# VM2

**Version 2.8.2**

## Free energy of binding for a host-guest series: tutorial 1

**VeraChem LLC**

Contact:

For information regarding VM2 software package licensing contact VeraChem LLC at sales@verachem.com

For technical support contact VeraChem LLC at support@verachem.com

For general enquiries contact VeraChem LLC at info@verachem.com

# VM2 Free Energy of Binding for a Host-Guest Series: Tutorial 1

## Host-guest example: Sampl6 Octa-acids and guests

## Host-guest example: Sampl6 Octa-acids and guests

This is a full example of setup, execution of calculations, and collection of binding affinity results for the host molecules octa-acid (OA) and methylated octa-acid (TEMOA) and series of eight guests (ligands), for a total of sixteen complexes. The data sets – starting SD files and experimental binding affinities - are taken from the Sampl6 challenge repository. (*1*)

**NOTE:** You will need a working installation of AmberTools with the $AMBERHOME environment variable set to carry out the full procedure as described below. Please see http://ambermd.org/ to download AmberTools and for its documentation.

To proceed, first, untar the examples file vcCompChem_2_8_2_examples.tar.bz2, which is provided with the package:

```
tar xvf vcCompChem_2_8_2_examples.tar.bz2
```

The main directory for this example is:

```
vcCompChem_2_8_2_examples/host_guest/Sampl6/oa_gaff_vcharge
```

it contains a readme file: README.sampl6.oa , which describes the overall process, stepping through the following three directories in turn

```
Sampl6/oa_gaff_vcharge/setup
Sampl6/oa_gaff_vcharge/run
Sampl6/oa_gaff_vcharge/results
```

An outline of each step now follows. You can skip the setup section by going straight to Section 2. and making use of the “-d reference” option, described in Sections 2.1.2. and 2.2.2.

### 1. Setup

The procedure starts with setup, namely structure preparation, typing, and charge assignment of the host and guest molecules. A step-by-step description of the setup process now follows. Also, see:

```
Sampl6/oa_gaff_vcharge/setup/README.setup
```

### 1.1. Host Setup

The relevant subdirectories are:

```
Sampl6/oa_gaff_vcharge/setup/hosts/source_files
Sampl6/oa_gaff_vcharge/setup/hosts/prepareHosts
```

#### 1.1.1. Source files

The /source_files directory contains .sdf, .mol2, and .pdb files for the host molecules octa-acid (OA) and methylated octa-acid (TEMOA) taken from the Sampl6 challenge repository. It also contains .mol files, derived from the .sdf files, along with a script mol_2_sdf.py to combine these .mol files into a single SD file, oa_hosts.sdf, for processing in /prepareHosts.

```
python  mol_2_sdf.py oa_hosts.sdf
```

### 1.1.2. Generate partial charges and assign parameters

Ambertools is used to assign bond, angle, torsion, and non-bonded Lennard-Jones parameters, while atom partial charges can be generated either by VeraChem's VCharge method or by AM1-BCC through AmberTools – for this example VCharge will be used. The resulting prmtop and inpcrd files are then converted to the [crd,top,mol] file set used by VM2.

The prepareLigands.pyc script (it can be used for host molecules as well as ligands) automates this process. First, go to the prepareHosts directory

```
Sampl6/oa_gaff_vcharge/setup/hosts/prepareHosts
```

then copy over the host sdf file just generated

```
cp ../source_files/oa_hosts.sdf .
```

Then, to execute the script choosing VCharge partial atomic charges type:

```
./run_prepareHosts.sh &
```

This script contains the command line:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc -charge_method
vcharge oa_hosts.sdf >& run_prepareHosts.out &
```

To assign charge using AM1-BCC instead remove the charge method argument:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc oa_hosts.sdf >&
run_prepareHosts.out &
```

You can compare your results against those in the reference subdirectories.

## 1.2. Ligand Setup

The relevant subdirectories are:

```
Sampl6/oa_gaff_vcharge/setup/ligands/source_files
Sampl6/oa_gaff_vcharge/setup/ligands/prepareLigands
```

The steps basically mirror those just described for the host molecules.

### 1.2.1. Source files

The /source_files directory contains .sdf and .mol2 files for the ligand molecules OA-G0 to OA-G7 taken from the Sampl6 challenge repository. It also contains a script combine_sdfs.py to combine the SD files into a single SD file, oa_ligands.sdf, for processing in /prepareLigands.

```
python  combine_sdfs.py oa_ligands.sdf
```

### 1.2.2. Generate partial charges and assign parameters

Ambertools is used to assign bond, angle, torsion, and non-bonded Lennard-Jones parameters, while atom partial charges can be generated either by VeraChem's VCharge method or by AM1-BCC through AmberTools – for this example VCharge will be used. The resulting prmtop and inpcrd files are then converted to the [crd,top,mol] file set used by VM2.

The prepareLigands.pyc script automates this process. First, go to the prepareLigands directory

```
Sampl6/oa_gaff_vcharge/setup/hosts/prepareLigands
```

then copy over the ligand sdf file just generated

```
cp ../source_files/oa_ligands.sdf .
```

Then, to execute the script choosing VCharge partial atomic charges type:

```
./run_prepareLigands.sh &
```

This script contains the command line:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc -charge_method
vcharge oa_ligands.sdf >& run_prepareLigands.out &
```

To assign charge using AM1-BCC instead remove the charge method argument:

```
      $VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc oa_ligands.sdf
>&    run_prepareLigands.out &
```

You can compare your results against those in the reference subdirectories.

The setup stage is now complete.

## 2. Run Calculations

The next step is to run the host-guest, host, and ligand, free energy calculations. The relevant directories and readme file are:

```
Sampl6/oa_gaff_vcharge/run/1_ligand_confgen
Sampl6/oa_gaff_vcharge/run/2_vm2_runs
Sampl6/oa_gaff_vcharge/run/README.runvm2
```

Ligand conformations can be pre-generated in /1_ligand_confgen and used to seed the VM2 calculations in /2_vm2_runs.

## 2.1. Generation of Ligand Starting Conformations

Randomly orientated conformations of the ligand are generated, which are read-in to seed the actual host-guest VM2 free energy calculations.

### 2.1.1. Example run

Go to the directory

```
run/1_ligand_confgen
```

This directory contains a python script to generate run directories for conformer generation, and a python script to run the conformer generation calculations. Example usage is as follows:

```
python build_ligand_start_conf_dirs.py
```

will first populate the directory

```
1_ligand_confgen/gen_ligand_start_confs_rndm
```

with the required subdirectories, input files, and data files to run. Then the following command

```
python run_ligand_confs_gen.py -r slurm
```

will step through all these subdirectories, generating slurm scripts, and submitting the calculations to the batch queue. See Section 2.1.3 below for additional submission options through the -r flag.

### 2.1.2. Options available for building conformer generation directories

The python script build_ligand_start_conf_dirs.py can take a number of arguments for non-default control the source of the system data etc.:

```
  -d or --data     reference        : Populate 'input_data' directory using the
                                      data in the setup 'reference' directories
                                      e.g. /setup/ligands/prepareLigands/reference,
```

| | | |
|---|---|---|
| | | and subsequently build the run directories with this data. |
| | new | : Populate 'input_data' directory using the new data in the setup directories e.g. /setup/ligands/prepareLigands, and subsequently build the run directories with this data. (Default behavior.) |
| | reuse | : Reuse the data from an already populated 'input_data' directory. |
| -c or --clear | input | : Delete the contents of 'input_data' directory. |
| | rundirs | : Delete the contents of the run directories 'gen_ligand_start_confs_rndm' and 'gen_ligand_start_confs_snap'. |
| | all | : Delete content from the 'input_data' directory and the run directories. |

Example usage:

```
python build_ligand_start_conf_dirs.py -c rundirs -d reuse
```

This will clear the contents of previously generated run directories and use the data already present in ./input_data to regenerate the run directories i.e. data will not be taken from the setup directories in this case.

### 2.1.3. Options available for running conformer generation

The python script run_ligand_confs_gen.py can take a number of arguments:

| | | |
|---|---|---|
| -r or --runscript | bsh | : Generate and use bash shell scripts for submission of each calculation. (Default behavior.) |
| | csh | : Generate and use c-shell scripts for submission of each calculation. |
| | pbs | : Generate a pbs script for submission of each calculation to a queue. |
| | slurm | : Generate a slurm script for submission of each calculation to a queue. |
| -q or --partition | 'queue name' | : For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used. |

|  |  |
|---|---|
| -p or --prepmode | : If present the run scripts are generated and placed in every directory, but the calculations are not submitted. |

## 2.2. Host-guest calculations

Two main types of VM2 host-guest free energy calculation are available. One is regular VM2, which carries out iterative rounds of conformational searching until convergence; the other type carries out geometry optimizations of host-guest conformations constructed from ligand conformers read-in and processes them for free energy. The latter is much faster, but much less exhaustive in terms of sampling conformational space. In combination, there are two ways to seed these two VM2 calculation types with ligand conformers: multiple conformers randomly orientated in space, but placed at the center of geometry of the host – see Section 2.1. above, and a single conformer, based on the geometry in which it was prepared originally, and also placed at the center of geometry of the host. This provides for four different overall VM2 calculation schemes, which cover various types of use scenarios.

### 2.2.1. Example run

Go to the directory

```
run/2_vm2_runs
```

This directory contains a python script to generate run directories for host-guest VM2 free energy calculations, and a python script to step through the directories and run the calculations. Example usage is as follows:

```
python build_vm2_run_dirs.py
```

will first populate the following four directories, which cover the calculation types described above, with the required subdirectories, input files, and data files to run.

```
/2_vm2_runs/fast_vm2_rndm
/2_vm2_runs/fast_vm2_single
/2_vm2_runs/vm2_rndm
/2_vm2_runs/vm2_single
```

**Note:** For "_rndm" types, the corresponding pre-generation of ligand conformers – Section 2.1. - must already have occurred.

Then the following command:

```
python run_vm2_calculations.py -s random -v fast -r slurm
```

will step through the subdirectories of /2_vm2_runs/fast_vm2_snap, generating slurm scripts, and submitting the calculations to the batch queue. Similarly, any of the other three calculations types may be run by setting the appropriate flags – see Section 2.2.2 below. See Section 2.2.3 below for additional submission options through the -r flag.

### 2.2.2. Options available for building VM2 directories

The python script build_vm2_run_dirs.py can take a number of arguments for non-default control of the source of the system data etc.:

| | | |
|---|---|---|
| -d or --data | reference | : Populate 'input_data' directory using the data in the setup 'reference' directories e.g. /setup/ligands/prepareLigands/reference, and the ligand start conformer generation reference directory /run/1_ligand_confgen/reference and subsequently build the run directories with this data. |
| | new | : Populate 'input_data' directory using the new data in the setup directories e.g. /setup/ligands/prepareLigands and the ligand start conformer generation directory /run/1_ligand_confgen/gen_ligand_start_confs_rndm<br><br>and subsequently build the run directories with this data. (Default behavior.) |
| | reuse | : Reuse the data from an already populated 'input_data' directory. |
| -s or --startconfs | random | : Requests run directory set up for VM2 free energy calculations where randomly oriented ligand conformers are placed at the host center of geomatry and are used to generate starting host-guest conformations. |
| | single | : Requests run directory set up for VM2 free energy calculations where a single ligand starting conformation is used based on the supplied ligand .crd file coordinates. The placement is set as the center of geometry of the host molecule. |
| | all | : Requests both types of directory to be set up. (Default behavior.) |
| -c or --clear | input | : Delete the contents of 'input_data' directory. |
| | rundirs | : Delete the contents of the run directories. |
| | all | : Delete content from the 'input_data' directory and the run directories. |
| -v or --vm2type | regular | : Requests run directory set up for regular VM2 |

|  |  |  |
|---|---|---|
|  |  | host-guest free energy calculations, which carry out extensive conformational searching. |
|  | fast | : Requests run directory set up for fast VM2 host-guest free energy calculations, which calculate free energies via geometry optimizing host-guest conformations generated from read-in ligand conformers previously generated. |
|  | all | : Requests set up for both types of VM2 calculation. |
| -k or --keyfile | 'ligand_key_filename' | : Name of text file containing the subset of ligands in the series - one on each line (see ligand_key_5.txt.) |

### 2.2.3. Options available for running VM2 calculations

The python script run_ligand_confs_gen.py can take a number of arguments:

|  |  |  |
|---|---|---|
| -s or --startconfs | random | : Requests that VM2 free energy calculations are run for the series where randomly oriented ligand conformers are placed in the active site and are used to generate starting protein-ligand conformations. (Default behavior.) |
|  | single | : Requests that VM2 free energy calculations are run for the series where a single ligand/guest conformation is placed at the host's center of geometry generating a single starting host-guest conformation. |
|  | all | : Requests both types of run be carried out. |
| -r or --runscript | bsh | : Generate and use bash shell scripts for submission of each calculation. (Default behavior.) |
|  | csh | : Generate and use c-shell scripts for submission of each calculation. |
|  | pbs | : Generate a pbs script for submission of each calculation to a queue. |
|  | slurm | : Generate a slurm script for submission of each calculation to a queue. |

```
-q or --partition     'queue name'          : For pbs and slurm run scripts, the name of the
                                              queue or partition if the default queue is not
                                              being used.

-p or --prepmode                  : If present the run scripts are generated and placed
                                    in every directory, but the calculations are not
                                    submitted.


-v or --vm2type    regular    : Requests regular VM2 protein-ligand free energy
                                    calculations for the series, which carry out
                                    extensive conformational searching.

                   fast         : Requests fast VM2 VM2 protein-ligand free energy
                                    calculations for the series, which  calculate
                                    free energies via geometry optimizing
                                    protein-ligand conformations generated from
                                    read-in ligand conformers snapped to a template
                                    scaffold. (Default behavior.)

                  all            : Requests both types of VM2 calculation are run for
                                    the series.

-g or --gpu                        : If present requests use of CUDA enabled VM2
                                     executable.

-o or --ompthreads    1          : If -g not set results in MPI parallelism only.
                                     Enforced for ligand only runs.

                      2          : If set will result in MPI+OpenMP run (8 MPI processes
                                     (default), 2 OpenMP threads per process). If -g also set
                                     will result in MPI+OpenMP+CUDA parallelism.


-m or --molsystems    complexes+ligands     |
                                            |
                       complexes+hosts      |
                                            |
                         hosts+ligand       |
                                            |
                          complexes         |----> Run subset of the moleculer system
                                            |       types.
                                            |
                            ligands         |
                                            |
                            hosts           |

                             all            : Default. Run ligands, complexes, and
                                              hosts.
```

Example usage:

```
    nohup python run_vm2_calculations.py -g -o 2
```

Run default fast-random set of calculations (fast_vm2_randm directory) with 8 MPI process calculations for ligand calculations, but MPI+OpenMP+CUDA calculations for the complexes and the hosts.

This run utilizes 8 MPI processes with 1 GPU per MPI process and 2 OpenMP threads per MPI process. It therefore requires 16 compute cores and 8 GPUs.

## 3. Results Collection

When the host-guest (ligand), host, and ligand VM2 free energy calculations for the complete ligand series have completed, the binding free energies may then be calculated, and the formatted files, e.g., .mol2, .pdb, .sdf, containing the associated molecular structures collected.

The relevant directories and readme file are:

```
Sampl6/oa_gaff_vcharge /results
Sampl6/oa_gaff_vcharge /results/conformers
Sampl6/oa_gaff_vcharge /results/README.results
```

### 3.1. Generate binding free energy spreadsheets and collect conformer files

Go to the directory

```
Sampl6/oa_gaff_vcharge /results
```

To generate spreadsheets and collect molecule conformer files for the "fast_vm2_rndm" calculations from Section 2.2.1 type:

```
python create_vm2_summaries.py -c fast_vm2_rndm  -l OA-G0
```

Requirements:

File containing experimental data:  sampl6_oa_experimental_data.txt

The filename must contain the text "experimental_data".
The format is <hostname_ligandname>, <value>  e.g.

```
OA_OA-G0, -5.68
OA_OA-G1, -4.65
OA_OA-G2, -8.38
OA_OA-G3, -5.18
OA_OA-G4, -7.11
```

:

Output spreadsheets:

```
results/OA_TEMOA _fast_vm2_rndm_complex.csv
results/OA_TEMOA _fast_vm2_snap_host.csv
results/fast_vm2_rndm_ligand.csv
results/OA_TEMOA _fast_vm2_rndm_SUMMARY.csv
```

The last of these contains the binding free energies.

Output conformer files:

For the protein, each ligand, and each host-ligand complex, formatted files (e.g. mol2, pdb, sdf, xyz) containing the lowest energy conformer, and the eight lowest energy conformers are written to:

```
results/conformers/fast_vm2_rndm/complexes
results/conformers/fast_vm2_rndm/ligands
results/conformers/fast_vm2_rndm/hosts
```

**3.2. Results generation options**

For the script create_vm2_summaries.py the following commandline argument is mandatory with the following options:

```
-c or --calctype      fast_vm2_rndm       : Identify the calculation type
                                           to collect and summarize run
                      fast_vm2_single      data for.

                      vm2_rndm

                      vm2_single
```

There are three additional non mandatory arguments:

```
-n or --receptorname         : Provide the name of the receptor
                               e.g. for this case the hosts
                              are named “OA” and “TEMOA”
                              This is useful if more than one host
                               and separate summary files are required
                               for each host or if you want the results
                               files labeled with the host name.

-l or --refligand            : Provide the name of the reference
                               ligand to be used in relative binding
                               affinity calculation i.e. for Delta(DeltaG)
                               The default is no reference.
```

| | | |
|---|---|---|
| -g or --getconfs | <number of confs> | : The number of conformers to keep in the extracted formated conformer files e.g. .sdf, .mol2, .pdb. The default is 8 plus a set of formatted files each with the lowest energy conformer. |

## References

1. A. Rizzi *et al.*, Overview of the SAMPL6 host–guest binding affinity prediction challenge. *J Comput Aided Mol Des* **32**, 937-963 (2018).