Massachusetts
Institute of
Technology

User's Manual

`molSimplify` version 1.0

KULiK GROUP

April, 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.

1.1 Obtaining molSimplify

The binaries, source code and useful documentation can be obtained online by visiting: `http://molsimplify.mit.edu`.

1.2 Citing molSimplify

Any published work that utilizes molSimplify shall include the following reference:

Efthymios I. Ioannidis, Terry Z. H. Gani, and Heather J. Kulik. molSimplify: A toolkit for automating discovery in inorganic chemistry. *J. Comp. Chem.*, 37(22):2106–2117, aug 2016

1.3 License

1.3.1 molSimplify

The software is distributed free of charge under the GPL license:

1.3.2 Pybrain

molSimplify makes use of the open-source Pybrain library (`http://pybrain.org/pages/home`) for neural network implementation. This work is not endorsed by Pybrain. Pybrain is licensed under the BSD licence:

1.4 Acknowledgments

This software was developed by Efthymios I. Ioannidis, JP Janet, Terry Z. H. Gani and Heather J. Kulik at the Massachusetts Institute of Technology. The authors would like to especially thank Terry Z. H. Gani for thoroughly testing the code and providing valuable feedback.

2 Installation

2.1 System requirements

The binaries and the source code have been tested on 64-bit Ubuntu 14.14 and on 64-bit OSX 10.10.7 and are expected to work on any newer system equipped with Python 2.7 or greater. No particular memory requirements or processor capabilities are needed.

2.2 Conda Package

Conda (`http://conda.pydata.org/docs/`) is a free, multi-platform and multi-language package and environment manager. Using Conda, it is possible to easily obtain molSimplify and all the environmental dependencies in one step. This is the recommended method for end users to acquire molSimplify. In order to acquire molSimplify using Conda, follow these directions:

1. Ensure you have an up-to-date version of Conda installed. Binary installers are available for all platforms from `https://www.continuum.io/downloads` - we recommend using Anaconda (as opposed to Miniconda).

2. Ensure that you have an up-to-date version of the Anaconda Cloud Client by running

   ```
   $ conda list -n anaconda-client
   ```

   If the package is not found, you need to install it (we recommend running this command as SU on Linux systems):

   ```
   $ sudo conda install anaconda-client
   ```

3. Now you are ready to install molSimplify by running:

   ```
   $ conda install -c hjkgroup molSimplify
   ```

   Here, '-c' means that the files will come from the Anaconda Cloud, and hjkgroup is our repository there. Don't forget the capital S! This single command will bring openbabel with python bindings with it, if they are missing.

4. If you want to use the GUI (graphical user interface), you'll need PyQt5, which you can get (if you don't have it) with

   ```
   $ conda install -c dsdale24 pyqt5
   ```

5. Now you are ready to go! Launch molSimplify from the terminal using:

   ```
   $ molsimplify
   ```

You can add any arguments here in the normal manner, for example:

```
$ molsimplify -core Fe -lig water -ligocc
```

would generate a 6-coordinated iron-hexaaqua complex.

While Conda remains our recommended method, other installation options exist:

2.3 From source

In order to install molSimplify from source it is necessary to install openbabel with its Python bindings (pybel) and numpy. Furthermore, additional software is required in order to use the Graphical User Interface (GUI).

2.3.1 Linux

For Linux Ubuntu, the only requirements are a working version of Openbabel with the Python bindings (pybel) and Numpy installed. To install numpy, you can follow the online instructions or use pip install numpy. We suggest installing pybel from source using the following procedure:

1. wget `https://sourceforge.net/projects/openbabel/files/openbabel/2.3.2/openbabel-2.3.2.tar.gz`
2. tar -xvf openbabel-2.3.2.tar.gz
3. mkdir build ; cd build
4. cmake .. -DPYTHON_BINDINGS=ON
5. sudo make
6. sudo make install

.

In order to enable the GUI, the library PyQt5 with its dependencies needs to be installed first. In order to do that we suggest the following procedure:

1. Download and install Qt5 from source (`developer.qt.nokia.com/`).
2. Install SIP(`http://pyqt.sourceforge.net/Docs/sip4/installation.html`) from source. If Python.h is missing use `sudo apt-get install python-dev` .
3. Install PyQt5 from source (`http://downloads.sourceforge.net/project/pyqt/PyQt5/`).

Once you have installed PyQt5, open an interactive Python terminal window and check whether

```
Once you have installed Pybel (and optionally PyQt5), you proceed with the
\begin{lstlisting}[language=bash]
  $ sudo python setup.py install
```

The code can then be used from the command line as described in the Conda section.

2.3.2 Mac OSX

In order to run molSimplify from source, we need to install pybel first. We suggest using `brew` as explained in section 2.4.2 or compiling from source. Then install imagemagick (see section 2.4.2). In order to use the GUI, PyQt5 and its dependencies need to be installed as well. The easiest way is to use the following procedure:

1. Install Qt5 with: `sudo brew install qt5`
2. Install SIP with: `sudo brew install sip`
3. Install PyQt5 from source (`http://downloads.sourceforge.net/project/pyqt/PyQt5/`) with option --sip-incdir=/usr/local/include

Once the correct dependenices are installed, you can follow the same setup instructions as in the Linux case.

2.4 Binaries

Binaries of the program have been compiled for both Ubuntu Linux and OSX. The installation of the binaries depends on the platform and analytical instructions are given below.

2.4.1 Linux

For Linux Ubuntu, the only requirement is a working version of Openbabel with the Python bindings (pybel) installed. We suggest installing pybel from source using the following procedure:

1. wget `https://sourceforge.net/projects/openbabel/files/openbabel/2.3.2/openbabel-2.3.2.tar.gz`
2. tar -xvf openbabel-2.3.2.tar.gz
3. mkdir build ; cd build

4. cmake .. -DPYTHON_BINDINGS=ON

5. sudo make

6. sudo make install

Alternatively pybel can be installed using the standard ubuntu package manager with `apt-get install python-openbabel`.
In both cases, once the installation is complete, open an interactive Python terminal and make sure that `import openbabel` and `import pybel` work before running molSimplify.
With pybel installed, the user can download the binary and the supporting files and run the program. The program will guide the user in order to set up a configuration file that is stored under `/home/user/.molSimplify` in the user's home directory and contains a line that specifies the installation directory (top directory) of the program where all the supporting files are located. The line has the format `INSTALLDIR=path`. The binary can be moved anywhere, however if the top directory of molSimplify is moved the `.molSimplify` file will have to be manually updated. The configuration file contains also the paths for files used in database search and post processing analysis (see section 2.6).

2.4.2 Mac OSX

For OSX, a package installer has been created that automatically installs the required software for running the molSimplify app. The user can open the `molSimplify.pkg` installer, follow the instructions and get molSimplify installed.
If the package installer fails, the user can manually install the required packages and then download a copy of the molSimplify app. Two packages are required in order to run the molSimplify app, pybel and imagemagick. The easiest way to install both is to use the `brew` package manager and issue the following commands on a terminal window:

1. sudo brew install open-babel --with-python

2. sudo brew install imagemagick

With pybel installed, the user can download the app, place it under /Applications/molSimplify.app and run the program.
If `brew` is not installed, install it by running `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"` in a terminal window.

2.5 Installed files

With both the binaries or the source code a set of additional files are stored that are required by molSimplify. If you use molSimplify with the molSimplify.app (OSX) these files are stored under /Applications/molSimplify.app/Contents/Resources/, otherwise they are in the installation directory of molSimplify as indicated in the `.molSimplify` file in the home directory. These additional files include the following:

- icons: Folder containing the icons used in the GUI.
- Data: Folder containing the available coordinations/geometries and Metal-Ligand bond length database.
- Cores: Folder containing the local saved cores.
- Ligands: Folder containing the local saved ligands.
- Bind: Folder containing the local saved extra molecules.

You can manually update the contents of these folders, however it is recommended to use the modules included in the GUI in order to update them.

In the Conda version, package resource management is handled by Conda and Python, and you don't need to worry about them.

2.6 Databases & Post-processing

2.6.1 Chemical database setup

The program is able to interact with external chemical databases such as ChEMBL. These databases are essentially sdf files that contain information about molecules. In order to link these databases to molSimplify the user needs first of all to download the database sdf file. Once the file is downloaded, we recommend using Openbabel's fast search index generation by issuing `babel DBfile.sdf -ofs`. Openbabel will then create a .fs file that can be used for fast screening and will substantially accelerate the database search. Note that OpenBabel can handle sdf files smaller than 4GB, so you may have to break larger files before fast indexing them.
All the sdf and fs files should be placed inside a common directory. Then by clicking on the search DB window on the GUI, a prompt will ask you to specify that directory. Once you specify the directory containing your sdf and fs files, molSimplify will automatically detect them and the database search will be ready to use. Alternatively the user can specify the path by modifying the `.molSimplify` file in the home directory and adding a line with `CHEMDBDIR=path`.

A short list of available chemical databases is:

- ChEMBL `https://www.ebi.ac.uk/chembl/`
- eMolecules `https://www.emolecules.com/`
- ChEBI `http://www.ebi.ac.uk/chebi/`
- Zinc `http://zinc.docking.org/`
- PubChem `http://pubchem.ncbi.nlm.nih.gov/`

2.6.2 Multiwfn setup

Many of the features included in the post processing module of molSimplify depend on the external program Multiwfn. The user needs to manually install Multiwfn from `https://multiwfn.codeplex.com/` and then specify the installation directory by clicking the post processing option in the GUI and then following the prompt instructions. Alternatively the user can specify the path to the Multiwfn executable by modifying the `.molSimplify` file in the home directory and adding a line with `MULTIWFN=path`.

3 GUI overview

Using the main window of the Graphical User Interface (GUI-Figure2), the user is able to access most of the functionality of molSimplify. On the left of the main window, the input options for building and customizing structures can be specified. On the bottom left, clicking the corresponding buttons the user can list the available ligands, cores and extra molecules as well as add new molecules to the local database. In the middle, there are options for randomly generating structures and general options for building complexes. Finally on the right, the user can specify additional molecules for studying intermolecular interactions, parameters for generating quantum chemistry input files and queuing system jobscripts.

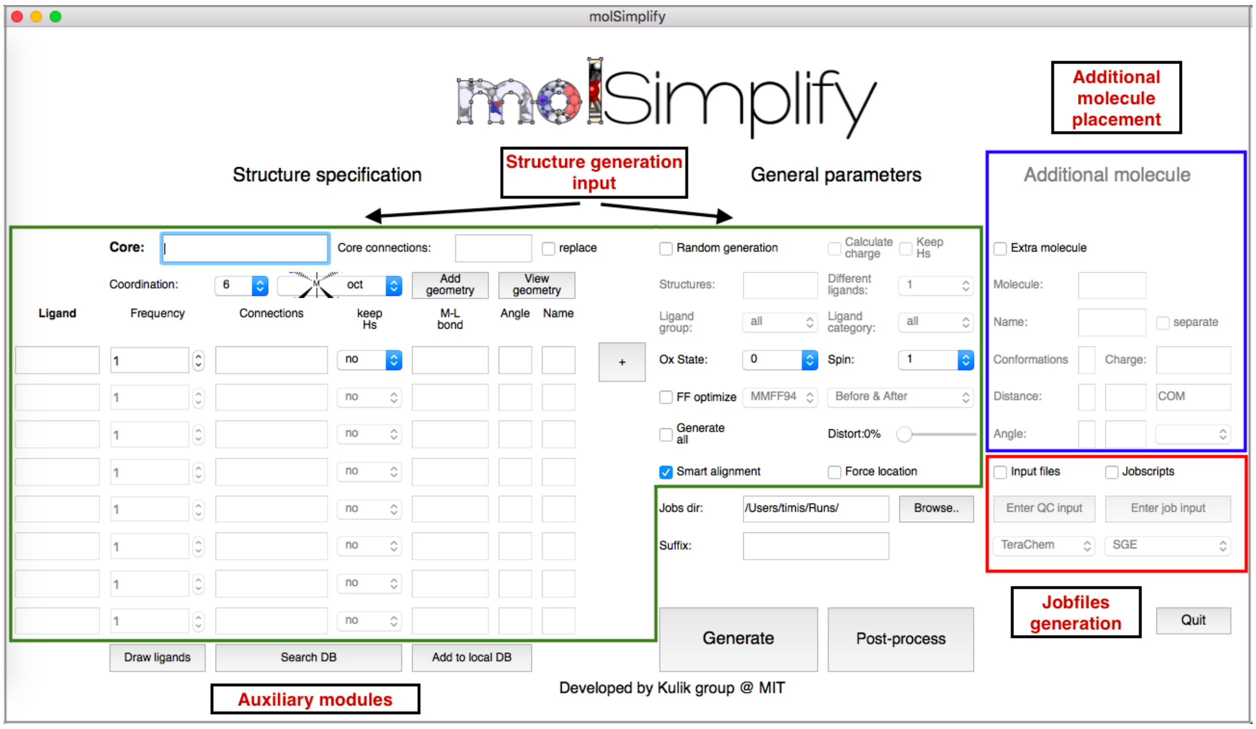


Figure 1: The main window of the GUI where the structure generation is coordinated.

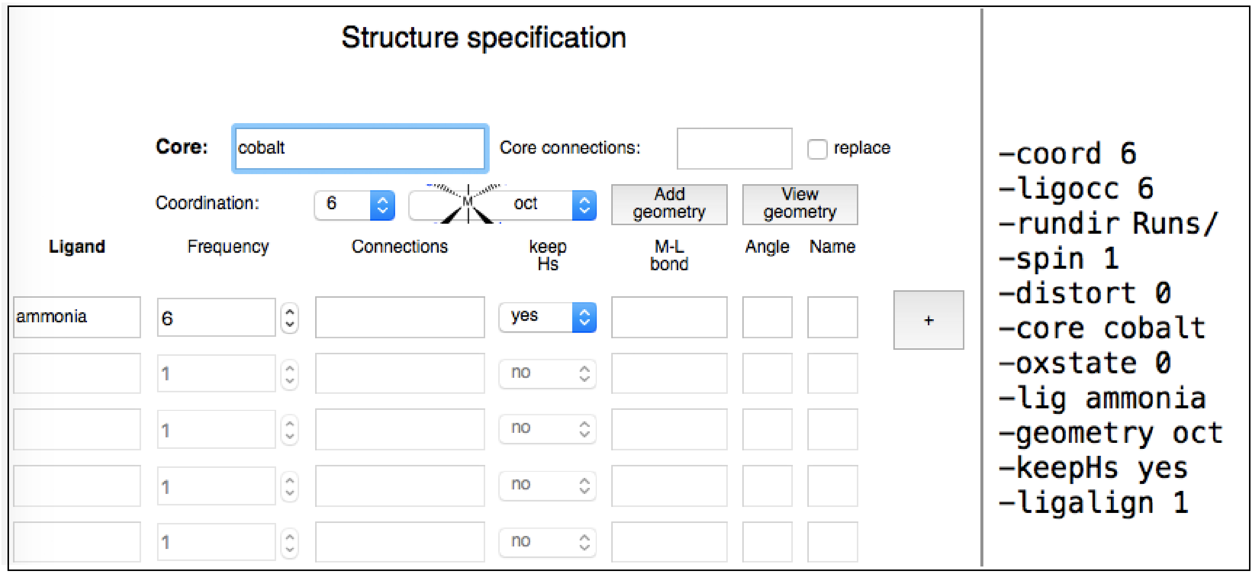
In all cases where the GUI is used, an input file is generated and stored in the jobs directory under the names `geninput.inp` for the structure generation module, `dbinput.inp` for the database search module and `postproc.inp` for the post-processing module. The program can be used without the gui by running from the command line and specifying the input file using the `-i` flag (e.g. molSimplify -i geninput.inp). The software supports command line arguments as well. An input file can be generated from the GUI using the *Save as* button and existing input files can be loaded back to the GUI using the *Load* button. The program also generates a log file named `molSimplify.log` in the jobs directory where the input options and additional information is being printed for debugging and logging.

4 Structure generation module

The main function of molSimplify is building new structures or editing existing ones. In this part of the user guide, several examples for building structures are included.

4.1 Building simple structures

Building a simple symmetric metal coordination complex in molSimplify requires minimal input by the user. As an example, we are going to build an octahedral, 6-coordinate complex with a single cobalt metal in the middle and 6 ammonia ligands coordinated around (Figure 3 left). The corresponding input parameters on the GUI and the generated input file are shown below.



Core: cobalt Core connections: ☐ replace

Coordination: 6 oct

Ligand	Frequency	Connections	keep Hs	M-L bond	Angle	Name
ammonia	6		yes			
	1		no			
	1		no			
	1		no			
	1		no			

```
-coord 6
-ligocc 6
-rundir Runs/
-spin 1
-distort 0
-core cobalt
-oxstate 0
-lig ammonia
-geometry oct
-keepHs yes
-ligalign 1
```

Figure 2: Input parameters for building a simple symmetric octahedral complex using either the GUI or an input file.

On the top left of the GUI, the user needs to specify first the core structure. This core structure can be either a predefined atom or molecule from the local database or a new one in SMILES format or a file path to an xyz or mol file. For example, the following are valid cores for molSimplify: `iron`, `[Fe]`, `~/iron.xyz`. Multiple cores can be also specified separating them with comma or whitespace. The program will then loop and generate one structure for each specified core. Multiple cores are also supported in other features such as the random generation (see section 5).
Next, the coordination is specified by selecting first the coordination number on the metal (6 in this example) and then a corresponding geometry (octahedral here). New geometries can

be added using the appropriate option and the current geometry can be viewed by clicking the `View geometry` button.
Once the core is specified, the user needs to specify the ligands that are going to be attached to the core. In this example, we only use one ligand, namely ammonia. A ligand can be either a predefined atom or molecule, a SMILES string or a path to an xyz/mol file (for example: `ammonia, N, ~/NH3.xyz` are all valid options). Once the ligand is specified under the `Ligand` field, the frequency or number of copies of that ligand is specified. In our symmetric ligand we are just going to use 6 ammonia copies (specified on the `Frequency` field). The field `Connections` contains the indices of the atoms on the ligand that are going to be connected to the metal and for predefined ligands (such as ammonia), it's not required. Because the program by default strips one Hydrogen from the connection atom of the ligand (to allow the bond formation with the metal), we need to specify that we want to keep all Hydrogens in ammonia by selecting `yes` in the `keep Hs` field, otherwise we would end up with NH_2 instead. Next, in the `M-L bond` field, the user can overwrite the default value and specify a custom Metal-Ligand (in this case cobalt-nitrogen) bond length in Angstroms. The software contains a database with trained Metal-Ligand bond lengths collected from various DFT simulations and it uses this information if available. In case the Metal-Ligand pair is not present in the database, the program searches for a similar pair or uses the sum of the covalent-radii to approximate the bond. Next, the user can distort the specified geometry by selecting a pair of angles (see section 4.5). Finally, if the user requests a ligand using a file path or a SMILES string, a name for that ligand can be specified (not required for predefined ligands).
The same options can be specified using an input file (automatically generated if using the GUI). Along with the options specified in Core/Ligand fields, some additional options are included by default in the input file. In this case the keyword `-coord` specifies the coordination number, `-ligocc` the frequency of the ligands, `-rundir` the directory where the structure files will be placed, `-spin` the spin multiplicity of the system (by default it's 1), `-distort` the percentage of random distortion of the geometry (by default it's 0), `-core` specifies the core of the structure, `-oxstate` specifies the oxidation state of the metal (default is 0), `-lig` the ligands included in the structure, `-geometry` the geometry of the complex, `-keepHs` whether to keep the Hydrogens on the ligand's connection atom (default no) and `-ligalign` is a flag that forces a specific order when the complex is being built (see section 4.4).
Once the program finishes it generates an xyz file with the coordinates of the desired structure (Figure 3, left).

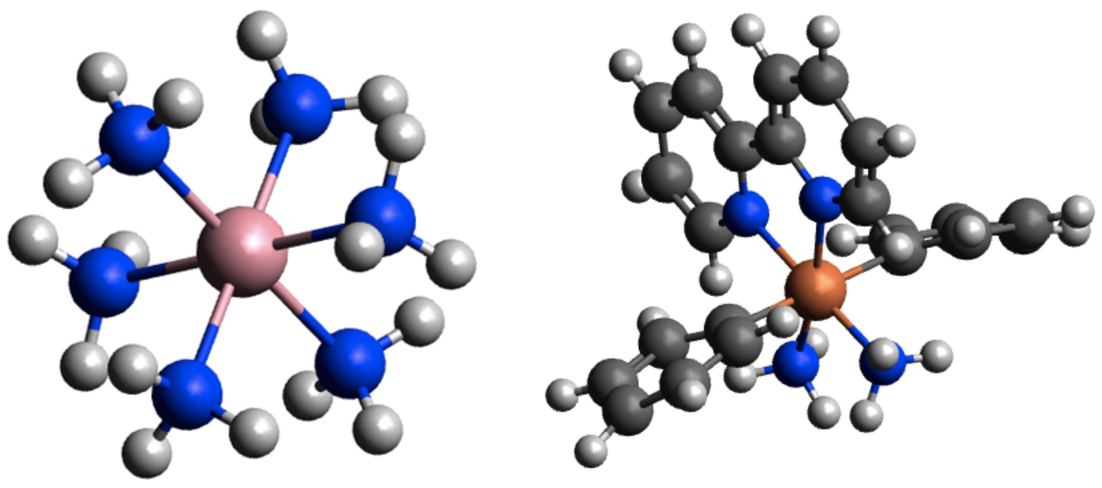
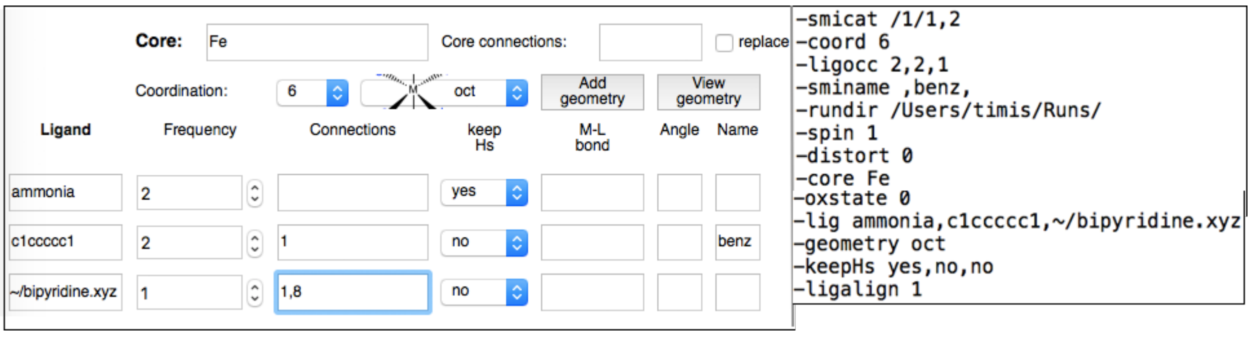


Figure 3: Left:A six-coordinated octahedral structure with a cobalt center and 6 ammonia ligands around. Right: A six-coordinated octahedral structure with an iron center, 2 benzenes, 2 ammonias, and 1 bipyridine around.

4.2 Building more complicated structures

In case of asymmetric structures, more ligands need to be specified in the input. In this example we are going to generate again an octahedral, 6-coordinate structure, this time with an iron center and 3 different ligands, namely ammonia, benzene and bipyridine (Figure 3, right). The GUI options and the corresponding input file are shown in Figure 4.


Core: Fe   Core connections: replace

Coordination: 6   oct   Add geometry   View geometry

Ligand	Frequency	Connections	keep Hs	M-L bond	Angle	Name
ammonia	2		yes			
c1ccccc1	2	1	no			benz
~/bipyridine.xyz	1	1,8	no			

```
-smicat /1/1,2
-coord 6
-ligocc 2,2,1
-sminame ,benz,
-rundir /Users/timis/Runs/
-spin 1
-distort 0
-core Fe
-oxstate 0
-lig ammonia,c1ccccc1,~/bipyridine.xyz
-geometry oct
-keepHs yes,no,no
-ligalign 1
```

Figure 4: Input parameters for building more complicated structures.

Once the core and coordination are specified, the options for the 3 different ligands have to be included. Initially, we select ammonia from the list of predefine ligands as in the previous example, however here we only want 2 copies of the ligand, thus we select 2 in the `Frequency` field. Next, we specify 2 copies of a benzene ligand using its corresponding SMILES string *c1ccccc1* and selecting 2 in the `Frequency` field. Because this ligand is not predefined we also need to specify the index of the atom that is going to be connected to the metal (a carbon atom). The user can visualize the ligand with its atom labels in a molecular viewer such as Avogadro, or use the `View ligands` option on the bottom left of the GUI to locate the correct connection atom index (Figure 5). When the `View ligands` option is selected, the program will draw a 2D representation of the ligands with their atomic labels and also generate a png image inside the running directory. In this case atom 1 corresponds to a carbon atom on the ring that is going to get a Hydrogen stripped before it gets connected to the metal (option `no` in the `keep Hs` field). We use the default values for M-L bond length and do not distort the octahedral geometry (empty `M-L bond` and `Angle` fields) and name this ligand `benz` using the `Name` field. Similarly, we specify the last ligand which in this case is bipyridine, defined using an input file located at ∼`/bipyridine.xyz`. Because this ligand is chelating and connects to the metal center with 2 connection atoms (bidentate), we need to specify 2 indices in the `Connections` field. If we visualize the ligand, we can see that the 2 nitrogens that are coordinating the metal center have the atomic labels 1 and 8 (Figure 5).

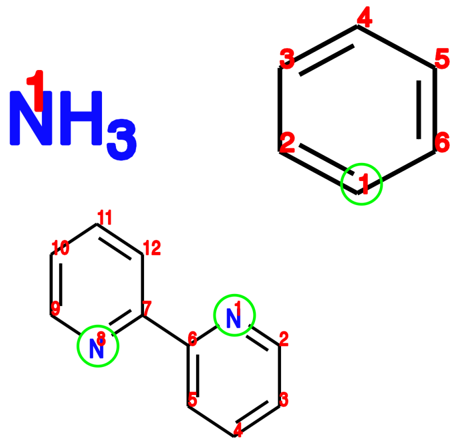
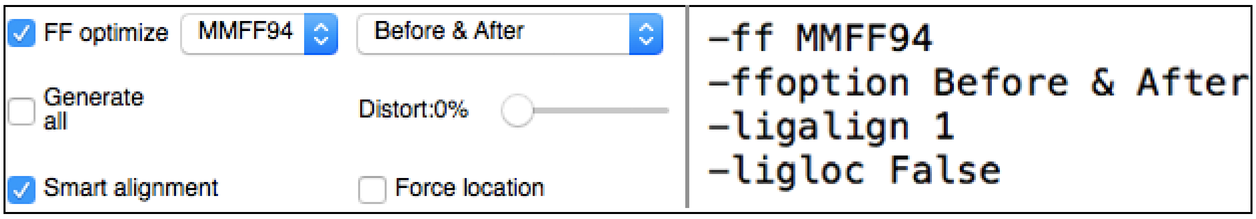


Figure 5: 2D representation of ligands with their corresponding atomic labels. Highlighted are the connection atoms for each particular ligand that need to be specified in the input.

In the corresponding input file, the connection atoms are specified with the `-smicat` keyword and are separated with '/' for each ligand (predefined ligands just need an empty value).

4.3 Force field optimization

In order to improve the quality of the generated structures, molSimplify has several options for relaxing them using force fields. The program supports 4 different force fields (default and recommended is MMFF94) and 3 different optimization options. Using the `Before` option, the several ligands included in the generation are force field optimized in isolation, before they are attached to the metal core. Using the `After` option, the ligands are force field optimized along with the metal core after they have been attached. In this option the whole structure is being optimized in the end as well, however the metal center is ignored and the connection atoms of the ligands held fixed during the optimization. Finally using the `Before & After` option, both `Before` and `After` optimizations are performed. In order to enable force field optimization, the user needs to select the corresponding option on the GUI or use the `-ff` keyword in the input file (Figure 6). Note that in some cases (such as very small ligands) FF optimization is not necessary or can result in worse structures. Therefore, ligands can be blacklisted by removing the force field optimization option during addition to to the database.



```
-ff MMFF94
-ffoption Before & After
-ligalign 1
-ligloc False
```

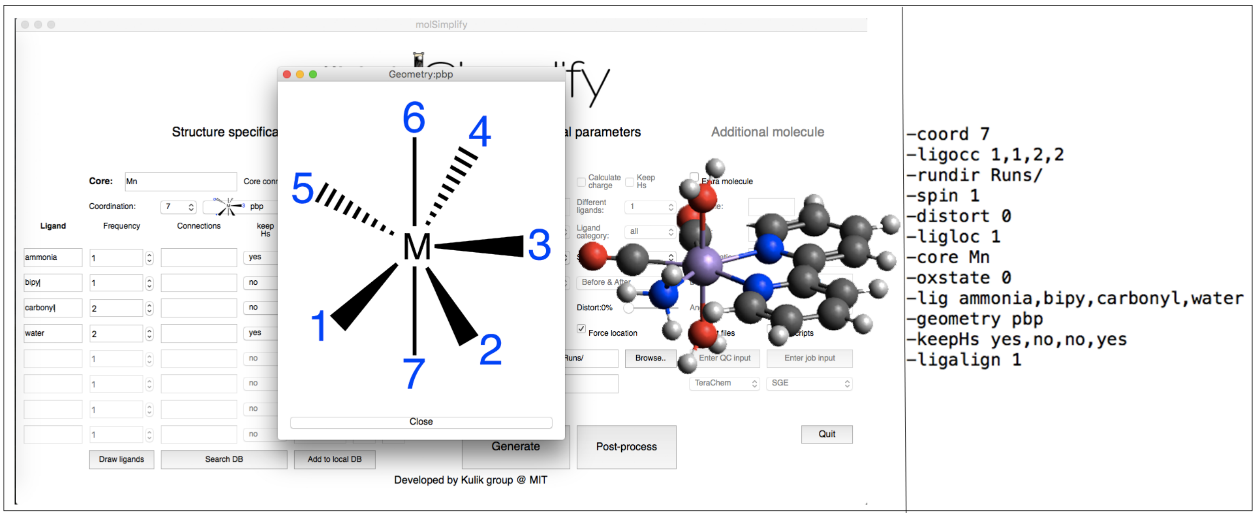
Figure 6: Force field optimization input parameters in the GUI and in the input file.

By selecting the generate all checkbox or specifying the corresponding keyword in the input file (`-genall True`), the program will automatically generate 4 xyz files. One with trained Metal-Ligand (ML) bond lengths and FF optimization, one with trained ML bond lengths and without FF optimization, one with simple covalent radii guess for the ML bond lengths and with FF optimization and one with covalent radii guess for the ML bond lengths and without FF optimization. This feature is particularly important in cases where the user doesn't know a priori if FF improves the predicted structure and if the trained ML bond length is a good estimate of the true bond length.

4.4 Smart alignment & Force order

In the previous examples, the options `Smart alignment` and `Force location` were first mentioned. The first option which is enabled in the input file with the keyword `-ligalign True` tells molSimplify to use a smart alignment algorithm. This means, that the order that the ligands will be attached to the metal will be shuffled according to their denticities and size so that the best possible structure is obtained (default is True).
The `Force location` feature can be used in order to generate complexes with desired stereochemical properties. By default, molSimplify will attach ligands to the best possible positions on the coordination template and not necessary in the order specified by the user. If the user wants to attach ligands to specific points in space, the force location option needs to be enabled. For example, let's say we have a pentagonal bipyramidal structure as shown

in Figure 7, where we want to attach an ammonia ligand in position 1, a bipyridine ligand in positions 2 and 3, 2 carbonyls in positions 4,5 and 2 water molecules in the axial positions 6,7. In order to do that, we specify the ligands in the correct order (fist ammonia, then bipyridine, then carbonyl and then water) and select the force location option to get the desired location. Force location and smart alignment are 2 independent options, the first one specifying the location of each ligand and the second one the order that the ligands will be attached.



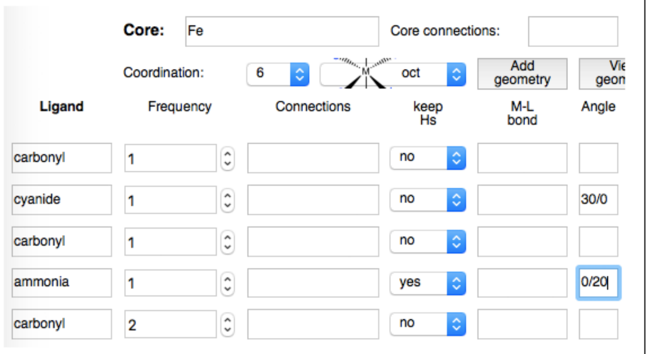
```
-coord 7
-ligocc 1,1,2,2
-rundir Runs/
-spin 1
-distort 0
-ligloc 1
-core Mn
-oxstate 0
-lig ammonia,bipy,carbonyl,water
-geometry pbp
-keepHs yes,no,no,yes
-ligalign 1
```

Figure 7: Input options for custom core ligand replacement.

Additionally, if the user wants for example to have 2 ammonia ligands at specific points, 2 separate entries are needed in the ligands table, both with ammonia as the ligand but in different rows that correspond to the desired connection points on the geometry template.

4.5 Custom angles

The user has fine control of the predefined geometries. The default connection points for the ligands can be distorted according to the needs of the specific application. For each ligand two distortion angles can be specified in the corresponding `Angle` field on the ligand table. The first one is a polar angle θ (0-360^o) and the second one an azimuthal angle ϕ (0-180^o) and correspond to a spherical coordinate system centered at the corresponding connection point. As an example we are going to generate an octahedral complex with 3 different ligands, carbonyl, cyanide and ammonia, with two of them corresponding to distorted sites on the octahedral complex (Figure 8). The two angles are separated using '/' on the corresponding field and the default value is 0.0/0.0 (no distortion). In the input file, the custom angles are specified with the keyword `-pangles`.



Core: Fe

Core connections:

Coordination: 6 oct

Ligand	Frequency	Connections	keep Hs	M-L bond	Angle
carbonyl	1		no		
cyanide	1		no		30/0
carbonyl	1		no		
ammonia	1		yes		0/20
carbonyl	2		no		

```
-coord 6
-ligocc 1,1,1,1,2
-rundir Runs/
-spin 1
-distort 0
-ligloc 1
-core Fe
-pangles ,30/0,,0/20,
-oxstate 0
-lig carbonyl,cyanide,carbonyl,ammonia,carbonyl
-geometry oct
-keepHs no,no,no,yes,no
-ligalign 1
```

Figure 8: Input options for distorting the default geometry by custom angles.

The cyanide ligand will have a 30^o distortion along the polar direction which can be specified using the value 30/0 in the corresponding `Angle` field on the table. In Figure 9, top right, we can see that cyanide is bent by 30^o. Similarly, the ammonia ligand that is connected to point 4 will be distorted by an azimuthal angle of 20^o as can be seen in Figure 9, bottom right.

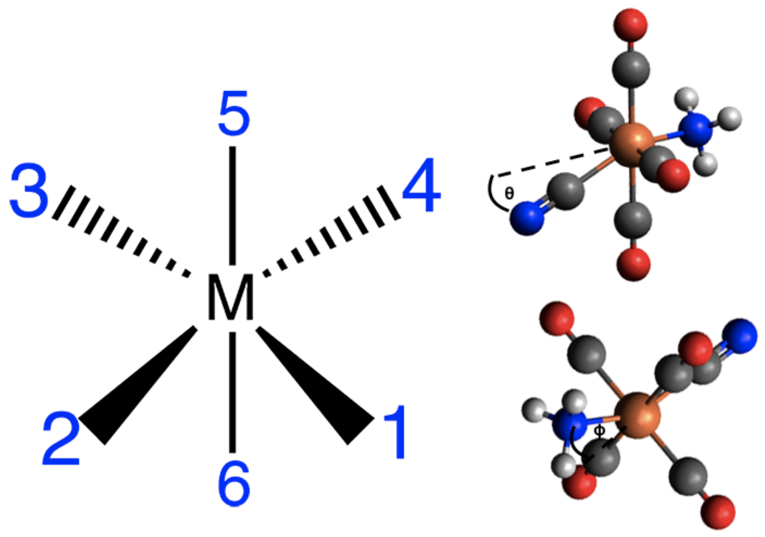


Figure 9: Two examples of distorted coordination geometries. On the left is the default octahedral geometry, on the top right the cyanide ligand is distorted by a polar angle of 30^o, while on the bottom right an ammonia ligand is distorted by an azimuthal angle of 20^o.

4.6 Building custom structures

The program is able to generate structures using non-standard cores. If the user specifies a core that is not a single metal center, then the coordination and geometry specified will be ignored and a custom structure will be built instead. Any structure can be used as core, on which additional ligands will be attached. For example, we are going to generate a complex using an iron porphyrin as our core structure and functionalize it with imidazole rings in 4 points (Figure 10).

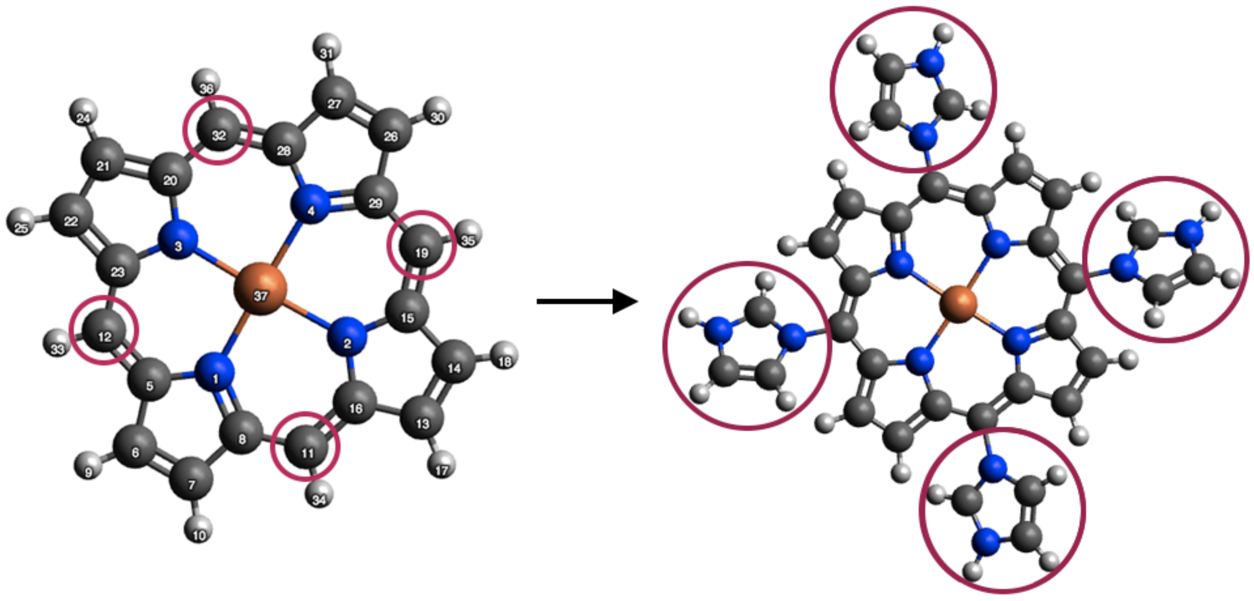
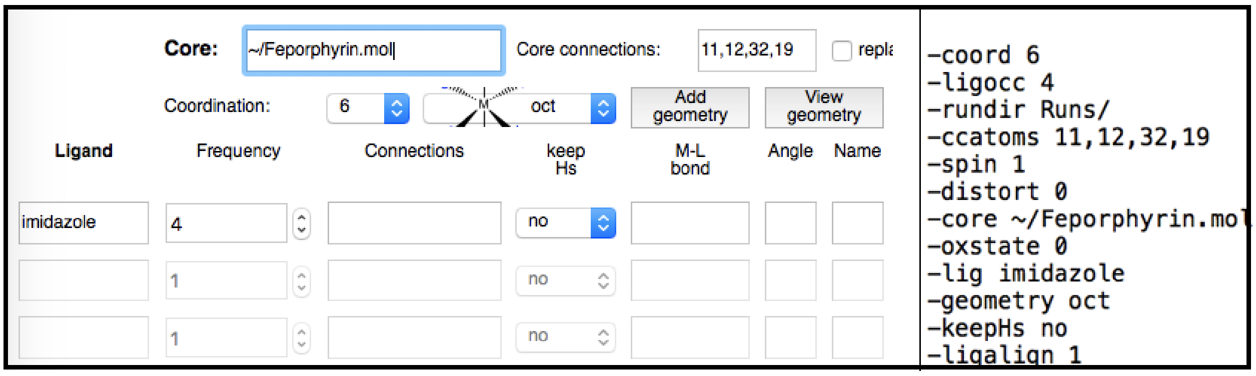


Figure 10: Custom core functionalization.

First, the user needs to specify the iron porphyrin as the core which in this case is done through a mol file. Then the user needs to specify the indices of the atoms on the core where the ligands are going to be attached. In this case, we select 4 carbon atoms with indices 11,12,19 and 32 and specify them in the `Core connections` field. Then, we need to specify again the ligands we are going to use (in this case 4 copies of the standard imidazole molecule). This input is specified on the top left of the GUI (Figure 11).
The same input can be specified in an input file using the corresponding keywords. The keyword `core` specifies the custom core, but this time we also need the keyword `ccatoms` to specify the atoms on the core that are going to be connected with the ligands. The rest of the input is similar to the simple coordination complexes case. Once the generation is complete, a new porphyrin structure will be generated with 4 imidazole rings (Figure 10).

4.7 Replacing existing ligands

Another feature that molSimplify offers is editing existing structures by replacing existing ligands with new ones. In order to enable this feature the user needs to enable it by clicking the `replace` checkbox on the GUI or specify the `replig` option in the input file.


Core: ~/Feporphyrin.mol Core connections: 11,12,32,19 ☐ repl

Coordination: 6 oct

Ligand	Frequency	Connections	keep Hs	M-L bond	Angle	Name
imidazole	4		no			
	1		no			
	1		no			

```
-coord 6
-ligocc 4
-rundir Runs/
-ccatoms 11,12,32,19
-spin 1
-distort 0
-core ~/Feporphyrin.mol
-oxstate 0
-lig imidazole
-geometry oct
-keepHs no
-ligalign 1
```

Figure 11: Input options for custom core functionalization.

As an example we are going to take the functionalized iron porphyrin that we generated in the previous example and replace 2 of the imidazole rings with benzene (Figure 12).

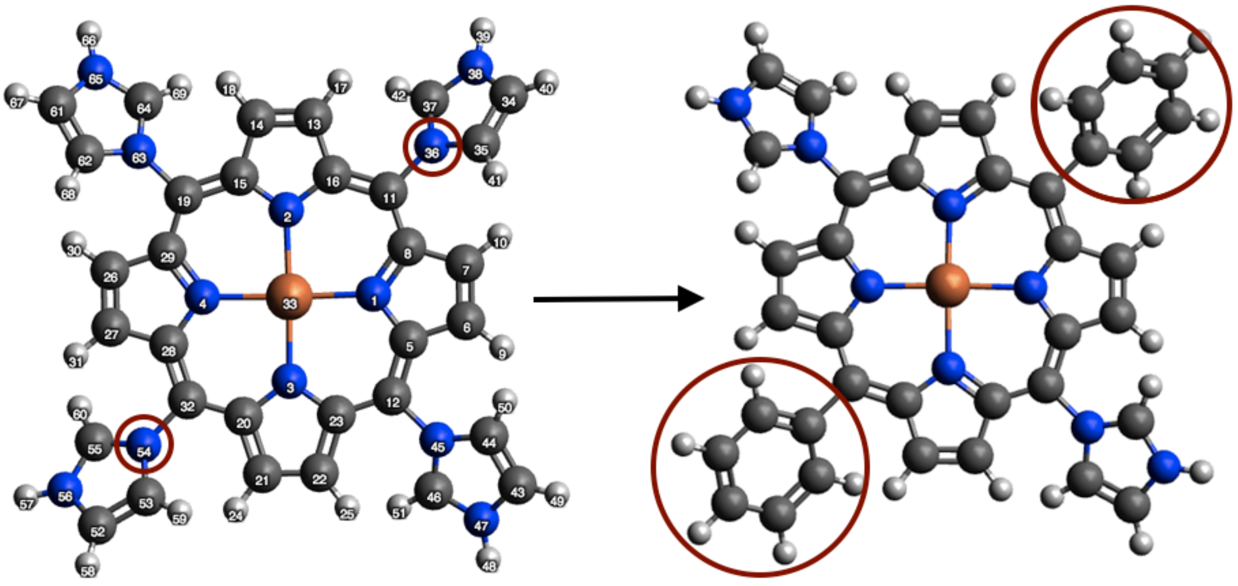
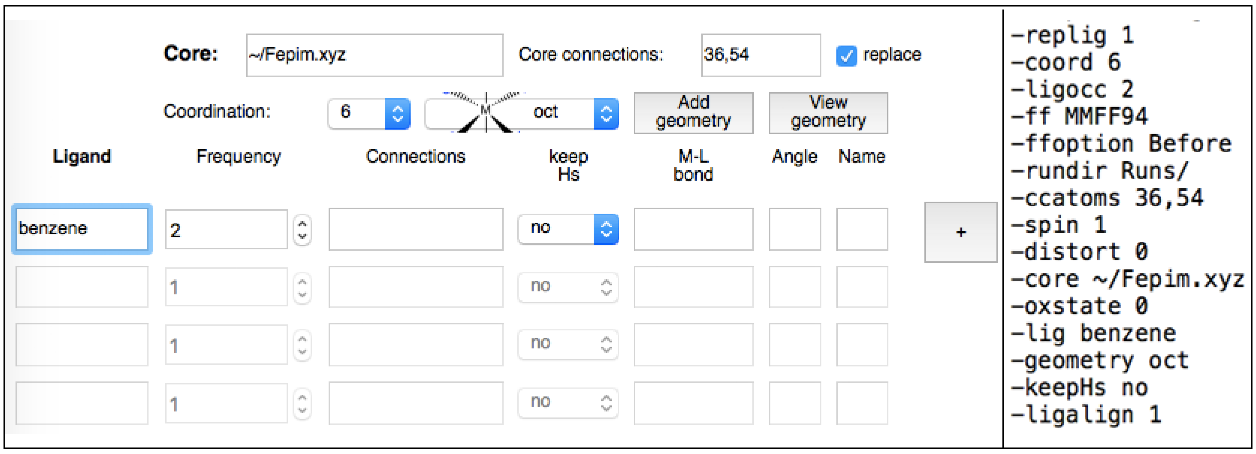


Figure 12: Custom core ligand replacement.

Initially, we need to specify the functionalized porphyrin as our new core structure using the xyz file that we generated earlier. In this case though, as connection atoms the user needs to specify the connection atoms on the **ligands** that are going to be replaced and are connected to the metal and not the atoms on the old core that are going to receive the ligands. In the case of imidazole the connections are the nitrogen atoms that are bonded with the carbons on the porphyrin, namely atoms 36 and 54. Then, the new ligands that are going to replace the existing ones are specified in the ligand table. These options are specified again on the GUI or the input file directly (Figure 13).



Core: ~/Fepim.xyz Core connections: 36,54 ✓ replace

Coordination: 6 oct

Ligand	Frequency	Connections	keep Hs	M-L bond	Angle	Name
benzene	2		no			
	1		no			
	1		no			
	1		no			

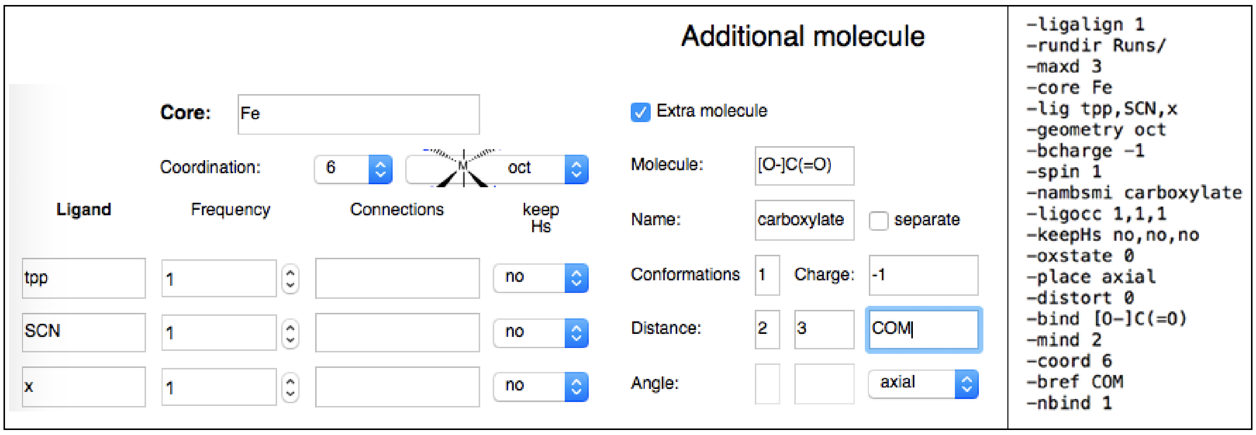
```
-replig 1
-coord 6
-ligocc 2
-ff MMFF94
-ffoption Before
-rundir Runs/
-ccatoms 36,54
-spin 1
-distort 0
-core ~/Fepim.xyz
-oxstate 0
-lig benzene
-geometry oct
-keepHs no
-ligalign 1
```

Figure 13: Input options for custom core ligand replacement.

The program loops over the specified connection points, removes the old ligands and replaces them with the new ones for a new final structure (Figure 12). Note that the force field optimization options described earlier (see section 4.3) can be used in custom core building or editing as well.

4.8 Additional molecule placement

In order to study intermolecular interactions, molSimplify is able to build supramolecular complexes by placing additional molecules around the initially generated structure, a feature denoted as `Additional` or `Extra molecule` placement.



Core: Fe

Coordination: 6 oct

Ligand	Frequency	Connections	keep Hs
tpp	1		no
SCN	1		no
x	1		no

Additional molecule

✓ Extra molecule

Molecule: [O-]C(=O)

Name: carboxylate ☐ separate

Conformations: 1 Charge: -1

Distance: 2 3 COM

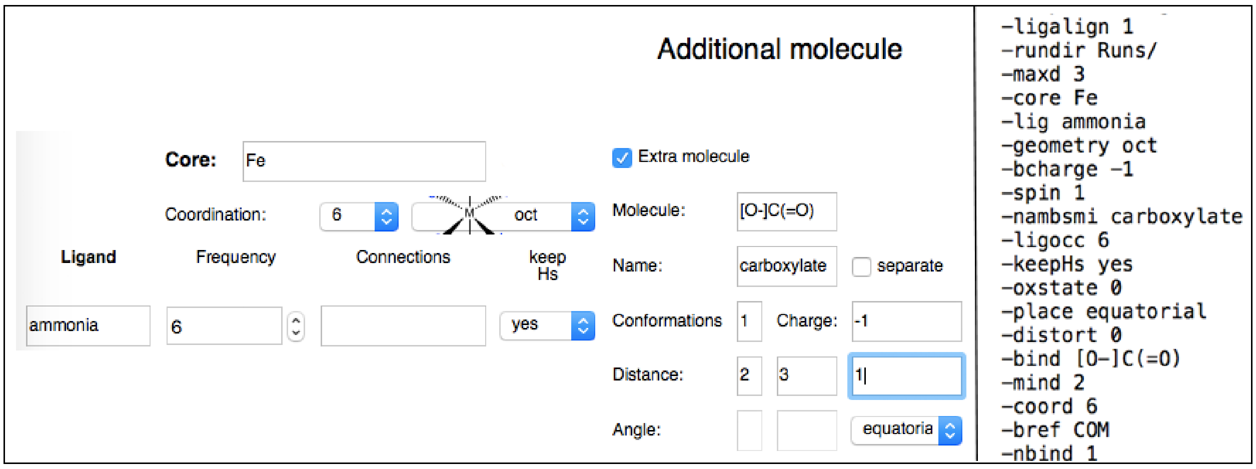
Angle: axial

```
-ligalign 1
-rundir Runs/
-maxd 3
-core Fe
-lig tpp,SCN,x
-geometry oct
-bcharge -1
-spin 1
-nambsmi carboxylate
-ligocc 1,1,1
-keepHs no,no,no
-oxstate 0
-place axial
-distort 0
-bind [O-]C(=O)
-mind 2
-coord 6
-bref COM
-nbind 1
```

Figure 14: Input parameters for additional molecule placement.

As an example we are going to place a simple carboxylate (formate) around an octahedral porphyrin structure in order to calculate the binding energy of the carboxylate on

the core structure (Figures 15-16). Initially the user specifies the base structure, namely a tetraphenylporphyrin (tpp) with an iron center, a thiocyanate (SCN) ligand on the first axial position and an empty site denoted as the x ligand on the other axial position. We include an empty site here so that the additional molecule can be placed close to that point and then bind on that site.
Once the base structure is specified according to sections 4.1-4.2, we need to tell the program where to place the additional molecule and what kind of molecule that is. This is done on the `Additional molecule` block on the GUI (top right) by enabling the `Extra molecule` input. In the `Molecule` field we can again specify a predefined molecule from the local database, a SMILES string or a file path to an xyz or mol file. The corresponding keyword in the input file is `-bind`. Next we select a name for the complex (carboxylate in this case, keyword `-nambsmi` in the input file). The flag `separate` can be used to separate the two complexes in the QChem input file for running super-position error corrections for example. Next we select how many copies of the system we want in the `Conformations` field (keyword `-nbind`). Because by default the program uses random placement of the extra molecule we might want to explore different ways of binding and by specifying multiple conformations, different placements of the extra molecule will be generated. Then we can specify the charge of the complex in the corresponding field (keyword `-bcharge` in the input file).


Additional molecule

Core:	Fe		
Coordination:	6	oct	
Ligand	Frequency	Connections	keep Hs
ammonia	6		yes

☑ Extra molecule

Molecule:	[O-]C(=O)		
Name:	carboxylate	☐ separate	
Conformations	1	Charge:	-1
Distance:	2	3	1
Angle:			equatoria

```
-ligalign 1
-rundir Runs/
-maxd 3
-core Fe
-lig ammonia
-geometry oct
-bcharge -1
-spin 1
-nambsmi carboxylate
-ligocc 6
-keepHs yes
-oxstate 0
-place equatorial
-distort 0
-bind [O-]C(=O)
-mind 2
-coord 6
-bref COM
-nbind 1
```

Figure 15: Input parameters for additional molecule placement.

In the next line we can specify the distance of the additional complex relative to the base structure. The first field corresponds to the minimum distance in Angstroms while the second to the maximum distance (keywords `-mind` and `-maxd` respectively). The last field (COM in this case, denoting Center of Mass) tells the program the reference point on the additional for the distance calculation (keyword `-bref`). In this case the center of mass of the extra molecule will be placed between 2 and 3 Angstroms from the base structure. Other

options are atomic indices or element symbols (e.g. specifying `C` here would place the extra molecule in such a way that the carbon on the formate would be between 2 and 3 Angstroms from the base structure). In the last row of the input, we can specify the orientation of the placement by selecting either a set of angles (azimuthal angle (0-180^o) and polar angle (0-360^o) with respect to the base structure (keywords `-bphi` and `-btheta` respectively) or by selecting axial or equatorial placement (keyword `-place`). In this case, because we want the carboxylate to be placed on top of the porphyrin, the axial option was selected. The final placement is shown in Figure 16 (left).
As an additional example, we can place the same formate molecule on the equatorial position of a symmetric iron-ammonia complex by changing the corresponding flag in the `Angle` row (Figure 15).

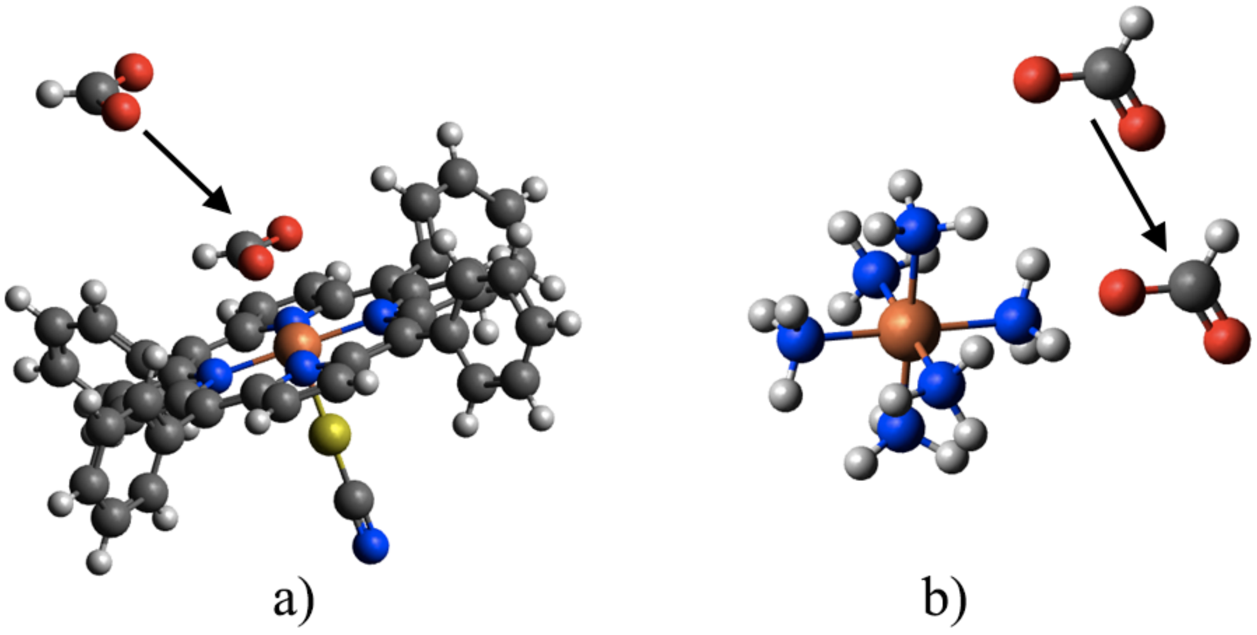
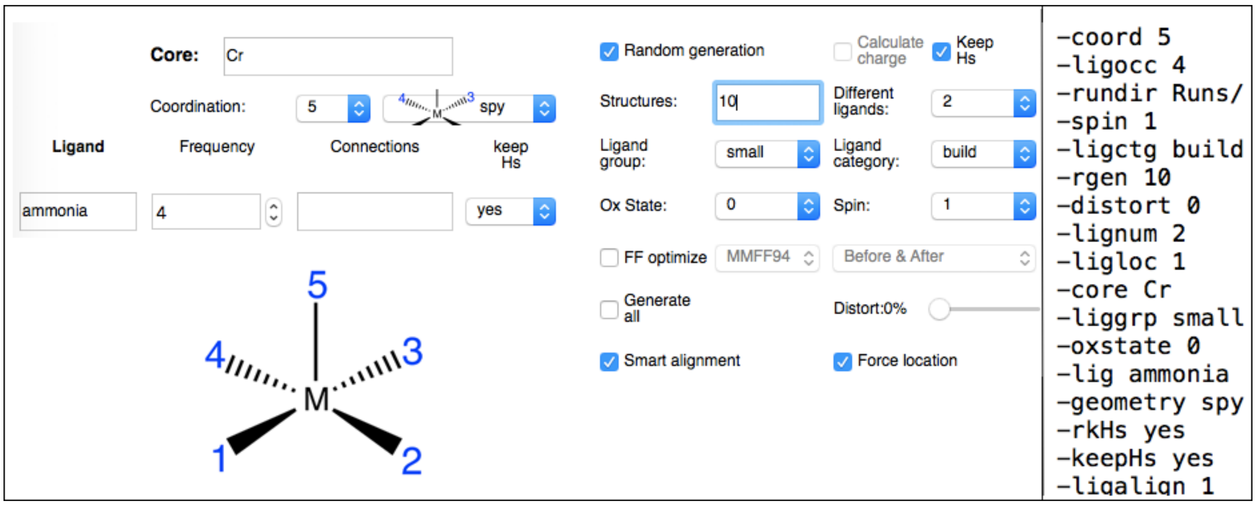


Figure 16: Examples of extra molecule placement.

When the additional molecule option is selected, a minimum of 3 xyz files will be generated, one with the base structure only, one with the additional molecule only and one with the combined structure.

5 Random generation module

One of the most useful features of molSimplify for novel structure generation is the random generation module. By enabling this feature, the user is able to perform a constrained random generation of a big number of complexes. In this module, the user specifies a limited number of input parameters and the program randomly selects the rest. As an example, we will generate a set of 10 different square pyramidal structures with chromium core, 4 ammonia ligands on the planar sites and 10 random ligands on the axial position (Figure 17).



Core: Cr
Coordination: 5 spy
Ligand: ammonia; Frequency: 4; keep Hs: yes
Random generation ✓; Keep Hs ✓
Structures: 10; Different ligands: 2
Ligand group: small; Ligand category: build
Ox State: 0; Spin: 1
Smart alignment ✓; Force location ✓

```
-coord 5
-ligocc 4
-rundir Runs/
-spin 1
-ligctg build
-rgen 10
-distort 0
-lignum 2
-ligloc 1
-core Cr
-liggrp small
-oxstate 0
-lig ammonia
-geometry spy
-rkHs yes
-keepHs yes
-ligalign 1
```

Figure 17: Input options for custom core ligand replacement.

In order to do that, we first specify the constraints, namely the chromium core and the 4 ammonia ligands following the same procedure as for normal structure generation. Next, in the random generation block we specify the number of different structures that we want in the `Structures` field (keyword `-rgen` in the input file) and then the number of different ligands that we want in the final structure (field `Different ligands` or keyword `-lignum` in the input file). In this case we already have ammonia as the first ligand and we want one additional one, therefore the total number of different ligands is only 2. Next we can limit the search for random ligands by selecting ligands from specific groups or categories using the corresponding options on the GUI or the keywords `-liggrp` and `-ligctg` respectively. Finally, we need to turn the force location option on to ensure that the random ligand will be placed on the axial position, otherwise the program might try to change the ligand locations. Note that we have selected to keep Hydrogens on the random ligands as well (keyword `-rkHs` in the input file).
Once the structure generation is complete, a set of xyz files will be generated each corresponding to a different structure (Figure 18) that conforms with the constraints imposed by the user.
As an additional example, we will generate a set of random octahedral structures with iron center. In this case, we don't specify any ligands, thus all of them will be randomly generated. We just specify the core of the structure (iron), the number of random structures that we want and the number of different ligands that we want (4 different ligands here).

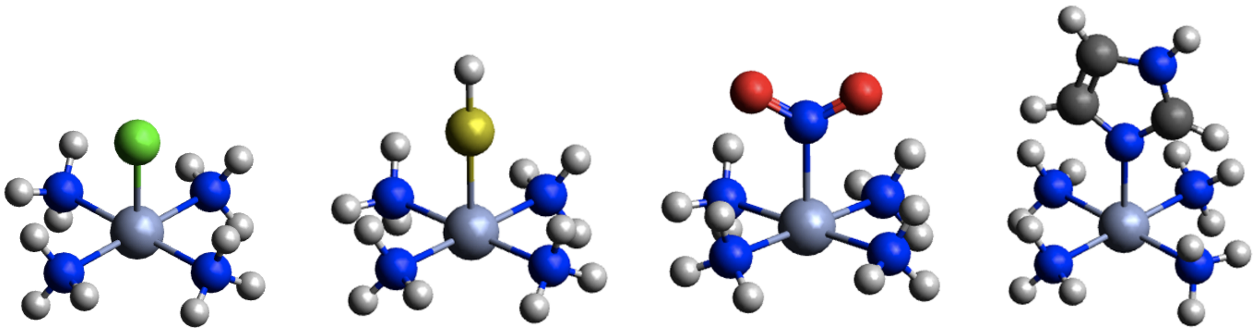
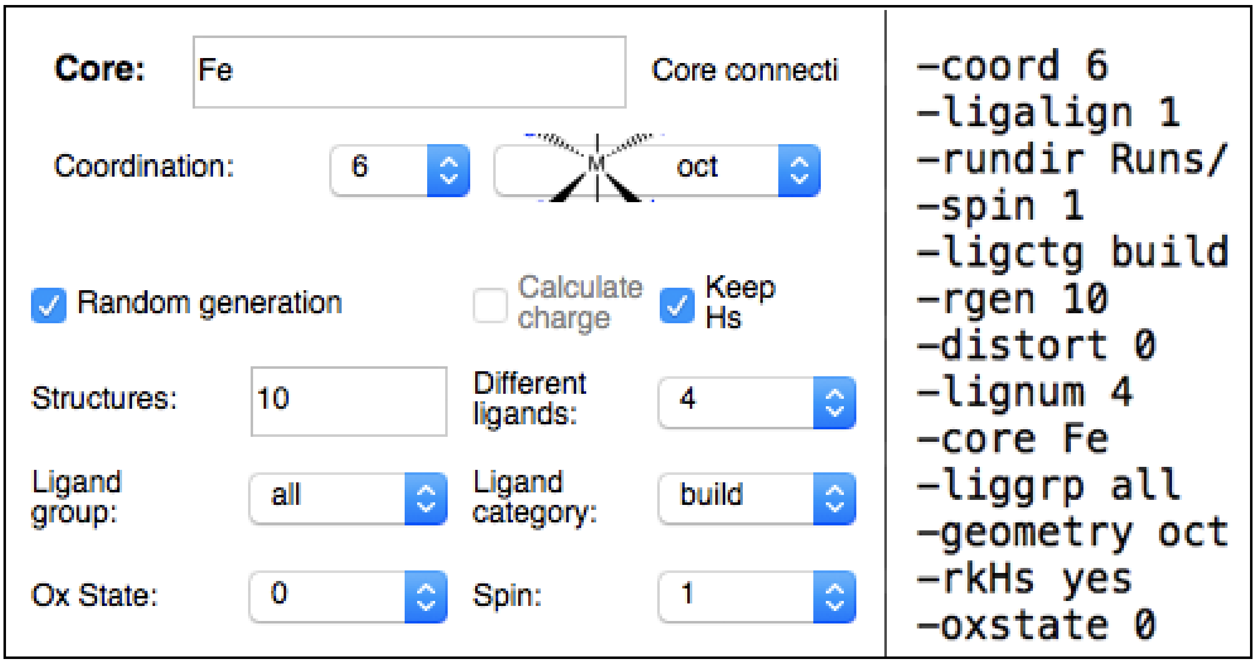


Figure 18: Input options for constrained random generation.



Core: Fe
Core connecti
Coordination: 6 oct
✓ Random generation
Calculate charge
✓ Keep Hs
Structures: 10
Different ligands: 4
Ligand group: all
Ligand category: build
Ox State: 0
Spin: 1

```
-coord 6
-ligalign 1
-rundir Runs/
-spin 1
-ligctg build
-rgen 10
-distort 0
-lignum 4
-core Fe
-liggrp all
-geometry oct
-rkHs yes
-oxstate 0
```

Figure 19: Example structures obtained from the random structure generation.

Two example output structures are shown in Figure 20, each with 4 different ligands. The first one has a chlorine, a carbonyl, two imidazole rings and two acetonitrile ligands. The second one has a bipyridine, a benzene ring, 2 hydroxyls and a nitroso ion.

6 Input files & jobscripts

In addition to generating the structures, molSimplify is able to generate input files for Quantum Chemistry (QC) calculations and jobscripts for submitting jobs in supercomputer queues. Currently, molSimplify supports 3 QC programs: TeraChem, GAMESS-US, QChem and 2 queuing systems, SGE and SLURM.

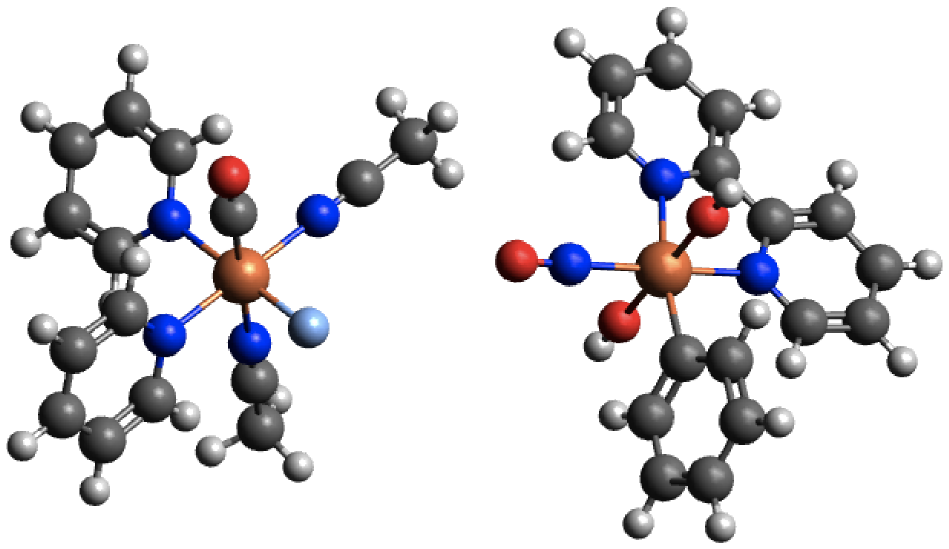
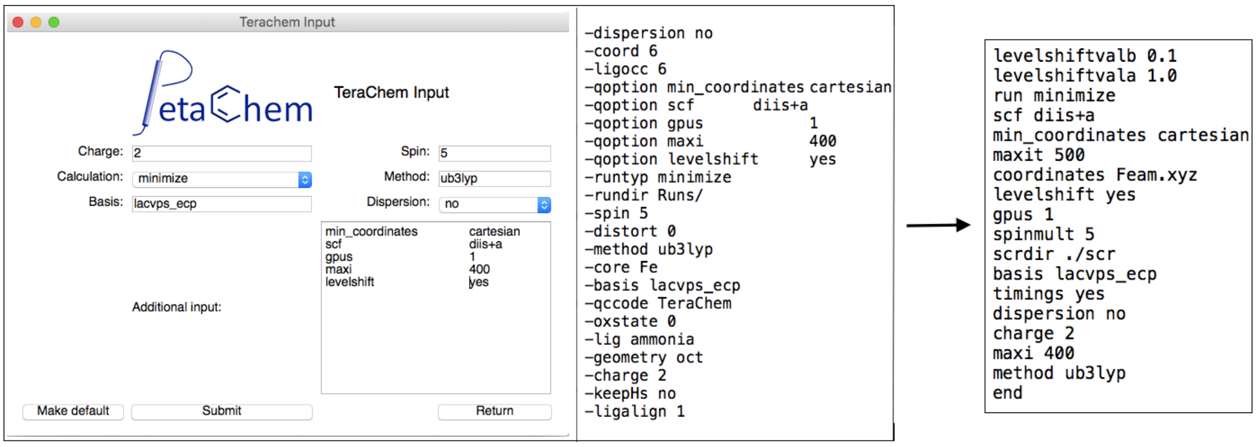


Figure 20: Example structures obtained from the random structure generation.

6.1 Input file generation

In order to generate QC input files, the corresponding checkbox should be selected on the GUI or the keyword `-qccode` specified in the input file. In this example, we will generate an input file for the TeraChem software (procedure for the other two QC programs is similar). Once we select the desired QC program we can bring forward on the GUI the corresponding window and start entering the input options. Basic options include the charge of the system (keyword `-charge` in the input file), the spin multiplicity (keyword `-spin`), the type of calculation (keyword `-runtyp`), the method (keyword `-method`), basis set (keyword `-basis`) and whether we want dispersion or not (keyword `-dispersion`). Additional options can be specified one line at a time with in the `Additional input` editor (keyword `-qoption` in the input file). For the charges, the program supports automatic charge calculation (keyword `-calccharge`) which is based on openbabel's default interpretation of charges. However, custom charges can be specified for the molecules using a mol file.
The final input file produced by the program is shown in Figure 21. Note that the program supports multiple methods, spins and charges, so if the user specifies for example for the method `ub3lyp,pbe` and charges `2,3` molSimplify will create 4 different folders each with one method and one charge.
Furthermore, in order to avoid entering similar input parameters every time the user wants to generate input files, there is a `Make default` option on the GUI that will overwrite the default options with the current ones.



```
Charge: 2                    Spin: 5
Calculation: minimize        Method: ub3lyp
Basis: lacvps_ecp            Dispersion: no

min_coordinates    cartesian
scf                diis+a
gpus               1
maxi               400
levelshift         yes
```

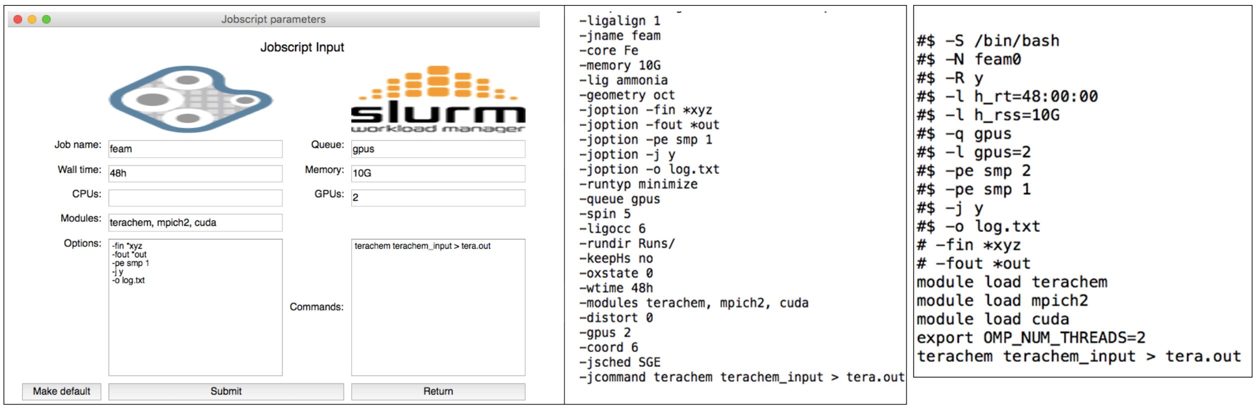
```
-dispersion no
-coord 6
-ligocc 6
-qoption min_coordinates cartesian
-qoption scf       diis+a
-qoption gpus             1
-qoption maxi             400
-qoption levelshift       yes
-runtyp minimize
-rundir Runs/
-spin 5
-distort 0
-method ub3lyp
-core Fe
-basis lacvps_ecp
-qccode TeraChem
-oxstate 0
-lig ammonia
-geometry oct
-charge 2
-keepHs no
-ligalign 1
```

```
levelshiftvalb 0.1
levelshiftvala 1.0
run minimize
scf diis+a
min_coordinates cartesian
maxit 500
coordinates Feam.xyz
levelshift yes
gpus 1
spinmult 5
scrdir ./scr
basis lacvps_ecp
timings yes
dispersion no
charge 2
maxi 400
method ub3lyp
end
```

Figure 21: Parameters for generation of QC input files.

6.2 Jobscript generation

In addition to the input files, molSimplify can generate jobscripts for supercomputer queues. Tow supercomputer queuing systems are currently supported, SGE and SLURM.
On the GUI the user can enable the jobscript generation, select the target queuing system and bring up the corresponding window. Standard options here include the job name base (keyword `-jname`), queue name (keyword `-queue`), wall time requested (keyword `-wtime`), memory (keyword `-memory`), number of cpu or gpus (keywords `-cpus`,`-gpus`) and the modules to be loaded (keyword `modules`). Additional options can be specified in the `Options` field or using the keyword `-joption` in the input file. Also extra commands for the jobscript can be specified in the corresponding field `Commands` or using the `jcommand` keyword in the input file.



```
Job name: feam                         Queue: gpus
Wall time: 48h                         Memory: 10G
CPUs:                                  GPUs: 2
Modules: terachem, mpich2, cuda
Options:                               Commands:
-fin *xyz                              terachem terachem_input > tera.out
-fout *out
-pe smp 1
-j y
-o log.txt
```

```
-ligalign 1
-jname feam
-core Fe
-memory 10G
-lig ammonia
-geometry oct
-joption -fin *xyz
-joption -fout *out
-joption -pe smp 1
-joption -j y
-joption -o log.txt
-runtyp minimize
-queue gpus
-spin 5
-ligocc 6
-rundir Runs/
-keepHs no
-oxstate 0
-wtime 48h
-modules terachem, mpich2, cuda
-distort 0
-gpus 2
-coord 6
-jsched SGE
-jcommand terachem terachem_input > tera.out
```

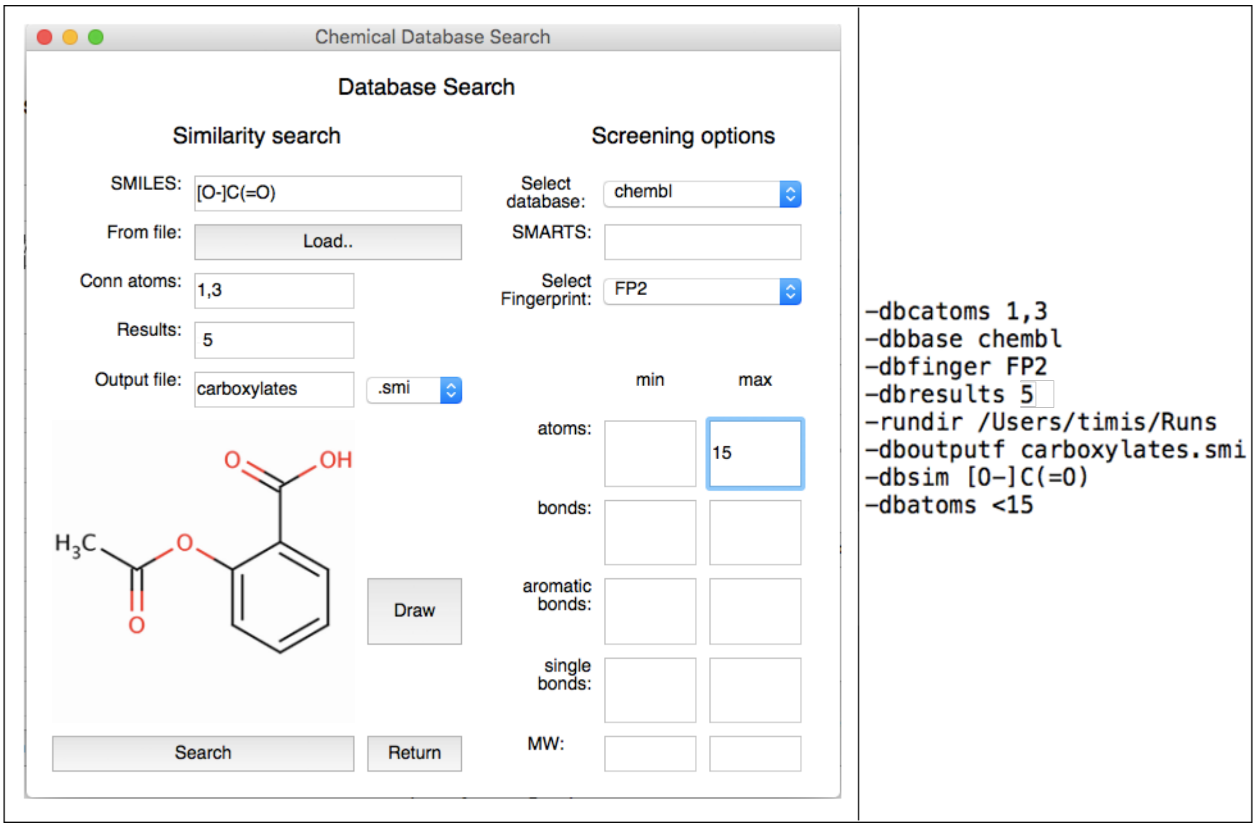
```
#$ -S /bin/bash
#$ -N feam0
#$ -R y
#$ -l h_rt=48:00:00
#$ -l h_rss=10G
#$ -q gpus
#$ -l gpus=2
#$ -pe smp 2
#$ -pe smp 1
#$ -j y
#$ -o log.txt
# -fin *xyz
# -fout *out
module load terachem
module load mpich2
module load cuda
export OMP_NUM_THREADS=2
terachem terachem_input > tera.out
```

Figure 22: Parameters for jobscript generation.

Figure 22 shows the jobscript generated using the specified input on the GUI. Furthermore, in order to avoid entering similar input parameters every time the user wants to generate jobscripts, there is a `Make default` option on the GUI that will overwrite the default options with the current ones.

7 Similarity search & screening

An additional module, complementary to the structure generation is the similarity search and screening of chemical databases. The program is able to search for molecules in chemical databases and use the results to build sets of structures.
The similarity search can be enabled by selecting the `Search DB` option on the GUI and bringing up the Chemical Database Search window (Figure 23).



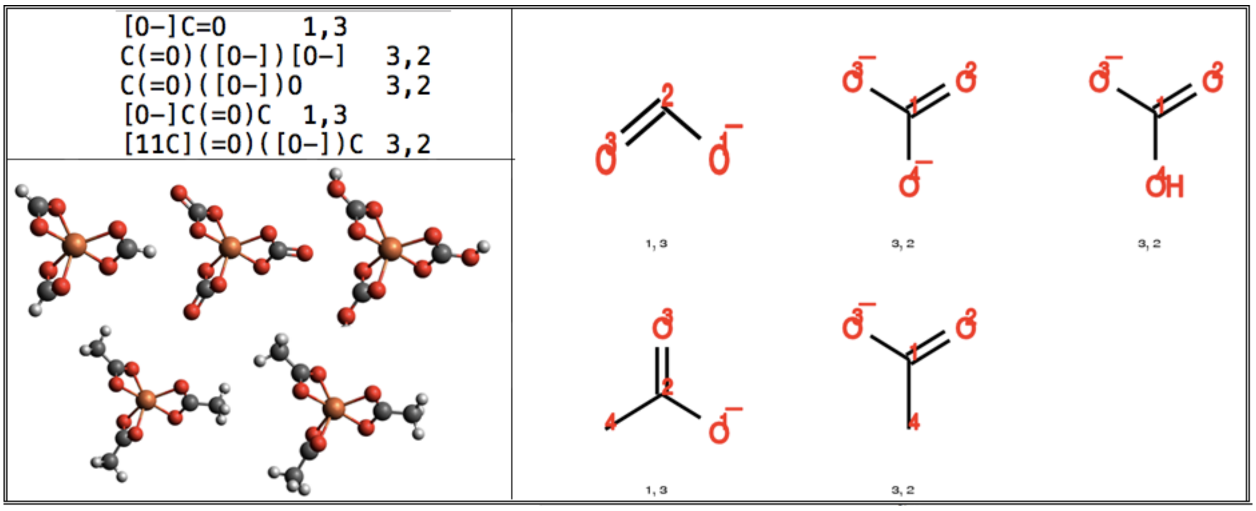
SMILES: [O-]C(=O)
Conn atoms: 1,3
Results: 5
Output file: carboxylates .smi
Select database: chembl
Select Fingerprint: FP2
atoms max: 15

```
-dbcatoms 1,3
-dbbase chembl
-dbfinger FP2
-dbresults 5
-rundir /Users/timis/Runs
-dboutputf carboxylates.smi
-dbsim [O-]C(=O)
-dbatoms <15
```

Figure 23: Input parameters for Chemical Database search.

In this example, we will use the database search to look for carboxylates. To search for similar structures in the `SMILES` field of the GUI (keyword `-dbsim`) we specify the string for a carboxylate structure. Alternatively, we can search for similar structures using an input file (xyz, mol or smi) and loading the file. Because, we want to use the results from the screening for structure generation, we specify the connection atoms on the carboxylate

SMILES string (1 and 3 corresponding to the two oxygens in this case) and the program will automatically calculate the connection atoms on the screening results. The input file keyword for the connection atoms is `-dbcatoms`. The we select the number of carboxylate molecules that we want (keyword `-dbresults`) and the output file name (keyword `-dboutputf`). In addition or simultaneously to similarity search, we can directly screen the database for specific structures using a SMARTS string (keyword `-dbsmarts`). For example, if someone wants to find structures similar to carboxylates that contain at leaste on double carbon carbon bond, the SMARTS string `C=C` should be specified in the corresponding field.
In any case we need to select the database we will use (keyword `-dbbase`). For screening, we can select various parameters, first the fingerprint used in the similarity search (keyword `-dbfinger`) and then options for the minimum and maximum number of atoms (keyword `-dbatoms`), the minimum and maximum number of bonds (keyword `-dbbonds`), the minimum and maximum number of aromatic bonds (keyword `-dbarbonds`), the minimum and maximum number of single bonds (keyword `-dbsbonds`) or the minimum and maximum molecular weight (keyword `-dbmw`).



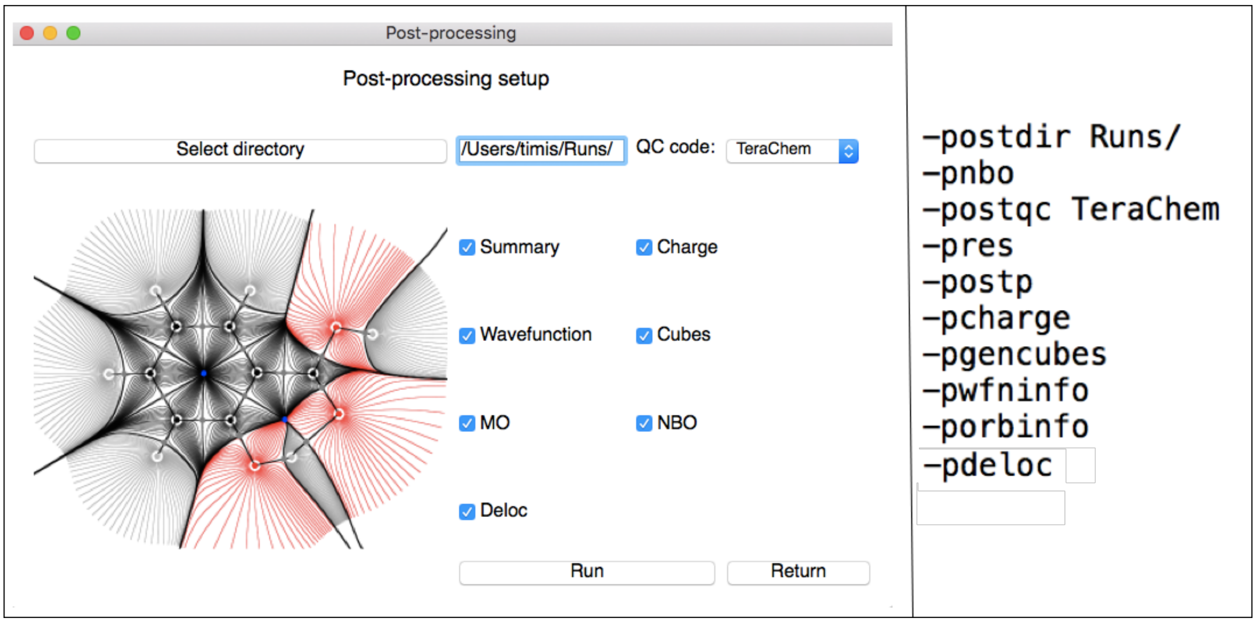
```
[O-]C=O      1,3
C(=O)([O-])[O-]   3,2
C(=O)([O-])O      3,2
[O-]C(=O)C  1,3
[11C](=O)([O-])C  3,2
```

Figure 24: Results of Database search and structure generation using those results.

Once the similarity search is concluded, an output file with the SMILES strings of the corresponding results will be generated under the running directory (Figure 24, top left). Note that because complexes are filtered to avoid salts and duplicates, the final number of structures might be smaller than the one specified in the beginning. The results of the search can be visualized using the `Draw` option (Figure 24, right). Furthermore, the resulting SMILES file can be used as an input for structure generation. If the user specifies that file as a ligand, the program will loop over all results and generate one structure for each entry (Figure 24, bottom left).

8 Post processing

An additional feature of molSimplify is post processing of Quantum Chemistry output. The parsing and processing of the output files is done using custom routines embedded in the program and using the external software Multiwfn.
To enable the post processing routines, the corresponding option must be selected from the GUI and it will bring up the setup window (keyword `-postp`). The available options for post-processing start by selecting the directory of the output files (keyword `-postdir`) and the QC code that was used to run the jobs (keyword `-postqc`). Note that your output files should have a .out extension in order for the program to be able to find it.



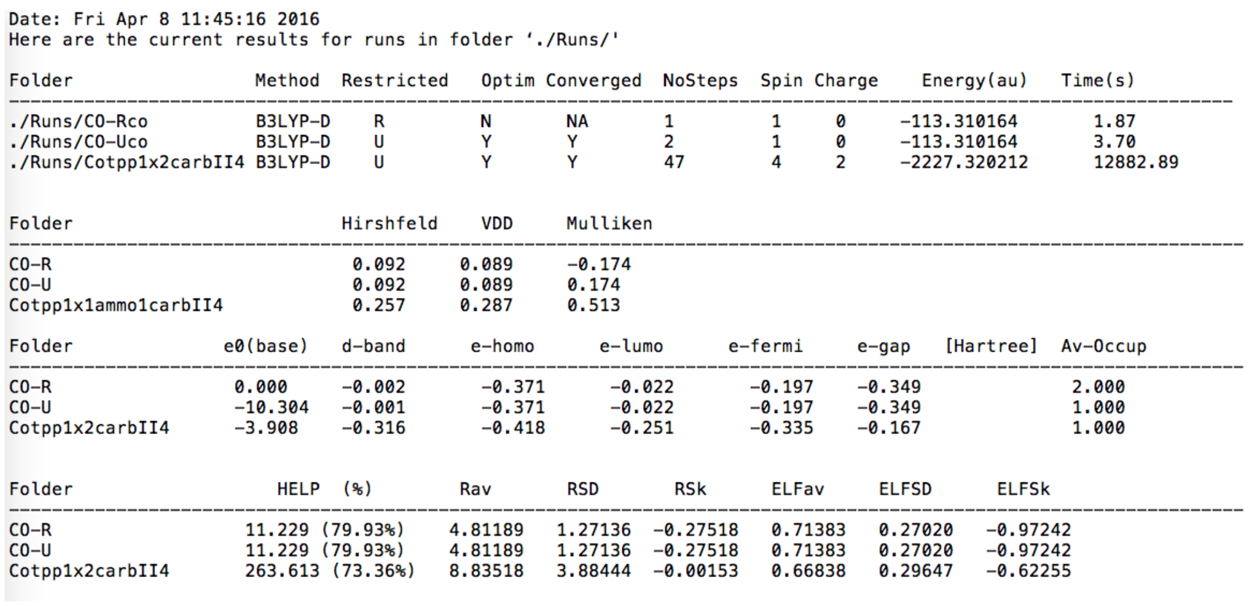
/Users/timis/Runs/ QC code: TeraChem

✓ Summary ✓ Charge
✓ Wavefunction ✓ Cubes
✓ MO ✓ NBO
✓ Deloc

```
-postdir Runs/
-pnbo
-postqc TeraChem
-pres
-postp
-pcharge
-pgencubes
-pwfninfo
-porbinfo
-pdeloc
```

Figure 25: Input options for post-processing.

The parsing and processing options include a summary of the results (keyword `-pres`) that contains information about the jobs such as energies, charges, runtime or spin state. Next, charges can be calculated (keyword `-pcharge`) using Multiwfn (currently supporting VDD, Mulliken and Hirshfeld charges). The wavefunction can be directly processed to get information such as the average electron distance, the variance of the electron positions, the HELP value or the average ELF (keyword `-pwfninfo`). The program is able to generate cube files for further processing (keyword `-pgencubes`) for the density, the spin density, α and β electron densities and the ELF. Information about the molecular orbitals (HOMO, LUMO, Fermi energy, d-band center) can be generated as well (keyword `-porbinfo`) as well as parsing of NBO data (keyword `-pnbo`). Finally the localization and delocalization indices of the metal can be calculated as well using QTAIM (keyword `-pdeloc`).



```
Date: Fri Apr 8 11:45:16 2016
Here are the current results for runs in folder './Runs/'

Folder                  Method   Restricted   Optim Converged  NoSteps  Spin Charge    Energy(au)    Time(s)
---------------------------------------------------------------------------------------------------------------
./Runs/CO-Rco           B3LYP-D    R           N       NA       1         1     0     -113.310164       1.87
./Runs/CO-Uco           B3LYP-D    U           Y       Y        2         1     0     -113.310164       3.70
./Runs/Cotpp1x2carbII4  B3LYP-D    U           Y       Y        47        4     2     -2227.320212      12882.89


Folder                          Hirshfeld    VDD     Mulliken
----------------------------------------------------------------------------------------------------------------
CO-R                             0.092     0.089     -0.174
CO-U                             0.092     0.089     0.174
Cotpp1x1ammo1carbII4             0.257     0.287     0.513

Folder                e0(base)   d-band      e-homo      e-lumo       e-fermi     e-gap   [Hartree]  Av-Occup
----------------------------------------------------------------------------------------------------------------
CO-R                   0.000     -0.002      -0.371      -0.022       -0.197     -0.349               2.000
CO-U                   -10.304   -0.001      -0.371      -0.022       -0.197     -0.349               1.000
Cotpp1x2carbII4        -3.908    -0.316      -0.418      -0.251       -0.335     -0.167               1.000


Folder                     HELP  (%)          Rav        RSD        RSk        ELFav      ELFSD      ELFSk
----------------------------------------------------------------------------------------------------------------
CO-R                     11.229 (79.93%)   4.81189   1.27136  -0.27518   0.71383   0.27020   -0.97242
CO-U                     11.229 (79.93%)   4.81189   1.27136  -0.27518   0.71383   0.27020   -0.97242
Cotpp1x2carbII4          263.613 (73.36%)  8.83518   3.88444  -0.00153   0.66838   0.29647   -0.62255
```

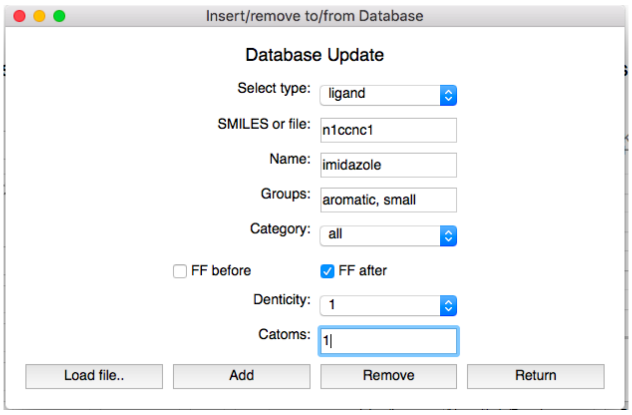
Figure 26: Example output summary files from the post processing module.

Various files are generated during the post-processing, each placed in a folder corresponding to the property being calculated. The summary of the results is placed on the top jobs directory. An example of some summary files are included in Figure 26.

9 Extras

9.1 Updating the database

The local database can be updated by adding or removing molecules. The window for updating the database is enabled using the `Add to local DB` button on the GUI (Figure 27).
Updating the database starts by selecting what type of structure you want to add/remove (ligand, core or binding/extra molecule). Then the SMILES string of the molecule or the path to an xyz/mol file is specified. The `Load file` feature can be used as well to load a molecular file. Then a name is specified for the new molecule. If the molecule is a ligand the corresponding groups and categories are specified and then the denticity and the connection atoms of the ligand. Also the options for force field optimization can be specified here. If a force field optimization option is deselected, the ligand will be blacklisted from the optimization and will not be included in any optimization when building structures.
A structure can be removed by specifying its name and then clicking the remove button. If someone wants to manually add/edit molecules, these are located in the installation directory as specified in the ∼/`.molSimplify` file under Cores, Ligands and Bind.

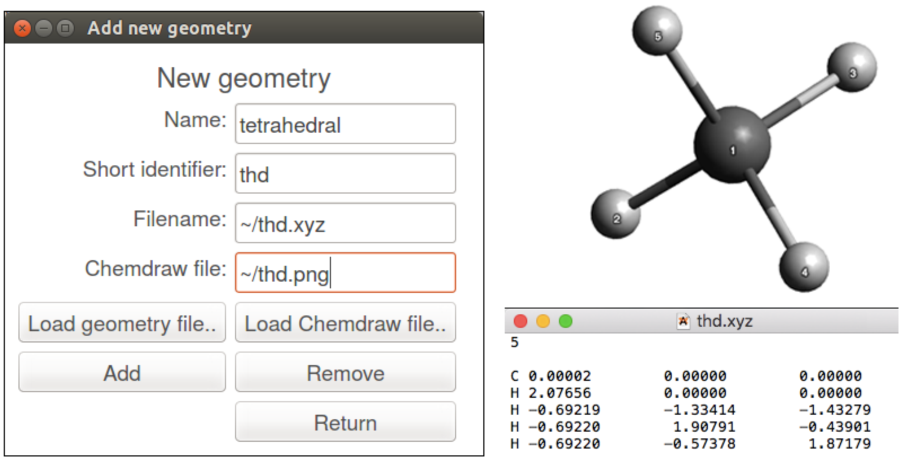


Select type: ligand
SMILES or file: n1ccnc1
Name: imidazole
Groups: aromatic, small
Category: all
FF before [✓ FF after]
Denticity: 1
Catoms: 1

Figure 27: Window for updating the local molecule database.

9.2 Add geometry

New coordination geometries can be added using the corresponding interface on the GUI. The window for updating the geometries is enabled using the `Add geometry` button on the GUI.



Name: tetrahedral
Short identifier: thd
Filename: ~/thd.xyz
Chemdraw file: ~/thd.png

```
5

C 0.00002       0.00000       0.00000
H 2.07656       0.00000       0.00000
H -0.69219      -1.33414      -1.43279
H -0.69220       1.90791      -0.43901
H -0.69220      -0.57378       1.87179
```

Figure 28: Window for updating the coordination geometries.

In order to add a new coordination geometry, the user needs to specify a name for the geometry and a short identifier (max 4 letters). Then an xyz file should be specified that will define the geometry. This file should include the minimum number of atoms that completely define the geometry with the metal center being the 1st atom in the file. An example is included in Figure 28 where a tetrahedral geometry is added. Finally a png file that shows a 2D Chemdraw of the geometry along with the atomic labels can be optionally included (see the default geometries on the GUI for examples).

To manually add/edit geometries these are located in the installation directory as specified in the ∼/.molSimplify file under Data and the png files under icons/geoms.

10 Command line options

A full list of the command line options used in the input files or directly as additional arguments is included here:

Keyword	***Description***	***Default value***
General parameters		
`-help`	show help message	-
`-i`	input file	-
`-rundir`	directory for jobs	/home/user/Runs
`-jobdir`	directory name for this specific job (relative to -rundir)	(based on complex)
`-name`	name for this specific complex (relative to -jobdir)	(based on complex)[2]
`-suff`	suffix for jobs folder	-
`-genall`	generate complex both with and without FF opt.	`False`

[2]In the general case, output will be found under rundir/jobdir/name.xyx

Structure generation parameters		
`-core`	core structure	-
`-ccatoms`	custom core connection atom(s) indices	-
`-replig`	flag for modify/replace feature	`False`
`-geometry`	coordination geometry	`oct` (octahedral)
`-coord`	coordination number	`6`
`-lig`	ligand structure name, SMILES or file path	-
`-ligocc`	frequency of corresponding ligands	`1`
`-keepHs`	do not remove Hydrogens from ligand	`False`
`-smicat`	custom ligand connection atom(s) indices	-
`-sminame`	custom ligand name	-
`-ligloc`	force location of ligands on the template	`False`
`-ligalign`	smart alignment of ligands	`True`
`-MLbonds`	custom M-L bond length for ligand (A)	`database value`
`-rgen`	number of random generated molecules	`1`
`-lignum`	number of different ligands in random generation	-
`-liggrp`	ligand group for random generation	`all`
`-ligctg`	ligand category for random generation	`all`
`-rkHs`	keep Hydrogens for random generation	`False`
`-ff`	select force field for FF optimization	`MMFF94`
`-ffoption`	select when to perform FF optimization	`Before & After`
`-distort`	randomly distort backbone by x%	`0%`
`-langles`	custom angles (polar, azimuthal) for ligand	`0/0`

Extra molecules parameters		
-bind	extra molecule name, SMILES or file	False
-bcharge	binding species charge	0
-bph	azimuthal angle phi for binding species	random
-bref	reference atoms for placement of extra molecules	COM
-bsep	flag for separating extra molecule in input or xyz file	False
-btheta	polar angle theta for binding species	random
-place	azimuthal angle for binding species relative to core	random
-bindnum	number of binding species copies for random placement	1
-nambsmi	name of custom extra molecule	-
-maxd	maximum distance for molecule placement (A)	0.0
-mind	minimum distance for molecule placement (A)	0.0
-oxstate	oxidation state of the metal	0

Jobscript parameters		
-jsched	job scheduling system	SGE
-jname	jobs main identifier	-
-memory	memory reserved per thread for job file in G	2G
-wtime	wall time requested in hours for queueing system	168h
-queue	queue name	gpus
-gpus	number of GPUS	1
-cpus	number of CPUs	1
-modules	modules to be loaded for the calculation	-
-joption	additional options for jobscript	-
-jcommand	additional commands for jobscript	-

Post-processing parameters		
-postp	post process results	-
-postqc	quantum chemistry code used	TeraChem
-postdir	directory with results	/home/user/Runs
-pres	generate calculations summary	False
-pdeninfo	calculate average properties for electron density	False
-pcharge	calculate charges	False
-pgencubes	generate cubefiles	False
-pwfninfo	get information about wavefunction	False
-pdeloc	get delocalization and localization indices	False
-porbinfo	get information about MO	False
-pnbo	post process nbo analysis	False

Quantum chemistry input

-qccode	quantum chemistry code	TeraChem
-charge	charge for system	0
-calccharge	flag to calculate charge	False
-spin	spin multiplicity for system	1
-runtyp	run type	optimize
-method	electronic structure method	ub3lyp
-basis	basis for terachem or qchem job	lacvp*
-dispersion	dispersion forces	False
-qoption	extra arguments for TeraChem in syntax	-
-exchange	exchange in qchem job	b3lyp
-correlation	correlation in qchem job	none
-remoption	extra arguments for qchem $rem block	-
-unrestricted	unrestricted calculation	True
-gbasis	GBASIS option in GAMESS	6
-ngauss	NGAUSS option in GAMESS	N31
-npfunc	NPFUNC option for diffuse functions in GAMESS	-
-ndfunc	NDFUNC option for diffuse functions in GAMESS	-
-sysoption	extra arguments for $SYSTEM GAMESS block	-
-ctrloption	extra arguments for $CONTRL GAMESS block	-
-scfoption	extra arguments for $SCF GAMESS block	-
-statoption	extra arguments for $STATPT GAMESS block	-

Database search input

-dbsim	SMILES/ligand/file for similarity search	-
-dbcatoms	connection atoms for similarity search	-
-dbresults	how many results for similary search or screening	-
-dboutputf	output file for search results	simres.smi
-dbbase	database for search	-
-dbsmarts	SMARTS string for screening	-
-dbfinger	fingerprint for similarity search	-
-dbatoms	number of atoms to be used in screening	-
-dbbonds	number of bonds to be used in screening	-
-dbarbonds	number of aromatic bonds to be used in screening	-
-dbsbonds	number of single bonds to be used in screening	-
-dbmw	molecular weight to be used in screening	-

Slab generation		
required input		
`-slab_gen`	Boolean, enables slab generation search	False
`-unit_cell`	.xyz file, containing a primitive until cell or slab for placement	-
`-cell_vector`	list of list of 3 floats,the lattice vector.7 If unit_cell is given, this argument must be provided	-
	–OR–	
`-cif_path`	path to cif file with unit cell	
`-duplication vector`	list of 3 int, number of times to duplicate unit cell	
	–OR–	
`-slab_size`	list of 3 floats, approximate size of slab in Å	
optional input		
`-freeze`	boolean -OR- int, in the input file, freeze atoms corresponding to bottom (or bottom n) layers	
`-expose_type`	string (atom symbol), type of atom (eg in an oxide) that should be exposed on the top layer	
`-shave_extra_layers`	int, number of additional vertical layers to shave	
`-miller_index`	list of 3 int, Miller Indices for exposed surface	

Placement on slab		
required input		
-place_on_slab	Boolean, enables placement	False
-target_molecule	.xyz file, object to place on the surface	-
-align_distance_method	one of: physisorption,chemisorption,custom, controls the distance at which the object is placed from the surface. If custom, must provide align_dist	-
-align_dist	float, distance to place object from surface, only if align_distance_method is custom	
-align_method	one of: center, staggered, alignpair, meaning centered in cell above symmetric center of surface, between two surface atoms, or full custom control	center
-object_align	string (atom symbol) or list of int, only used with alignpiar, type of atom in the object used for placement OR 1-based list of atom indicies (as they occur in geometry file) that should be aligned with surface. Not recommended to use 4+	-
optional input		
-surface_atom_type	string, atom symbol, only used with alignpiar. The type of atom on the surface to use, e.g. O if the surface is TiO_2	
–OR–		
-surface_atom_ind	int, only used with alignpiar, index of atom(s) on the surface to use, in order of the geometry file (not for use with simultaneous slab generation)	
-num_surface_atoms	int , only used with alignpiar, number of atoms on surface to align with items in object_align. Use a higher number of surface atoms (eg 2) to force attached atoms between 2 adjacent sites.	1
-num_placements	int, only used with alignpiar, number of copies of the object to place on the surface.	1
–OR–		
-coverage	float, between 0 and 1, only used with alignpiar,fraction of sites to occupy, as opposed to number of placements. Avoid high coverage for bulky objects!	
-multi_placement_centering	float, between 0 and 1, only used with alignpiar, used to control how multiple placements are spaced in the cell. 1 is maximum distance between adsorbates. 0 is biased towards the center of the surface.	1 if num_placements > 1, else 0.95

Placement on slab		
optional input, cont.		
`-control_angle`	float, angle in degrees, only used with alignpiar, only possible with a single object_align. If specified, gives the angle between the line joining the atom in object_align and the target in angle_control_partner and the axis given in angle_surface_axis	
`-angle_control_partner`	int, only used with alignpiar and control_angle, the second atom in the object with which to draw an axis for controlling angles on the surface.	
`-angle_surface_axis`	list of 2 float, x-y vector, used with alignpiar and control_angle, defines the axis in the surface plane with which the angle will be set.	
`-shave_extra_layers`	int, number of additional vertical layers to shave	
`-duplicate`	boolean, if true, mirror placement on the top and bottom of the slab.	False
`-freeze`	boolean -OR- int, in the input file, freeze atoms corresponding to bottom (or bottom n) layers	
`-shave_extra_layers`	int, number of additional vertical layers to shave	

References

[1] Efthymios I. Ioannidis, Terry Z. H. Gani, and Heather J. Kulik. molSimplify: A toolkit for automating discovery in inorganic chemistry. *J. Comp. Chem.*, 37(22):2106–2117, aug 2016.