calculation on caffeine using DFT. The order of the keywords does not matter in the terachem input file. The first line specifies the type of run which in our case is geometry optimization (minimize) while an alternative would be a single point energy calculation (keyword energy). The next line specifies the coordinates file which contains the 3D coordinates and the type of atoms included in the molecule. The `basis` keyword specifies the basis set to be used, while the `method` keyword specifies the electronic structure method to be used. In our case we will run DFT calculations using the B3LYP exchange-correlation functional. The charge and the spin multiplicity of the system are specified next followed by the `scrdir` keyword that tells the program where to place all the output files produced during the run.

Listing 3: Example `terachem` input file

```
run minimize                    # run to be performed
coordinates caffeine.xyz        # coordinates file (xyz or pdb)
### method details ###
basis 6-31gs                    # basis set used
method b3lyp                    # method to use (HF, DFT etc)
### system properties ###
charge 0                                # system charge
spinmult 1                      # spin multiplicity (2S+1)
### technical details ###
scrdir scr                      # directory for output files
end
```

You can submit your job using the `qsub jobscript` command. Once the program finishes, various files will be generated that could be parsed in order to extract useful information from the run.