

**User’s Manual**   
**version 2.00**

**Development Version**  
  
  
Distributed by the Potoff and Schwiebert Groups  
© Wayne State University

1. Introduction (What is GOMC?)
2. How to get the software
3. Building the code
   1. Required compilers, libraries, programs (this includes psfgen and tcl which are needed to build the PSF files)
   2. Supported operating systems
   3. Serial
   4. GPU
4. Input file formats
   1. PDB, PSF, parameter, control file (define each part of the files here)
5. Output file formats
   1. Working with VMD
6. System setup
   1. Building initial coordinates with Packmol
   2. Creating a PSF file
   3. Creating the control file
7. Running the simulation
8. Intermolecular Energy and Virial function.
9. How to get help/technical support.

**Tutorial Overview**

This document will instruct a new user how to download, compile, and run the GOMC molecular simulation code. A basic understanding of statistical physics is recommended to complete this tutorial.

To demonstrate the capabilities of the code, the user is guided through the process of downloading and compiling a GOMC executable. That executable is then used to perform saturated vapor and liquid equilibria (VLE) studies on systems of pure isobutane (R600a), a branched alkane that whose application as a refrigerant/propellant is increasing.

<http://en.wikipedia.org/wiki/Isobutane>

The Transferable Potentials for Phase Equilibria (TraPPE) united atom (UA) force field is used to describe the molecular geometry constraints and the intermolecular interactions.

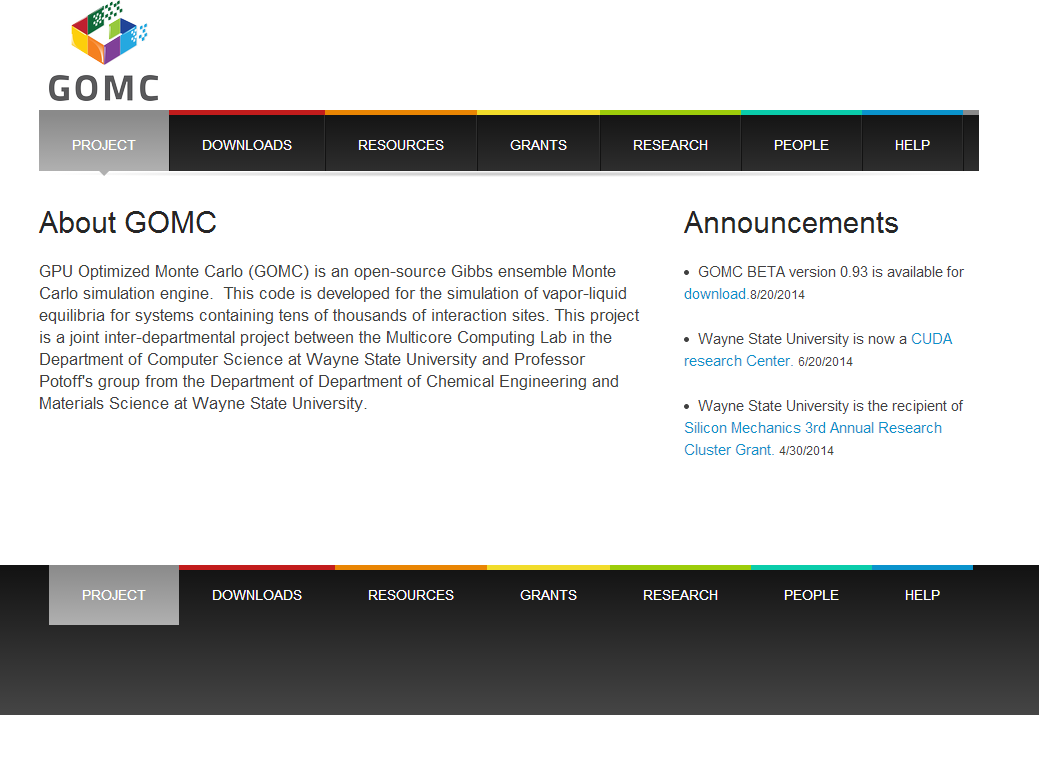
**Introduction (What is GOMC?)**

Monte Carlo (MC) simulation is a type of simulation driven by stochastic processes. "GO" stands for GPU-Optimized; this code was intended to run optimally on modern graphics process hardware.

More specifically, this engine includes serial and GPU-Optimized (multi-threaded) codes designed to run Markov chain Boltzmann sampling of chemical systems -- effectively sets of points defined by topological maps and interaction algorithms in a simulation box. From statistical mechanics, we know this is one way to sample phase space and model chemical systems.

GOMC currently supports canonical, isobaric-isothermal, Grand canonical, constant volume Gibbs ensemble, and constant pressure Gibbs ensemble simulations. GOMC employs widely-used simulation file types (PDB, CHARMM-style parameter file, PSF). GOMC includes configurational bias algorithms for both linear and branched charged, and none charged systems.

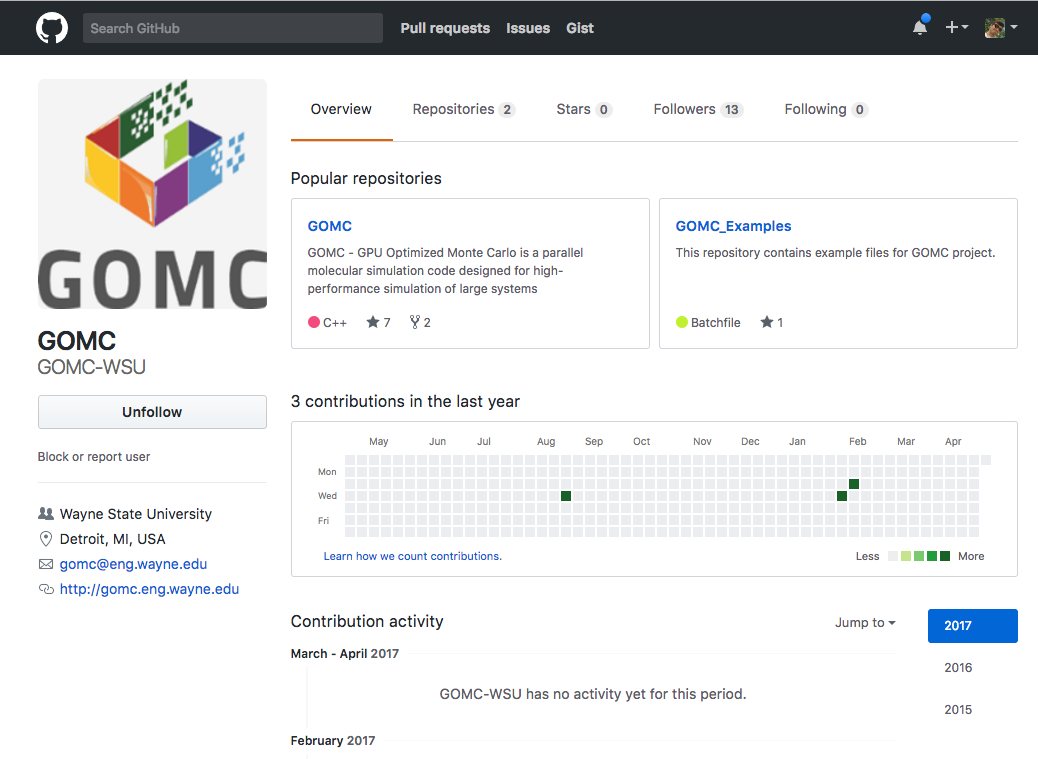
**How to get the software**The latest public code builds, project logo, manual, and other resources can be obtained via the following website:

****[gomc.eng.wayne.edu](http://gomc.eng.wayne.edu/)

The code can be found under the download tab, below and to the right of the logo. When new betas (or release builds) are announced, they will replace the prior code under the downloads tab. An announcement will be posted on the front page to notify users.

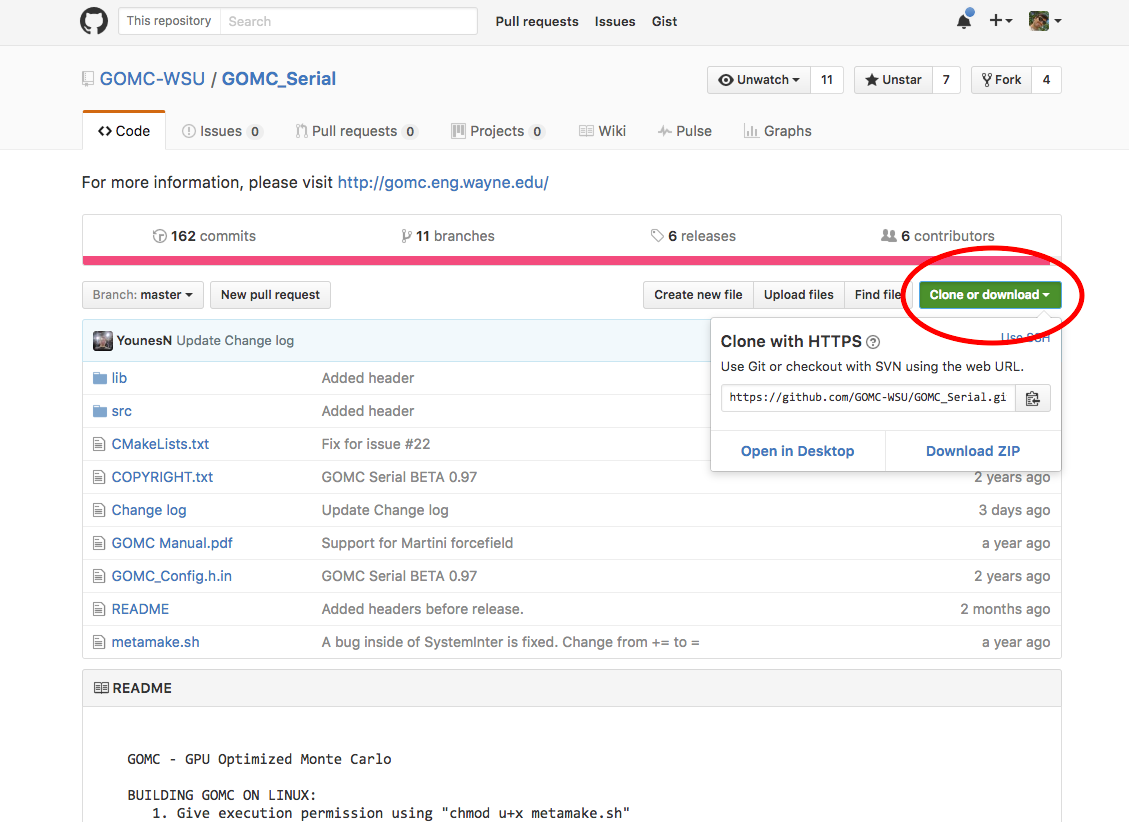
Currently, version control is handled through the GitHup repository. The posted builds in Master branch are “frozen” versions of the code that have been validated for a number of systems and ensembles. Other branches are created as a means of implementing new features. The latest updated code builds, project logo, manual, example files, and other resources can be obtained via the following GitHub repository:

https://github.com/GOMC-WSU



The CPU and GPU code are merged together under GOMC repository and can be found under the main page. In addition, Examples repository can be found under the main page. Under each repository, the code and manual can be downloaded by clicking on the Clone or download tab. For more information regarding GitHub, visit the following link:

<https://guides.github.com/activities/hello-world/>



**Platform and Software Requirements**

*Supported Operating Systems*  
  
GOMC officially supports **Windows 7**, **8**, and most modern distributions of **Linux** (see the next section). This software has the ability to compile on recent versions of **OS X**; however, such a platform is not officially supported.   
  
*Required Software Prequisites*

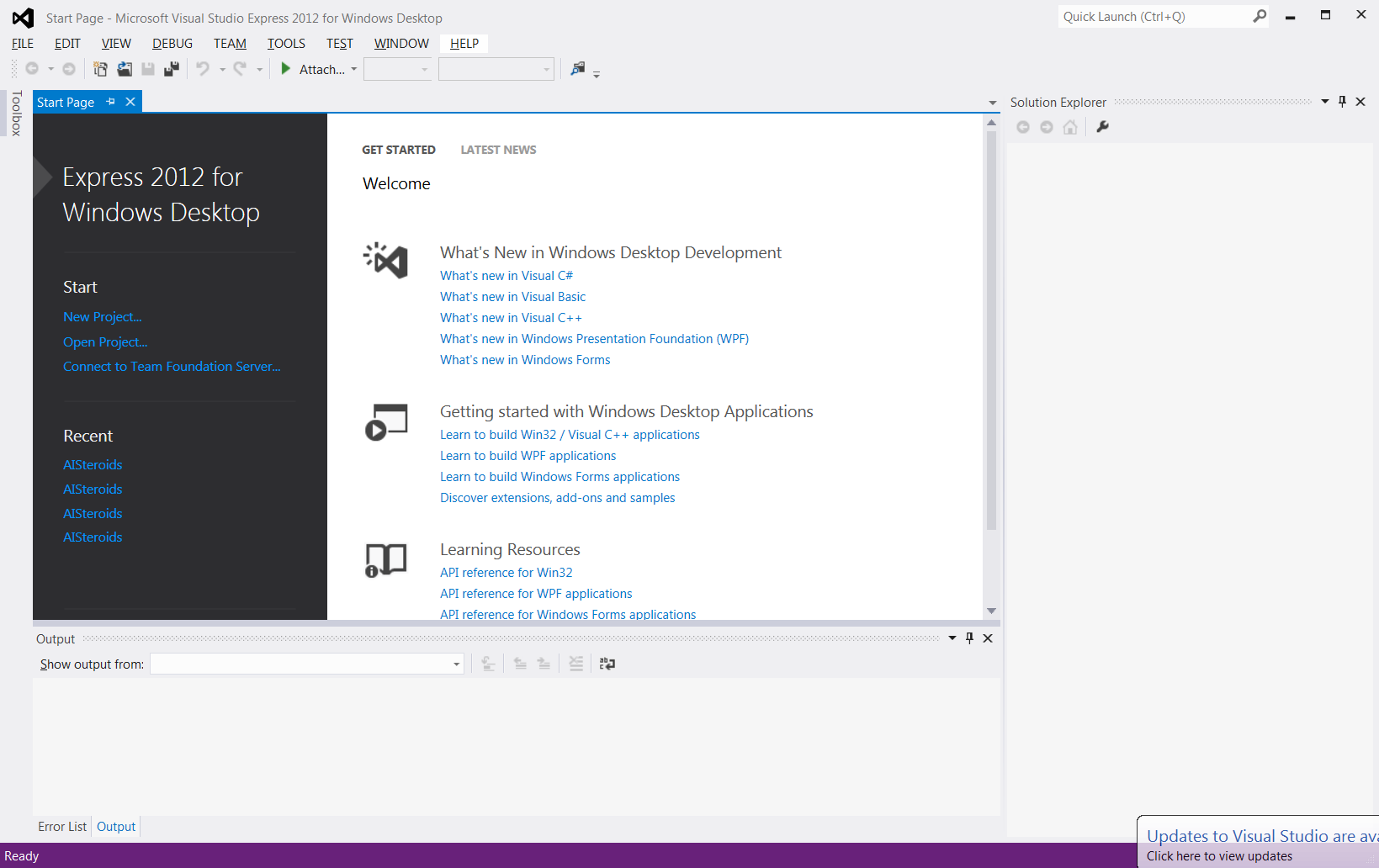
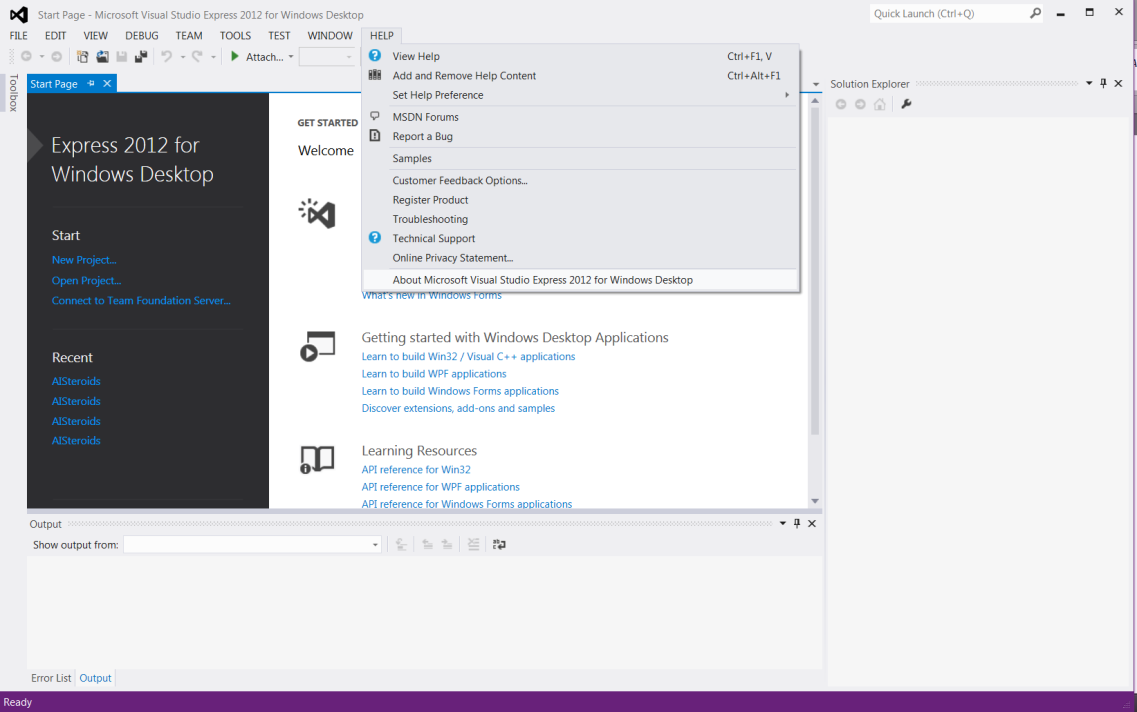
GOMC has some mild software requirements, which are widely available for Linux operating systems. Required software requirements are:

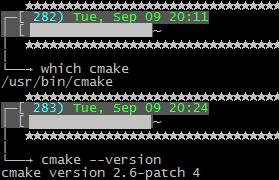
1. C++03 Compliant Compiler
   1. Linux/OS X
      1. icc (Intel c++ compiler)   
         Type the following command in a terminal.  
           
           
         If gives a version number 4.4 or later, you’re all set. If it’s older than 4.4 (released in 2009), we recommend upgrading.  
           
         In Linux, the Intel compiler will generally produce the fastest serial executables (when running on Intel Core processors).

icc --version

* + 1. g++ (GNU GCC)  
       Type the following command in a terminal.  
         
         
       If gives a version number 4.4 or later, you’re all set. If it’s older than 4.4 (released in 2009), we recommend upgrading.

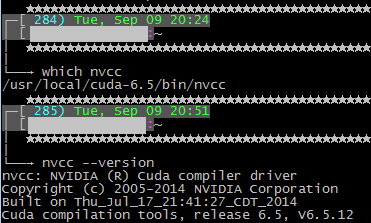
g++ --version

* 1. Windows
     1. Visual Studio  
        Microsoft’s Visual Studio 2010 or later is recommended.   
        To check the version:  
        *Help* (top tab) 🡪 *About Microsoft Visual Studio*  
          
          
        

1. cmake (if compiling on Linux)  
     
   To check if cmake is installed:  
     
     
   To check the version number:  
     
     
   Here’s an example from one of our systems:  
   

cmake --version

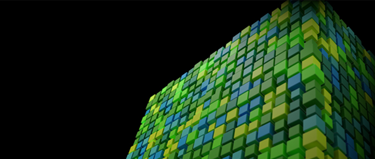
which cmake

1. nvcc/CUDA libs  
     
   The GPU builds of the code requires NVIDIA’s CUDA 6.0 or newer…  
     
   To check if nvcc is installed:  
     
   To check the version number:  
     
     
   Here’s an example from one of our systems:  
     
     
     
   CUDA is viewed as an essential requirement, but is not used to compile the serial code, which can be compiled on systems without CUDA.

which nvcc

nvcc --version

To download CUDA visit NVIDIA’s webpage:

<https://developer.nvidia.com/cuda-downloads>  
  
  
  
CUDA is required to compile the GPU executable in both Windows and Linux. Please refer to CUDA Developer webpages to select an appropriate version for the desired platform.  
  
To install CUDA in Linux *root*/*sudo,* privileges are generally required. In Windows, administrative access is required.

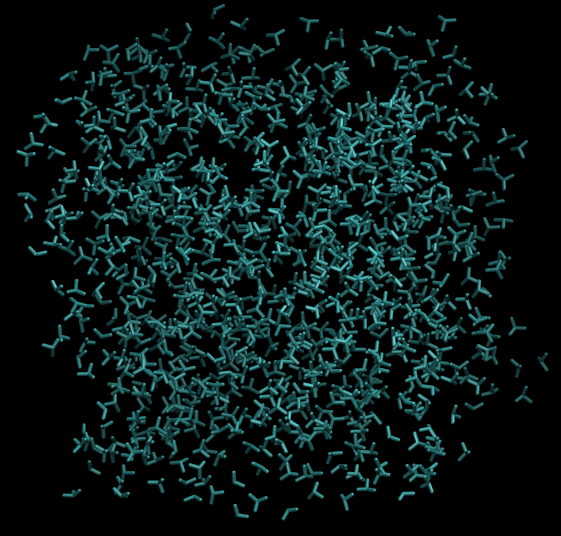
*Highly Recommended Software Tools*

**NOTE:**

The listed programs are used in this manual and are generally considered **necessary**.  
  
**VMD**  
VMD (Visual Molecular Dynamics) is a 3-D visualization and manipulation engine for molecular systems written in C-language. VMD is distributed and maintained by the University of Illinois at Urbana-Champaign. Its sources and binaries are free to download. It comes with a robust scripting engine, which is capable of running python and tcl scripts.

More info can be found here:

<http://www.ks.uiuc.edu/Research/vmd/>

Although GOMC uses the same fundamental file types – PDB (coordinates) and PSF (topology) as VMD, it uses some special tricks to obey certain rules of those file formats.  
  
  
One useful purpose of VMD is visualization of your systems.   
  
  
Figure: A system of united atom isobutane molecules

Nonetheless, the most critical part of VMD is a tool called PSFGen. PSFGen uses a tcl or python script to generate a PDB and PSF file for a system of one or more molecules. It is, perhaps, the most convenient way to generate a compliant PSF file.

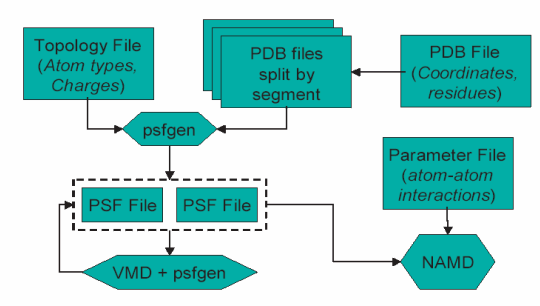


FIGURE: An overview of the PSFGen file generation process and its relationship to VMD/NAMD

To read more about PSFGen, reference:  
  
Plugin homepage @ UIUC  
<http://www.ks.uiuc.edu/Research/vmd/plugins/psfgen/>  
  
“Generating a Protein Structure File (PSF)”, part of the NAMD Tutorial from UIUC  
<http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-html/node6.html>

In-Depth Overview [PDF]  
<http://www.ks.uiuc.edu/Research/vmd/plugins/psfgen/ug.pdf>

**Packmol**Packmol is a molecule packing tool created by José Mario Martínez, a professor of mathematics at the State University of Campinas, Brazil. It is written in Fortran and is free to download. More information is available on their homepage:

<http://www.ime.unicamp.br/~martinez/packmol/>

To compile it, a Fortran language compiler is needed, such as gfortran. Many Linux distributions no longer come with Fortran compilers automatically, so this may need to be installed additionally.

Packmol allows a specified number of molecules to be packed at defined separating distances within a certain region of space. One of Packmol’s limitations is that it is unaware of topology; it treats each molecule or group of molecules as a rigid set of points.

*WARNING*  
Another more serious limitation is that it is not aware of periodic boundary conditions (PBC). As a result, when using Packmol to pack PDBs for GOMC, it is recommended to pack to a box 1 Angstroms smaller than the simulation box size. This prevents hard overlaps over the periodic boundary.

*Other Useful Software Tools*

**Grace**

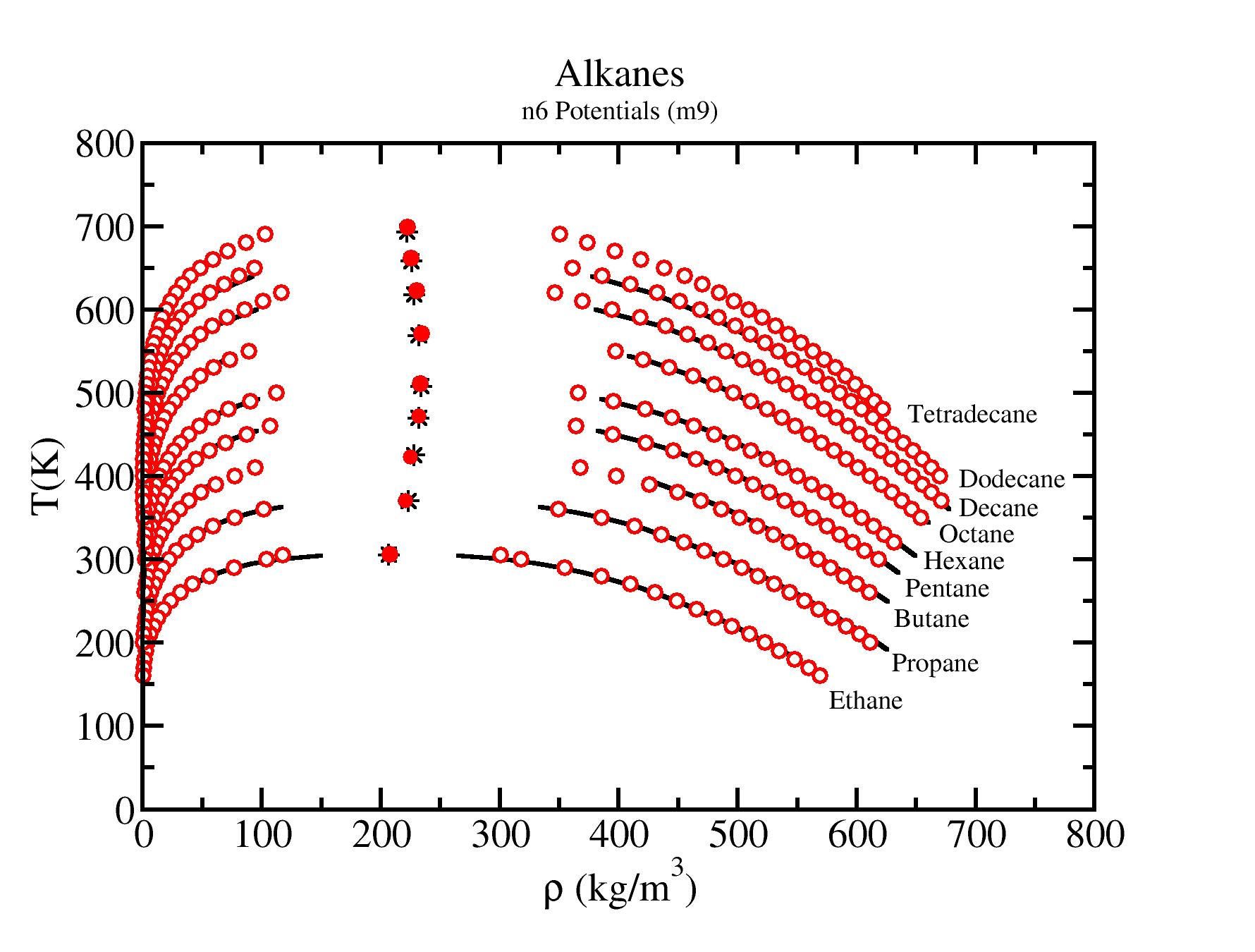
Grace is a piece of graphing software written and maintained by the Weizmann Institute of Science's Plasma Laboratory (Rehovot, Israel). Mostly used in Linux, it can also be compiled in Windows. The developers warn it may be missing some functionality.



In-depth information and the source can be found on the project page, here:

<http://plasma-gate.weizmann.ac.il/Grace/>

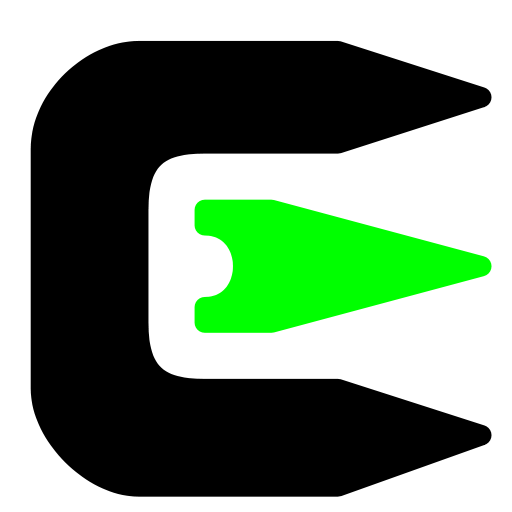
When compiled, Grace’s executable in Linux is typically named “xmgrace”. This tool allows the production of high quality, precise line and dot graphs, ideal for visualizing much of the thermodynamic data from the GOMC engine. Below is an example of the results of simulations of saturated VLE densities of linear alkanes produced with Grace.



**Cygwin**

Cygwin is one option to assist in building and visualizing systems in Windows. It provides Microsoft Windows users with a Unix-like environment and command-line interface, and offers Windows-compatible ports of common Linux applications.

<https://cygwin.com/>



The software is a free and open source, licensed under the GNU General Public License version 3. Its primary maintainers are Red Hat Inc. and NetApp. One of the most impressive abilities of Cygwin is its ability to launch a full Windows-compatible X-server Window, which allows convenient visualization of Linux app GUIs. It is compatible with the Grace graphing software. In practice, this package behaves most analogously to a Linux virtual machine in Windows.

**Compiling GOMC**

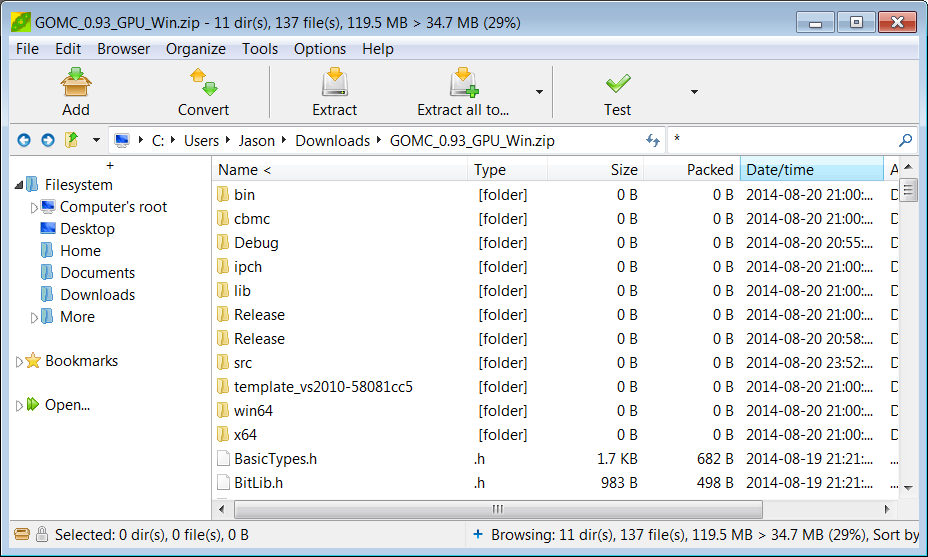
*Extracting the Code*

GOMC is distributed as a compressed folder, containing the source and build system. To compile the code after downloading it, the first step is to extract the compressed build folder.

In Windows, the folder for the GPU code is compressed using a standard \*.zip file format. To unzip simply use a utility like Peazip:

<http://peazip.sourceforge.net/>

Below is an example of what the downloaded code looks like when unzipping in Peazip.



In Linux, the GPU and Serial codes are compressed using gzip and tar (\*.tar.gz). To extract, simply move to the desire folder and type in the command line:

tar –xzvf <file name>.tar.gz

*Compiling the Code*

*GPU Code:*

Compilation on Windows:

Once the code is extracted, to compile it on Windows, you need to load the project into Visual Studio by opening the extracted folder and double clicking on the solution file of the desired Visual Studio version. After the solution is opened in Visual Studio, go to the “Build” menu and select “Build solution” to compile the code. You can compile either with release mode or with debug mode by selecting the desired mode from the “Solution Configuration” drop box.

To run the project, simply click the run button or hit F5 on the keyboard.

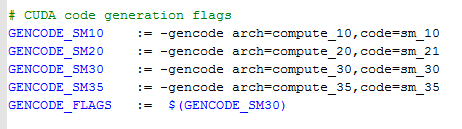
Compilation on Linux:

To compile the GPU code on Linux, go to the directory of the project, and type in the command line:

make

You can configure the “makefile” file to choose different C compilers, to select the desired compute capability, and to configure many more compilation flags.

The default compute capability is 3.0. To change the compute capability, go to the GENCODE\_FLAGS option, and set it to one of the compute capability flags that are defined in the file.



To run the program, run the executable “GOMC.out”. The system’s LD\_LIBRARY\_PATH will need to be configured to support CUDA (more on this later).

*Serial Code:*

Compilation on Windows:

See GPU “Compilation on Windows” section and follow an identical procedure for the released serial code. See README for instructions on how to use the CMake-GUI to build the configuration and solution files necessary for the Windows build.

Compilation on Linux:

In Linux, the CPU code uses a simple makefile. Enter the directory and type in the command line:

make all

This will use the Makefile to compile a GPU-compatible executable called “GOMC.out”. To run, the system’s LD\_LIBRARY\_PATH will need to be configured to support CUDA (more on this later).

For the serial code, which uses cmake for compilation, go to the base directory and type in the command line:

./metamake.sh

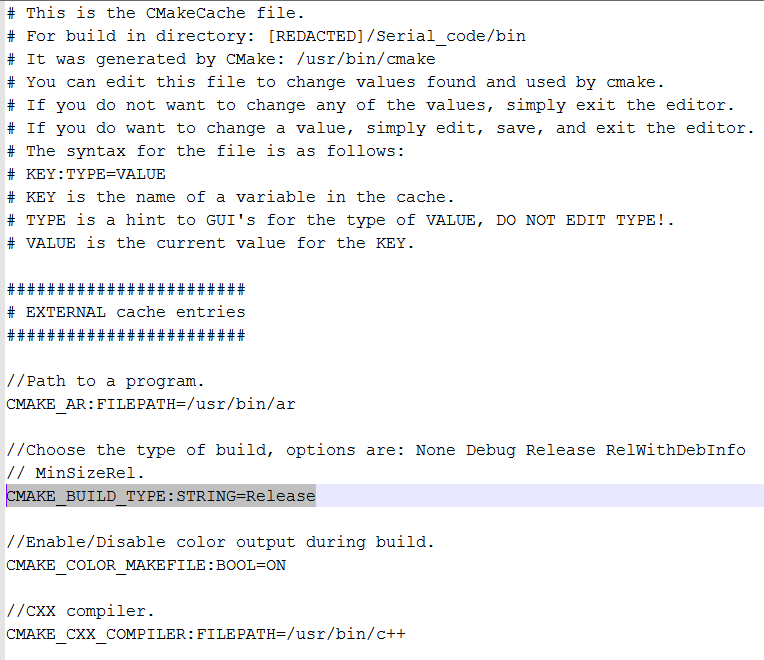
cd bin

This cmake script will create a directory named “bin”. Enter this directory:

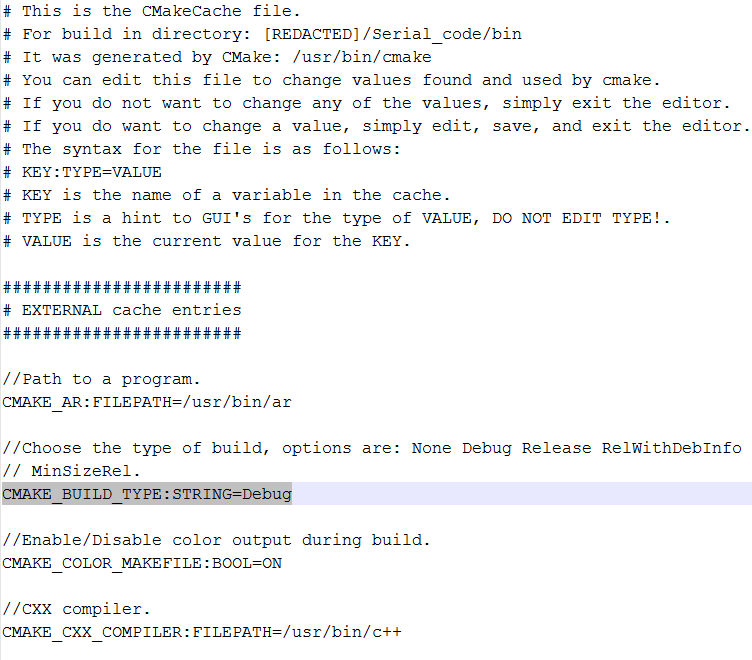
and type:

make

Four executables – GOMC\_Serial\_GEMC (Gibbs ensemble), GOMC\_Serial\_NVT (NVT ensemble), GOMC\_Serial\_NPT (isobaric-isothermal ensemble), and GOMC\_Serial\_GCMC (Grand canonical ensemble) – will be produced. By default, the distribution compiles in release mode. To compile in debug mode (if you’re using the code as a developer), open the file “CMakeCache.txt” while still in the “bin” folder. This file contains information used by cmake to build the executables. To compile in debug mode, change the value after “CMAKE\_BUILD\_TYPE:STRING=” from “Release” to “Debug”.



To



And retype the command:

make

The output executables should now be compiled with debugger symbols.

You can also swap the compiler by modifying the “CMAKE\_CXX\_COMPILER” variable. For more information, refer to the Cmake documentation.

Running GOMC in parallel using Openmp:

To run the parallel version of CPU code, it needs to be compiled with openmp library. Open the file “CMakeCache.txt”, while still in the “bin” folder, and change the value after “CMAKE\_CXX\_FLAGS\_RELEASE:STRING=” from “-O3 -DNDEBUG” to “-O3 -qopenmp -DNDEBUG”.

And retype the command:

make

**Input File Formats**

In order to run simulation in GOMC, the following files need to be provided:

* GOMC executable
* Input file “NAME.conf” (proprietary control file)
* PDB file or files
* PSF file or files
* Parameter file

**PDB**

The PDB file stores coordinates for the simulation. The file format is widely adopted.

* Protein Databank (PDB) Files (plural: PDB files)
* Open format, well-documented
* Fixed-width format (hence white space is significant)
* Up to 13.5m page views a month; up to 55.8m FTP requests per month
* Used by NAMD, GROMACS, CHARMM, ACEMD, Amber

An overview of the PDB standard can be found here:

<http://www.wwpdb.org/docs.html>

The advantage of PDB files is their ubiquity and thorough documentation. Disadvantages include limited fixed point floating precision for coordinates, unused space, and proprietary implementations creating inconsistencies.

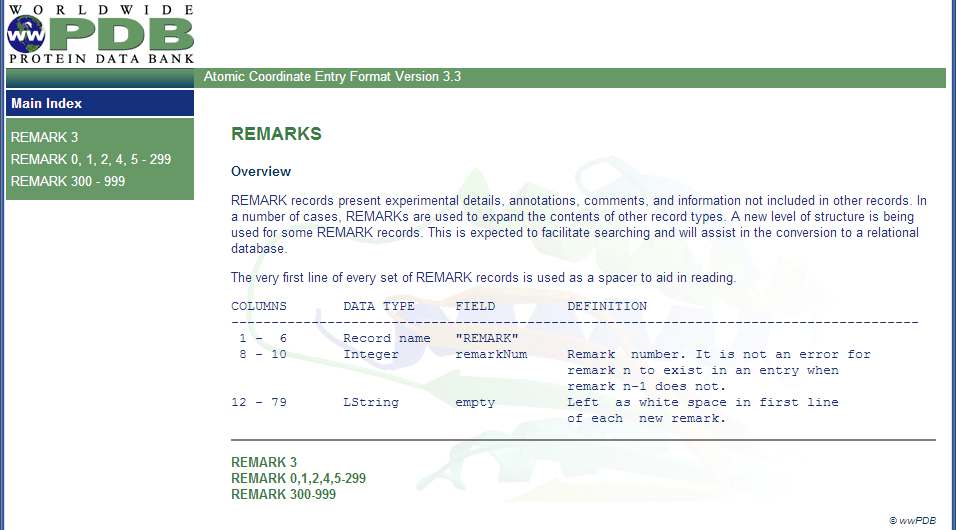
One PDB file is required per box. For NVT ensemble simulations, one file is expected; for Gibbs and grand canonical ensemble, two files are required.

GOMC recognizes the following keywords in PDB files:

* **REMARK**
* **CRYST1**
* **ATOM**
* **END**

Currently, **REMARK** is ignored. Formerly, it was used to store proprietary information in frames (e.g. step number). Packmol typically leaves the following remark

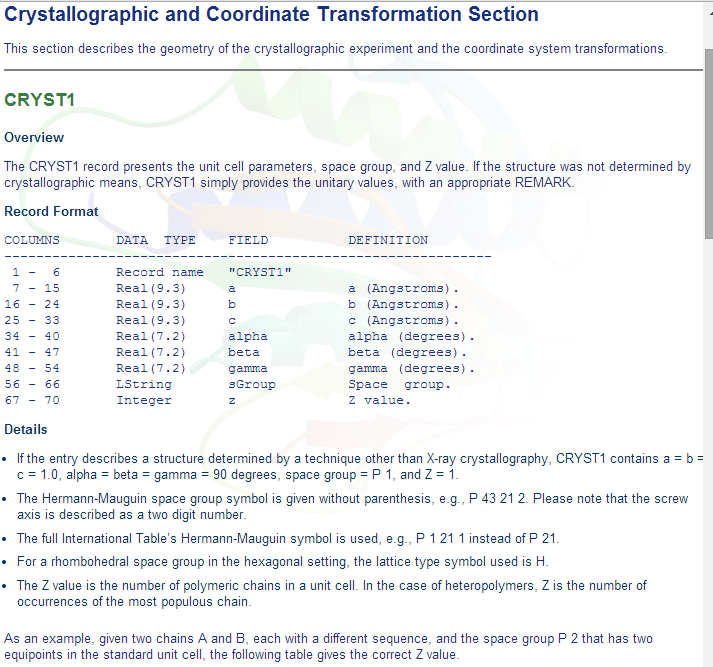
REMARK original generated coordinate pdb file

at the top of the file. Note that this is another example of an inconsistency with the spec. As of the PDB v3.30 specification the REMARK entry contains an identifying integer, which is supposed to occupy lines 8-10.   
  
  
  
A file generated by Packmol has “ori” in this position. Hence you may see future codes that are incompatible with this legacy kind of remarks.

Note also that the spaces 7 and 11 are not reserved; hence, they may be used in proprietary specifications.

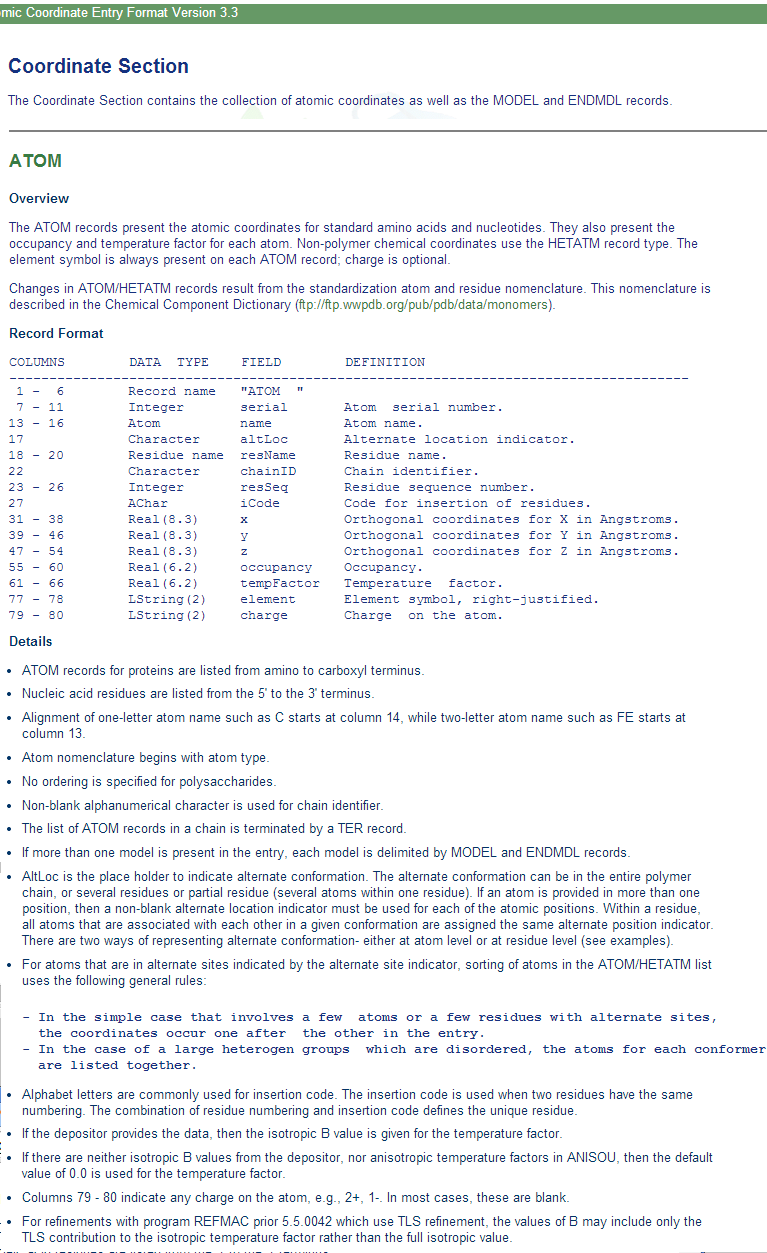
**CRYST1** can be used to store the cell dimensions, which can also be put as a tag in the proprietary control file.

<http://www.wwpdb.org/documentation/format33/sect8.html#CRYST1>

  
  
Here is an example:

REMARK 1\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_3\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_5\_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_7\_\_\_\_\_\_\_\_\_8  
REMARK 8901234567890123456789012345678901234567890123456789012345678901234567890  
CRYST1 71.490 71.490 71.490 90.00 90.00 90.00 P 1 1  
ATOM 1 C1 ISB A 1 24.378 36.667 45.645 0.00 0.00 ISB C  
  
This file indicates a box size of 71.49 Angstroms per side. The ’90.00’ entries, indicate that the cell is cubic.   
  
*NOTE:*  
Only cubic and orthogonal cells are supported in this code.

The main entry in the PDB file are **ATOM** entries. The keyword “ATOM” is always followed by two spaces. An entry has a number of fields.



The key parameters are the coordinates x, y, and z. The precision is limited to eight whole decimal digits and three fractional decimal digits.

Other important entries are the residue name, atom name, and chain ID. Numbering is important primarily because it represents an inconvenience in packing/loading large systems. Revisiting the previous example,

REMARK 1\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_3\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_5\_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_7\_\_\_\_\_\_\_\_\_8  
REMARK 8901234567890123456789012345678901234567890123456789012345678901234567890  
CRYST1 71.490 71.490 71.490 90.00 90.00 90.00 P 1 1  
ATOM 1 C1 ISB A 1 24.378 36.667 45.645 0.00 0.00 ISB C

the atom name is “C1” and residue name is “ISB”. The PSF file (next section) contains a lookup table of atoms. These contain the atom name from the PDB and the name of the atom kind in the parameter file it corresponds to. As multiple different atom names will all correspond to the same parameter, these can be viewed “atom aliases” of sorts. The chain letter (in this case ‘A’) is sometimes used when packing a number of PDBs into a single PDB file.

*A few important Notes/Warnings on Undocumented PDB Format Conventions:*

* While it is explicitly stated in some other sections of the PDB file, the general convention observed by most codes is to right align when padding with white space.
* Some codes (including PSFGen/VMD) use the 21st unused character to add a fourth letter to the residue (molecule name). This extension is currently supported, but is unofficial and, hence, may change in the future.
* VMD requires a constant number of ATOMs in a multi-frame PDB (multiple records terminated by “END” in a single file). To compensate for this, all atoms from all boxes in the system are written to the output PDBs of this code.
* For atoms not currently in a box, the coordinates are set to <0.00, 0.00, 0.00>
* The occupancy is commonly just set to “1.00” and is left unused by many codes. We recycle this legacy parameter by using it to denote, in our output PDBs, the box a particle is in (box 0 🡪 occupancy=0.00 ; box 1 🡪 occupancy=1.00)
* As the x, y, and z coordinates are fixed point with only three digits of precision, the energy values you get when restarting may be mildly different, particularly for bonded interactions due to roundoff in the coordinates. This will eventually be remedied by the implementation of a full-precision trajectory (e.g. DCD) file.
* The “ISB” entry in columns 73-75 is not an official part of the PDB standard. This is a proprietary entry called “Segname”, which has been embraced by NAMD and some other codes.

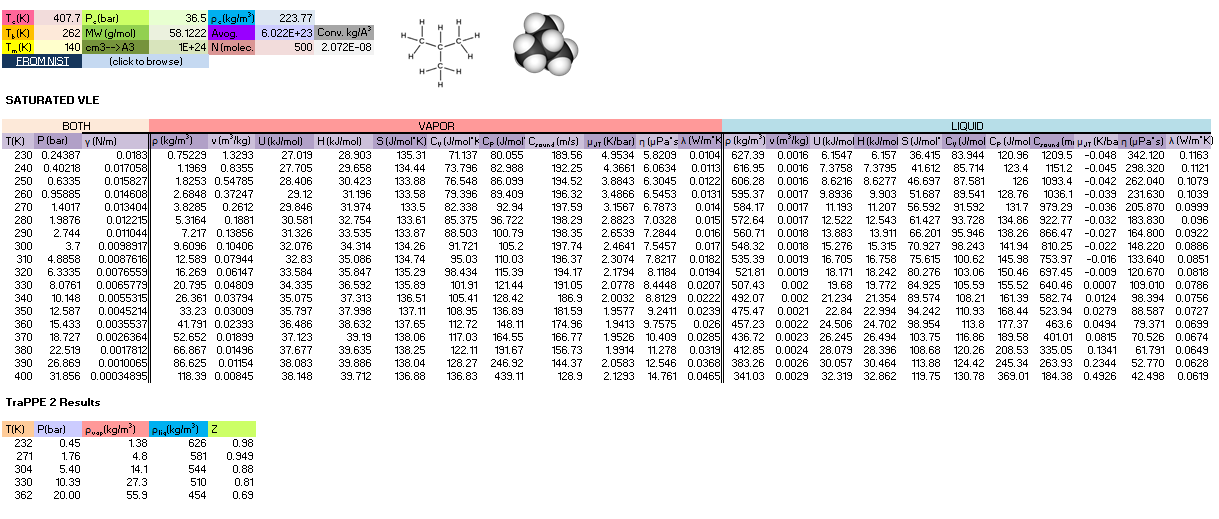
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A frame in the PDB file is terminated with the keyword **END**.

With that overview of the format in mind, the following steps describe how a PDB file is typically built.  
  
1. A single molecule PDB is obtained. In this example, the QM software package Gaussian was used to draw the molecule, which was then edited by hand to adhere to the PDB spec properly. The end result is a PDB for a single molecule:  
  
REMARK 1 File created by GaussView 5.0.8  
ATOM 1 C1 ISB 1 0.911 -0.313 0.000 C  
ATOM 2 C2 ISB 1 1.424 -1.765 0.000 C  
ATOM 3 C3 ISB 1 -0.629 -0.313 0.000 C  
ATOM 4 C4 ISB 1 1.424 0.413 -1.257 C  
END

2. Next, packings are calculated to place the simulation in a region of vapor-liquid coexistence. There are a couple of ways to do this in Gibbs ensemble:

* Pack both boxes to a single “middle” density, which is an average of the liquid and vapor densities.
* Same as 1, but add a modest amount to axis of one box (e.g. 10-30 A). This technique can be handy in the constant pressure Gibbs ensemble.
* Pack one box to the predicted liquid density and the other to the vapor density.



A good reference for getting the information needed to estimate packing is the NIST Web Book database of pure compounds:

<http://webbook.nist.gov/chemistry/>

3. After packing is determined, a basic pack can be performed with a Packmol script. Here is one example:

tolerance 3.0  
filetype pdb  
output STEP2\_ISB\_packed\_BOX\_0.pdb  
  
structure isobutane.pdb  
number 1000  
inside box 0.1 0.1 0.1 70.20 70.20 70.20  
end structure

Packmol scripts are typically saved with the extension \*.inp, so this might be named “pack\_isobutane.inp”.  
  
To run the script, we type the following line into the terminal:

./packmol < pack\_isobutane.inp

**PSF File**

The PSF file stores the topology, mass, charges, and atom identities of molecules in the system.

* Protein Structure File (PSF)
* Space-separated file
* Used by NAMD, CHARMM, X-PLOR

The PSF file is not as robustly documented as the PDB format, but a basic description of it can be found here:  
  
<http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node24.html>

The PSF file is generally composed of a series of sections. A line with a numeric value is typically at the top of each section. This value lists the number of entries in that section (lines can contain multiple entries; a dihedral, for example has two quadruplet entries of atom indices per line). Note that outside the remarks and atom section, this number is typically smaller than the number of lines by a factor of 2 to 4.

PSF files always start with the string “PSF” on their first line.

GOMC reuses PSF reading code from NAMD, hence it should have much of the same flexibility and limitations. By section, the segments of a PSF file are:

* TITLE: remarks on the file
* BONDS: the bonds (if applicable) in molecules
* ANGLE: the bonds (if applicable) in molecules
* DIHEDRAL: the bonds (if applicable) in molecules
* IMPROPER: the bonds (if applicable) in molecules
* (other sections such as cross terms)

The code currently skips the title section and reads the bonds, angles, dihedrals and impropers.

*A few important Notes/Warnings:*

* The PSF file format is a highly redundant file format. It repeats identical topology of thousands of molecules of a common kind in some cases. GOMC follows the same approach as NAMD, allowing this excess information externally and compiling it in the code.
* Other sections (e.g. cross terms) contain unsupported or legacy parameters and are ignored.
* Following the restrictions of VMD, the order of the PSF atoms must match the order in the PDB file.
* Improper entries are read and stored, but are not currently used. Support will eventually be added for this.

The PSF file is typically generated using PSFGen. It is convenient to make a script, such as the example below, to do this:  
  
psfgen << ENDMOL  
topology ./Top\_Branched\_Alkanes.inp  
segment ISB {  
 pdb ./STEP2\_ISB\_packed\_BOX\_0.pdb  
 first none  
 last none  
}

coordpdb ./STEP2\_ISB\_packed\_BOX\_0.pdb ISB

writepsf ./STEP3\_START\_ISB\_sys\_BOX\_0.psf  
writepdb ./STEP3\_START\_ISB\_sys\_BOX\_0.pdb

Typically, one script is run per box to generate a finalized PDB/PSF for that box. The script requires one additional file, the NAMD-style topology file. While GOMC does not directly read or interact with this file, it’s typically used to generate the PSF and, hence, is considered one of the integral file types. It will be briefly discussed in the following section.

Here’s a peek at how the generated PSF file looks for a packed isobutane system (abridged):

PSF

3 !NTITLE

REMARKS original generated structure x-plor psf file

REMARKS topology ./Top\_Branched\_Alkanes.inp

REMARKS segment ISB { first NONE; last NONE; auto angles dihedrals }

4000 !NATOM

1 ISB 1 ISB C1 CH1 0.000000 13.0190 0

2 ISB 1 ISB C2 CH3 0.000000 15.0350 0

3 ISB 1 ISB C3 CH3 0.000000 15.0350 0

4 ISB 1 ISB C4 CH3 0.000000 15.0350 0

5 ISB 2 ISB C1 CH1 0.000000 13.0190 0

6 ISB 2 ISB C2 CH3 0.000000 15.0350 0

7 ISB 2 ISB C3 CH3 0.000000 15.0350 0

8 ISB 2 ISB C4 CH3 0.000000 15.0350 0  
 .

.

.  
 3997 ISB 1000 ISB C1 CH1 0.000000 13.0190 0

3998 ISB 1000 ISB C2 CH3 0.000000 15.0350 0

3999 ISB 1000 ISB C3 CH3 0.000000 15.0350 0

4000 ISB 1000 ISB C4 CH3 0.000000 15.0350 0

3000 !NBOND: bonds

1 2 1 3 1 4 5 6

5 7 5 8  
 .

.

.

3997 3998 3997 3999 3997 4000

3000 !NTHETA: angles

2 1 4 2 1 3 3 1 4

6 5 8 6 5 7 7 5 8

.

.

.

3998 3997 4000 3998 3997 3999 3999 3997 4000

0 !NPHI: dihedrals

0 !NIMPHI: impropers

0 !NDON: donors

0 !NACC: acceptors

0 !NNB

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

.

.

.

**Topology File**

The topology is a whitespace separated file format, which contains a list of atoms and their corresponding masses, and a list of residue information (charges, composition, and topology). Essentially, it is a non-redundant lookup table equivalent to the PSF file.

This is followed by a series of residues, which tell PSFGen what atoms are bonded to a given atom. Each residue is comprised of four key elements:

* A header beginning with the keyword **RESI** with the residue name and net charge
* A body with multiple **ATOM** entries (not to be confused with the PDB-style entries of the same name), which list the partial charge on the particle and what kind of atom each named atom in a specific molecule/residue is.
* A section of lines starting with the word **BOND** contains pairs of bonded atoms (typically 3 per line)
* A closing section with instructions for PSFGen.

Here’s an example of a residue definition for isobutane:  
  
RESI ISB 0.00 ! isobutane – TraPPE  
GROUP  
ATOM C1 CH1 0.00 ! C3  
ATOM C2 CH3 0.00 ! C2-C1  
ATOM C3 CH3 0.00 ! C4  
ATOM C4 CH3 0.00 !  
BOND C1 C2 C1 C3 C1 C4  
PATCHING FIRS NONE LAST NONE

Here’s a full parameter file prepared to pack a system of isobutane:

\*  
\* Custom top file -- branched alkanes  
\*  
1 1  
!

MASS 1 CH3 15.035 C !

MASS 2 CH1 13.019 C !

RESI ISB 0.00 ! isobutane – TraPPE  
GROUP  
ATOM C1 CH1 0.00 ! C3  
ATOM C2 CH3 0.00 ! C2-C1  
ATOM C3 CH3 0.00 ! C4  
ATOM C4 CH3 0.00 !  
BOND C1 C2 C1 C3 C1 C4  
PATCHING FIRS NONE LAST NONE  
  
END

Note that the keyword **END** must be used to terminate this file and keywords related to the auto-generation process must be placed near the top of the file, after the **MASS** definitions.

More in-depth information can be found in the following links:

“Topology Tutorial” (PDF, in-depth)  
<http://www.ks.uiuc.edu/Training/Tutorials/science/topology/topology-tutorial.pdf>  
“NAMD Tutorial: 4. Examining the Topology File”  
<http://www.ks.uiuc.edu/Training/Tutorials/science/topology/topology-html/node4.html>  
“Developing Topology and Parameter Files”  
[http://www.ks.uiuc.edu/Training/Tutorials/science/force field-tutorial/force field-html/node6.html](http://www.ks.uiuc.edu/Training/Tutorials/science/forcefield-tutorial/forcefield-html/node6.html)  
“NAMD Tutorial: 25. Topology Files”  
<http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node25.html>

courtesy of UIUC.

**Parameter File(s):**

Currently, GOMC uses a single parameter file and the user has the two kinds of parameter file choices:

* CHARMM (Chemistry at Harvard Molecular Mechanics) compatible parameter file
* “EXOTIC” parameter file

If the parameter file type is not specified or if the chosen file is missing, an error will result.

Both force field file options are whitespace separated files with sections preceded by a tag. When a known tag (representing a molecular interaction in the model) is encountered, reading of that section of the force field begins. Comments (anything after a \* or !) and whitespace are ignored. Reading concludes when the end of the file is reached or another section tag is encountered.

*CHARMM format parameter file*

CHARMM contains a widely used model for describing energies in Monte Carlo and molecular dynamics simulations. It is intended to be compatible with other codes that use such a format, such as NAMD.

For a general overview of the CHARMM force field, see:

<http://www.charmmtutorial.org/index.php/The_Energy_Function>

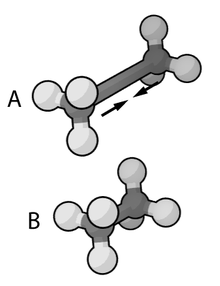
Here’s the basic CHARMM contributions that are supported in GOMC:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

As seen above, the following are recognized, read and used:

* **BONDS**
  + Quadratic expression describing bond stretching based on bond length (*b*) in Angstrom
  + Typically, it is ignored as bonds are rigid for Monte Carlo simulations. To specify that it is to be ignored, put a very large value i.e. “999999999999” for .

*Note:*

GOMC does not sample bond stretch.  
  
  
  
[Image Courtesy of Wikimedia Commons]

* **ANGLES** 
  + Describes the conformational behavior of an angle (*θ*) between three atoms, one of which is shared branch point to the other two. To fix any angle and ignore the related angle energy, put a very large value i.e. “999999999999” for .

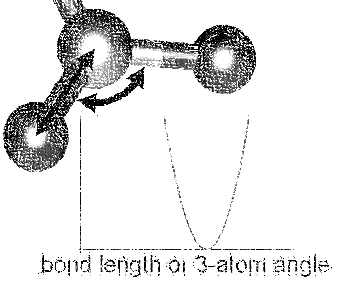
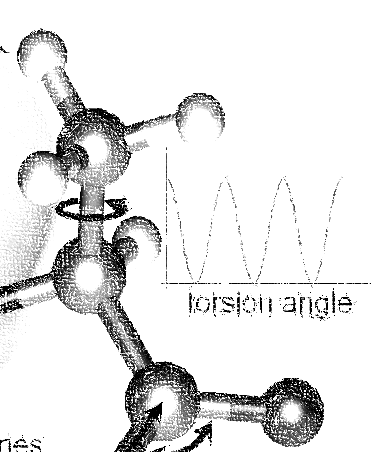
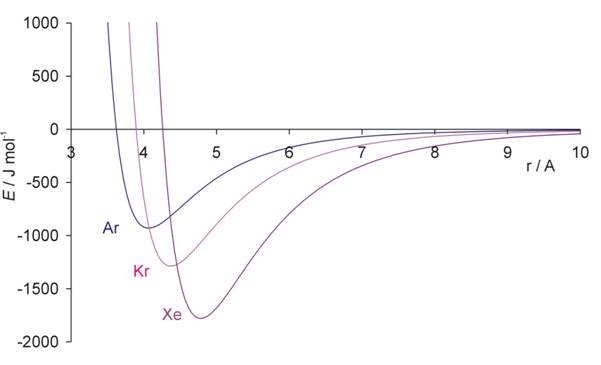


Image Courtesy of Wikimedia Commons

* **DIHEDRALS** Describes crankshaft-like rotation behavior about a central bond in a series of three consecutive bonds (rotation is given as ϕ)

  
 Image Courtesy of Wikimedia Commons

* **NONBONDED** This tag name only should be used if CHARMM force files are being used. This section describes 12-6 (Lennard-Jones) non-bonded interactions. Non-bonded parameters are assigned by specifying atom type name followed by polarizabilities (which will be ignored), minimum energy, and (minimum radius)/2. In order to modify 1-4 interaction, a second polarizability (again, will be ignored), minimum energy, and (minimum radius)/2 need to be defined; otherwise, the same parameter will be considered for 1-4 interaction.

  
Image Courtesy of the Univ. of Bristol

* **NBFIX** This tag name only should be used if CHARMM force field is being used. This section allows interaction between two pairs of atoms to be modified, done by specifying two atom type names followed by minimum energy and minimum radius. In order to modify 1-4 interaction, a second minimum energy and minimum radius need to be defined; otherwise, the same parameter will be considered for 1-4 interaction.

*NOTE*:

Please pay attention that in this section we define minimum radius, not (minimum radius)/2 as it is defined in the NONBONDED section.

Currently, supported sections of the CHARMM compliant file include **BONDS**, **ANGLES**, **DIHEDRALS**, **NONBONDED**, **NBFIX**. Other sections such as **CMAP** are not currently read or supported.

**BONDS** (“bond stretching”) is one key section of the CHARMM-compliant file. Units for the Kb variable in this section are in kcal/mol; the b0 section (which represents the equilibrium bond length for that kind of pair) is measured in Angstroms.

BONDS

!V(bond) = Kb(b - b0)\*\*2

!

!Kb: kcal/mole/A\*\*2

!b0: A

!

! Kb (kcal/mol) = Kb (K) \* Boltz. const.;

!

!atom type Kb b0 description

CH3 CH1 9999999999 1.540 ! TraPPE 2

*NOTE:*The Kb value may appear odd, but this is because a larger value corresponds to a more rigid bond. As Monte Carlo force fields (e.g. TraPPE) typically treat molecules as rigid constructs, Kb is set to a large value – 9999999999. Sampling bond stretch is not supported in GOMC.

**ANGLES** (bond bending), where θ and θ0 are commonly measured in degrees and Kθ is measured in kcal/mol/K. These values, in literature, are often expressed in Kelvin (K). To convert Kelvin to kcal/mol/K, multiply by the Boltzmann constant -- kb, 0.0019872041 kcal/mol. In order to fix the angle, it requires to set a large value for Kθ. By assigning a large value like 9999999999, specified angle will be fixed and energy of that angle will considered to be zero.

Here is an example of what is necessary for isobutane:

ANGLES

!

!V(angle) = Ktheta(Theta - Theta0)\*\*2

!

!V(Urey-Bradley) = Kub(S - S0)\*\*2

!

!Ktheta: kcal/mole/rad\*\*2

!Theta0: degrees

!S0: A

!

! Ktheta (kcal/mol) = Ktheta (K) \* Boltz. const.

!

!atom types Ktheta Theta0 Kub(?) S0(?)

CH3 CH1 CH3 62.100125 112.00 ! TraPPE 2

Some CHARMM **ANGLES** section entries include Urey-Bradley potentials (Kub, bub), in addition to the standard quadratic angle potential. The constants related to this potential function are currently read, but the logic has not been added to calculate this potential function. Support for this potential function will be added in later versions of the code.

The final major bonded interactions section of the CHARMM compliant parameter file are the **DIHEDRALS**. Each dihedral is composed of a dihedral series of 1 or more terms. Often, there are 4 to 6 terms in a dihedral. Angles for the dihedrals’ deltas are given in degrees.

Since isobutane has no dihedral, here are the parameters pertaining to 2,3-dimethylbutane:

DIHEDRALS

!

!V(dihedral) = Kchi(1 + cos(n(chi) - delta))

!

!Kchi: kcal/mole

!n: multiplicity

!delta: degrees

!

! Kchi (kcal/mol) = Kchi (K) \* Boltz. const.

!

!atom types Kchi n delta description

X CH1 CH1 X -0.498907 0 0.0 ! TraPPE 2

X CH1 CH1 X 0.851974 1 0.0 ! TraPPE 2

X CH1 CH1 X -0.222269 2 180.0 ! TraPPE 2

X CH1 CH1 X 0.876894 3 0.0 ! TraPPE 2

*NOTE*:   
The code allows the use of ‘X’ to indicate ambiguous positions on the ends. This is useful because this kind is often determined solely by the two middle atoms in the middle of the dihedral, according to literature.

**IMPROPERS** – energy parameters used to describe out-of-plane rocking are currently read, but unused. The section is often blank. If it becomes necessary, algorithms to calculate the improper energy will need to be added.

The next section of the CHARMM style parameter file is the **NONBONDED**. In order to use TraPPE this section of the CHARMM compliant file is critical. Here’s an example with our isobutane potential model:

NONBONDED

!

!V(Lennard-Jones) = Eps,i,j[(Rmin,i,j/ri,j)\*\*12 - 2(Rmin,i,j/ri,j)\*\*6]

!

!atom ignored epsilon Rmin/2 ignored eps,1-4 Rmin/2,1-4

!

CH3 0.0 -0.194745992 2.10461634058 0.0 0.0 0.0 ! TraPPE 1

CH1 0.0 -0.019872040 2.62656119304 0.0 0.0 0.0! TraPPE 2  
End

*NOTE*:  
The Rmin is different from σ. σ is the distance to the x-intercept (where interaction energy goes from being repulsive to positive). Rmin is the potential well-depth, where the attraction is maximum. To convert σ to Rmin, simply multiply σ by 0.56123102415, and flag it with a negative sign.

The last section of the CHARMM style parameter file is the **NBFIX**. In this section, individual pair interaction will be modified. First, pseudo non-bonded parameters have to be defined in **NONBONDED** and modified in **NBFIX**. Here’s an example if it is required to modify interaction between CH3 and CH1 atoms:

NBFIX

!V(Lennard-Jones) = Eps,i,j[(Rmin,i,j/ri,j)\*\*12 - 2(Rmin,i,j/ri,j)\*\*6]

!

!atom atom epsilon Rmin eps,1-4 Rmin,1-4

CH3 CH1 -0.294745992 1.10461634058 !   
End

*“Exotic” parameter file*

The exotic file is intended for use with nonstandard/specialty models of molecular interaction, which are not included in CHARMM standard. Currently, two custom interaction are included:

* **NONBONDED\_MIE** This section describes n-6 (Lennard-Jones) non-bonded interactions. The Lennard-Jones potential (12-6) is a subset of this potential. Non-bonded parameters are assigned by specifying atom type name followed by minimum energy, atom diameter, and repulsion exponent. In order to modify 1-4 interaction, a second minimum energy, atom diameter, and repulsion exponent need to be defined; otherwise, the same parameters would be considered for 1-4 interaction.
* **NBFIX\_MIE** This section allows n-6 (Lennard-Jones) interaction between two pairs of atoms to be modified. This is done by specifying two atoms type names followed by minimum energy, atom diameter, and repulsion exponent. In order to modify 1-4 interaction, a second minimum energy, atom diameter, and repulsion exponent need to be defined; otherwise, the same parameter will be considered for 1-4 interaction.

*NOTE:*

In EXOTIC force field, the definition of atom diameter(σ) is same for both **NONBONDED\_MIE** and **NBFIX\_MIE**.

Otherwise, the exotic file reuses the same geometry section headings -- **BONDS** / **ANGLES** / **DIHEDRALS** / etc. The only difference in these sections versus in the CHARMM format force field file is that the energies are in Kelvin (‘K’), the unit most commonly found for parameters in Monte Carlo chemical simulation literature. This precludes the need to convert to kcal/mol, the energy unit used in CHARMM.

The most frequently used section of the exotic files in the Mie potential section is **NONBONDED\_MIE**.   
  
Here are the parameters that are used to simulate alkanes:

NONBONDED\_MIE

!

!V(mie) = 4\*eps\*((sig\_ij/r\_ij)^n-(sig\_ij/r\_ij)^6)

!

!atom eps sig n eps,1-4 sig,1-4 n,1-4

CH4 161.00 3.740 14 0.0 0.0 0.0 ! Potoff, et al. '09

CH3 121.25 3.783 16 0.0 0.0 0.0 ! Potoff, et al. '09

CH2 61.00 3.990 16 0.0 0.0 0.0 ! Potoff, et al. '09

Here is an example if there is need to modify CH3 CH2 interaction:

NBFIX\_MIE

!

!V(mie) = 4\*eps\*((sig\_ij/r\_ij)^n-(sig\_ij/r\_ij)^6)

!

!atom atom eps sig n eps,1-4 sig,1-4 n,1-4

CH2 CH3 81.10 2.540 16 31.10 1.540 16!

*NOTE:*

Although the units (Angstroms) are the same, the exotic file uses σ, *not* the Rmin used by CHARMM. The energy in the exotic file are expressed in Kelvin (K), as this is the standard convention in the literature.

**Control File (\*.conf):**

The control file is GOMC’s proprietary input file. It contains key settings. The settings generally fall under three categories:

* Input/Simulation Setup
* System Settings for During Run
* Output Settings

*NOTE:*

The control file is designed to recognize logic values, such as yes/true/on or no/false/off.

*Input/Simulation Setup*

In this section, input file names are listed. In addition, if you want to restart your simulation or use integer seed for running your simulation, you need to modify this section according to your purpose.

* **Restart**: Determines whether to restart and, if so, what step to restart from.  
    
  (WARNING: Restarts are not currently supported)  
  + Value 1: <BOOLEAN> – true if restart, false otherwise
* **FirstStep**: Determines what step to restart from. If **Restart** was set to true, step number needs to be specified; otherwise, the program will be terminated.
  + Value 1: <ULONG> – step to restart from   
      
    Example:  
    #################################  
    # enable, step  
    #################################  
    Restart true

FirstStep 1000000

* **PRNG:** Dictates how to start the pseudo-random number generator (PRNG).
  + Value 1: <STRING >
    - RANDOM: Randomizes Mersenne Twister PRNG with random bits based on the system time.

Example:  
####################################  
# kind {RESTART, RANDOM, INTSEED}  
####################################  
PRNG RANDOM

* + - INTSEED: This option “seeds” the Mersenne Twister PRNG with a standard integer. When the same integer is used, the generated PRNG stream should be the same every time, which is helpful in tracking down bugs.
    - RESTART: Used for restarting a previous simulation. This option loads in the Mersenne Twister’s state from a saved file.

(WARNING: Restarts are not currently supported)

Example:   
####################################  
# kind {RESTART, RANDOM, INTSEED}  
####################################  
PRNG RESTART

* **Random\_Seed:** Defines seed number. If “INTSEED” is chosen, seed number needs to be specified; otherwise, the program will be terminated.
  + Value 1: ULONG or UINT: If “INTSEED” command is used (see above example).  
      
    Example:  
    ####################################

# kind {RESTART, RANDOM, INTSEED}

####################################

PRNG INTSEED

Random\_Seed 50

* **ParaTypeCHARMM:** Sets force field type to CHARMM style.
  + Value 1: <BOOLEAN> – true if it is CHARMM force field, false if it is not.

Example:  
####################################  
# FORCE FIELD TYPE  
####################################  
ParaTypeCHARMM true

* **ParaTypeEXOTIC:** Sets force field type to EXOTIC style.
  + Value 1: <BOOLEAN> – true if it is EXOTIC force field, false if it is not.

Example:  
####################################  
# FORCE FIELD TYPE  
####################################  
ParaTypeEXOTIC true

* **ParaTypeMARTINI:** Sets force field to MARTINI.
  + Value 1: <BOOLEAN> – true if it is MARTINI force field, false if it is not.

Example:  
####################################  
# FORCE FIELD TYPE  
####################################  
ParaTypeMARTINI true

* **Parameters:** Provides the name and location of the parameter file to use for the simulation.
  + Value 1: <STRING> – Sets the name of parameter file

Example:  
####################################  
# FORCE FIELD TYPE  
####################################  
ParaTypeCHARMM yes

Parameters ./../common/Par\_TraPPE\_Alkanes.inp

* **Coordinates:** Defines the PDB filenames (coordinates) for each box in the system.
  + Value 1: <INTEGER> – Sets box number (first box is box ‘0’)
  + Value 2: <STRING> – Sets pdb file name

*Note:*

NVT and NPT ensembles requires only one PDB file and GEMC/GCMC requires two PDB files. If the number of PDB files is not compatible to with the simulation type, the program will be terminated.

Example (NVT ensemble or NPT ensemble):  
####################################  
# INPUT PDB FILES  
####################################  
Coordinates 0 STEP3\_START\_ISB\_sys.pdb  
  
  
Example (Gibbs ensemble or GC ensemble):   
####################################  
# INPUT PDB FILES  
####################################  
Coordinates 0 STEP3\_START\_ISB\_sys\_BOX\_0.pdb

Coordinates 1 STEP3\_START\_ISB\_sys\_BOX\_1.pdb

* **Structures:** Defines the PSF filenames (structures) for each box in the system
  + Value 1: <INTEGER> – Sets box number (first box is box ‘0’)
  + Value 2: <STRING> – Sets PSF file name

*Note:*

NVT and NPT ensembles requires only one PSF file and GEMC/GCMC requires two PSF files. If the number of PSF files is not compatible with the simulation type, the program will be terminated.

Example (NVT ensemble or NPT ensemble):  
####################################  
# INPUT PSF FILES  
####################################  
Structure 0 STEP3\_START\_ISB\_sys.psf  
  
Example (Gibbs ensemble or GC ensemble):  
####################################  
# INPUT PSF FILES  
####################################  
Structure 0 STEP3\_START\_ISB\_sys\_BOX\_0.psf

Structure 1 STEP3\_START\_ISB\_sys\_BOX\_1.psf

*System Settings for During Run Setup*

This section contains all the variables not involved in the output of data during the simulation, or in the reading of input files at the start of the simulation. In other words, it contains settings related to the moves, the thermodynamic constants (based on choice of ensemble), and the length of the simulation.

Note that some tags, or entries for tags, are only used in certain ensembles (e.g. Gibbs ensemble). These cases are denoted with colored text.

* **GEMC**: (For Gibbs Ensemble runs only) Defines what type of Gibbs Ensemble simulation you want to run.  
    
  If neglected in Gibbs Ensemble, it simply defaults to constant volume (NVT) Gibbs Ensemble.  
  + Value 1: <STRING> – allows you to pick between isovolumetric (“NVT”) and isobaric (“NPT”) Gibbs ensemble simulations
    - NVT: Run simulation with constant mole number, volume and temperature.

Example:  
##################################  
# GEMC TYPE (DEFULT IS NVT\_GEMC)  
##################################  
GEMC NVT

* + - NPT: Run simulation with constant mole number, pressure, and temperature.
* **Pressure**: If “NPT” simulation is chosen, imposed pressure (in bar) needs to be specified; otherwise, the program will be terminated.
  + Value 1: <DOUBLE> – Constant pressure in bars

Example:  
##################################  
# GEMC TYPE (DEFULT IS NVT\_GEMC)  
##################################  
GEMC NPT

Pressure 5.76

* **Temperature**: Sets the temperature at which the system will run.
  + Value 1: <DOUBLE> – Constant temperature of simulation in degrees Kelvin
* **Rcut**: Sets a specific radius in angstrom that non-bonded interaction energy and force will be considered and calculated using defined potential function.
  + Value 1: <DOUBLE> – The distance to truncate the Lennard-Jones potential at.
* **RcutLow**: Sets a specific minimum distance in angstrom that reject any move that place any atom closer than specified distance.
* Value 1: <DOUBLE> – The minimum possible distance between any atoms.
* **LRC**: Defines whether or not long range corrections are used.
  + Value 1: <BOOLEAN> –True to consider long range correction. In case of using “SHIFT” or “SWITCH” potential functions, LRC will be ignored.
* **Exclude**: Defines which pairs of bonded atoms should be excluded from non-bonded interactions.
  + Value 1: <STRING> – Allows you to choose between “1-2”, “1-3” and “1-4”.
    - 1-2: All interaction pairs of bonded atoms, except the ones that separated with one bond, will be considered and modified using 1-4 parameters defined in parameter file.
    - 1-3: All interaction pairs of bonded atoms, except the ones that separated with one or two bonds, will be considered and modified using 1-4 parameters defined in parameter file.
    - 1-4: All interaction pairs of bonded atoms, except the ones that separated with one, two or three bonds, will be considered using non-bonded parameters defined in parameter file.

*Note:*

If no value was detected, by default “1-3” will be considered.

* **Potential**: Defines the potential function type to calculate non-bonded interaction energy and force between atoms.
  + Value 1: <STRING> – Allows you to pick between “VDW”, “SHIFT” and “SWITCH”
    - VDW: Nonbonded interaction energy and force calculated based on n-6 (Lennard-Johns) equation. This function will be discussed further in the Intermolecular energy and Virial calculation section.

Example:

#############################

# SIMULATION CONDITION

#############################

Temperature 200.00

Potential VDW

LRC true

Rcut 10

Exclude 1-4

* + - SHIFT: This option forces the potential energy to be zero at Rcut distance. This function will be discussed further in the Intermolecular energy and Virial calculation section.

Example:

#############################

# SIMULATION CONDITION

#############################

Temperature 200.00

Potential SHIFT

LRC false

Rcut 10

Exclude 1-4

* + - SWITCH: This option smoothly forces the potential energy to be zero at Rcut distance and starts modifying the potential at **Rswitch** distance. Depending on force field type, specific potential function will be applied. These functions will be discussed further in the Intermolecular energy and Virial calculation section.
* **Rswitch**: In the case of choosing “SWITCH” as potential function, a distance is set in which non-bonded interaction energy is truncated smoothly from to cutoff distance.
  + Value 1: <DOUBLE> – Define switch distance in angstrom. If the “SWITCH” function is chosen, Rswitch needs to be defined; otherwise, the program will be terminated.

Example:

#############################

# SIMULATION CONDITION

#############################

Temperature 200.00

Potential SWITCH

Rswitch 7

Rcut 10

RcutLow 1.0

LRC false

Exclude 1-3

*Note/Warning:*In CHARMM force field, the 1-4 interaction needs to be considered. Choosing “Exclude 1-3” will modify 1-4 interaction based on 1-4 parameter in parameter file. If a kind force field is used, where 1-4 interaction needs to be ignored, such as TraPPE, either “exclude 1-4” needs to be chosen or 1-4 parameter needs to be assigned a value of zero in the parameter file.

* **ElectroStatic**: Considers coulomb interaction or not.
  + Value 1: <BOOLEAN> – True if coulomb interaction needs to be considered and false if not.

*Warning:*

If MARTINI force field was used and charged molecule was used in simulation, **ElectroStatic** needs to be turn on. MARTINI force field uses short range coulomb interaction with constant dielectric 15.0.

* **Dielectric**: Defines dielectric constant for coulomb interaction in MARTINI force field.
  + Value 1: < DOUBLE > – Sets dielectric value used in coulomb interaction.

*Warning:*

In MARTINI force field, **Dielectric** needs to be set to 15.0. If MARTINI force field was chosen and if **Dielectric** was not specified, a default value of 15.0 will be assigned.

* **Ewald**: Considers standard Ewald summation method for electrostatic calculation.
  + Value 1: <BOOLEAN> – “true” if Ewald summation calculation needs to be considered and “false” if not.

*Warning:*

By default, **ElectroStatic** will be set to true if Ewald summation method was used to calculate coulomb interaction.

* **CachedFourier**: Considers storing the reciprocal terms for Ewald summation calculation in order to improve the code performance. This option would increase the code performance with the cost of memory usage.
  + Value 1: <BOOLEAN> – “true” to store reciprocal terms of Ewald summation calculation and “false” if not.

*Warning:*

By default, **CachedFourier** will be set to “true” if no value was set.

* **Tolerance**: Specifies the accuracy of the Ewald summation calculation. Ewald separation parameter and number of reciprocal vectors for the Ewald summation are determined based on the accuracy parameter.
  + Value 1: < DOUBLE > – Sets the accuracy in Ewald summation calculation. A reasonable value for the accuracy is 0.00001.

*Warning:*

If “Ewald” was chosen and no value was set for **Tolerance**, the program will be terminated.

* **PressureCalc**: Considers to calculate the pressure or not. If it is set to true, the frequency of pressure calculation need to be set.
* Value 1: < BOOLEAN > –“True” enabling pressure calculation during the simulation, “false“ disabling pressure calculation.
* Value 2: < ULONG > –The frequency of calculating the pressure.
* **1-4scaling**: Defines constant factor to modify 1-4 short range coulomb interaction.
* Value 1: < DOUBLE > –A fraction number between 0.0 and 1.0 that sets 1-4 scaling factor.

*Warning:*

CHARMM force field uses a value between 0.0 and 1.0. In EXOTIC force field, it needs to be set to 0.0 because 1-4 interaction will not be considered in this force field. In MARTINI force field, it needs to be set to 1.0 because 1-4 interaction will not be modified in this force field.

Example:  
################################

# ELECTROSTATIC

################################

ElectroStatic true

Ewald true

Tolerance 0.0001

1-4scaling 0.0

* **RunSteps**: Sets the total number of steps to run (one move is performed for each step) (cycles = this value / number of molecules in the system)
  + Value 1: <ULONG> – Total run steps
* **EqSteps**: Sets the number of steps necessary to equilibrate the system; averaging will begin at this step.
  + Value 1: <ULONG> – Equilibration steps
* **AdjSteps**: Sets the number of steps per adjustment to the maximum constants associated with each move (e.g. maximum distance in *xyz* to displace, the maximum volume in A3 to swap, etc.)
  + Value 1: <ULONG> – Number of steps per move adjustment   
      
    Example:  
    ################################

# STEPS

################################

RunSteps 25000000

EqSteps 5000000

AdjSteps 1000

* **ChemPot** (For grand canonical (GC) ensemble runs only**):** Chemical potential at which the simulation is run.
  + Value 1: <STRING> – The resname to apply this chemical potential w.r.t.
  + Value 2: <DOUBLE>– The chemical potential value in degrees Kelvin (should be negative).

Note:   
For binary systems, include multiple copies of the tag (one per residue kind).

Example:  
#################################################  
# Mol. Name Chem. Pot.(K)  
#################################################  
ChemPot AR -968

* **DisFreq:** Fractional percentage at which displacement move will occur
  + Value 1: <DOUBLE> – % displace
* **RotFreq:** Fractional percentage at which rigid rotation move will occur
  + Value 1: <DOUBLE> – % rotate
* **IntraSwapFreq:** Fractional percentage at which particle will be removed from a box and inserted into the same box.
  + Value 1: <DOUBLE> – % intra molecule swap
* **VolFreq:** Fractional percentage at which volume displacement move will occur
  + Value 1: <DOUBLE> – (For isobaric-isothermal ensemble and Gibbs ensemble runs only) % of volume swaps
* **SwapFreq:** Fractional percentage at which particle swap move will occur
  + Value 1: <DOUBLE> – (For Gibbs and GC ensemble runs only) % of molecule swaps

Example:  
################################

# MOVE FREQUENCY

################################

DisFreq 0.59

RotFreq 0.10

VolFreq 0.01

SwapFreq 0.20

IntraSwapFreq 0.10

*Note:*

All move percentages should add up to 1.0; otherwise, the program will be terminated.

* **useConstantArea:** Considers to change the volume of the simulation box by fixing the cross-sectional area (x-y plane).
  + Value 1: <BOOLEAN> – (For isobaric-isothermal ensemble and Gibbs ensemble runs only)-“true” volume will change only in z axis, “false” volume will change with constant axis ratio.

*Warning:*

By default, **useConstantArea** will be set to “false” if no value was set. It means, the volume of the simulation box will change in a way to maintain the constant axis ratio.

* **BoxDim**: Defines the axis lengths of simulation box. This tag may occur multiple times. It occurs once for NVT and NPT, but twice for Gibbs ensemble or GC ensemble.
  + Value 1: <INTEGER> – Sets box number (first box is box ‘0’)
  + Value 2: <DOUBLE> – x-axis length in Angstroms
  + Value 3: <DOUBLE> – y-axis length in Angstroms
  + Value 4: <DOUBLE> – z-axis length in Angstroms

*Note:*

If the number of defined boxes were not compatible to simulation type, the program will be terminated.

Example (NVT and NPT ensemble):  
###################################  
# BOX DIMENSION #, X, Y, Z  
###################################  
BoxDim 0 40.00 40.00 40.00   
  
  
Example (Gibbs ensemble or GC ensemble):   
###################################  
# BOX DIMENSION #, X, Y, Z  
###################################  
BoxDim 0 106.60 106.60 106.60  
BoxDim 1 176.60 176.60 176.60

* **CBMC\_First:** Number of CBMC trials to choose the first seed position (Lennard-Jones trials for first seed growth)
  + Value 1: <INTEGER> – Number of initial insertion sites to try
* **CBMC\_Nth**: Number of CBMC trials to choose the later seed positions (Lennard-Jones trials for first seed growth)
  + Value 1: <INTEGER> – Number of LJ trials for growing later atom positions
* **CBMC\_Ang:** Number of CBMC bending angle trials to perform for geometry (per the coupled-decoupled CBMC scheme)
  + Value 1: <INTEGER> – Number of trials per angle
* **CBMC\_Dih:** Number of CBMC dihedral angle trials to perform for geometry (per the coupled-decoupled CBMC scheme)
  + Value 1: <INTEGER> – Number of trials per dihedral.

Example:  
##############################

# CBMC TRIALS

##############################

CBMC\_First 10

CBMC\_Nth 4

CBMC\_Ang 100

CBMC\_Dih 30

*Output Controls*

This section contains all the values that control output in the control file. For example, certain variables control the naming of files dumped of the block-averaged thermodynamic variables of interest, the PDB files, etc.

* **OutputName:** Unique name for simulation used to name the block average, PDB, and PSF output files.
  + **Value 1**: <STRING> – Unique phrase to identify this system  
      
    Example:  
    ##########################################  
    # OUTPUT FILE NAME   
    ##########################################  
    OutputName ISB\_T\_270.00\_K
* **CoordinatesFreq:** Controls output of PDB file (coordinates) and PRNG state. If PDB dumping was enabled, one file for NVT or NPT and two files for Gibbs ensemble or GC ensemble will be dumped.
  + Value 1: <BOOLEAN> –“true” enables dumping these files; “false” disables dumping.
  + Value 2: <ULONG> – Steps per dump PDB frame. It should be less than or equal to **RunSteps**. If this keyword could not be found in configuration file, its value will be assigned a default value to dump 10 frames.

*Note/Warning:*  
The PDB file contains an entry for every **ATOM**, in all boxes read. This allows VMD (which requires a constant number of atoms) to properly parse frames, with a bit of help. Atoms that are not currently in a specific box are given the coordinate (0.00, 0.00, 0.00). The occupancy value corresponds to the box a molecule is currently in (e.g. 0.00 for box 0; 1.00 for box 1).

*Note:*  
At the beginning of simulation, a merged PSF file contains an entry for every **ATOM**, in which all boxes will be dumped. It also contains the topology for every molecule in both boxes, corresponding to the merged PDB format. Loading PDB files into merged PSF file in VMD allows the user to visualize and analyze the results.

* **RestartFreq**: Controls the output of the last state of simulation at a specified step in PDB files (coordinates) that contain “\_restart” phrase. If PDB dumping was enabled, one file for NVT or NPT and two files for Gibbs ensemble or GC ensemble will be dumped.
  + Value 1: <BOOLEAN> –“true” enables dumping these files; “false” disables dumping.
  + Value 2: <ULONG> – Steps per dump last state of simulation to PDB files. It should be less than or equal to **RunSteps**. If this keyword could not be found in the configuration file, **RunSteps** value will be assigned by default.

*Note/Warning:*  
The restart PDB file contains only **ATOM** that exist in each boxes at specified steps. This allows the user to use psfgen and tcl to build a new PSF file and reload it to **RESTART** the simulation.

*Warning:*

**CoordinatesFreq** must be a common multiple of **RestartFreq** or vice versa.

* **ConsoleFreq**: Controls the output to STDIO (“the console”) of messages such as acceptance statistics, and run timing info. In addition, instantaneously-selected thermodynamic properties will be output to this file.
  + Value 1: <BOOLEAN> – “true” enables message printing; “false” disables dumping.
  + Value 2: <ULONG> – Number of steps per print. If this keyword could not be found in the configuration file, the value will be assigned by default to dump 1000 output for **RunSteps** greater than 1000 steps and 100 output for **RunSteps** less than 1000 steps.
* **BlockAverageFreq:** Controls the block averages output of selected thermodynamic properties. Block averages are averages of thermodynamic values of interest for chunks of the simulation (for post-processing of averages or std. dev. in those values).
  + Value 1: <BOOLEAN> – “true” enables printing block average; “false” disables it.
  + Value 2: <ULONG> – Number of steps per block-average output file. If this keyword cannot be found in the configuration file, its value will be assigned a default to dump 100 output.
* **HistogramFreq**: Controls the histograms. Histograms are a binned listing of observation frequency for a specific thermodynamic variable. In this code, they also control the output of a file containing energy/particle samples; it only will be used in GC ensemble simulations for histogram reweighting purposes.
  + Value 1: <BOOLEAN> – “true” enable printing histogram; “false” disables it.
  + Value 2: <ULONG> – Number of steps per histogram output file. If this keyword cannot be found in the configuration file, a value will be assigned by default to dump 1000 output for **RunSteps** greater than 1000 steps and 100 output for **RunSteps** less than 1000 steps.

Example:  
#####################################

# STATISTICS Enable, Freq.

#####################################

CoordinatesFreq true 10000000

RestartFreq true 1000000

ConsoleFreq true 100000

BlockAverageFreq true 100000

HistogramFreq true 10000

The next section controls the output of the energy/particle sample file and the distribution file for particle counts, commonly referred to as the “histogram” output. This section is only required if GC ensemble simulation was used.

* **DistName**: Sets short phrase to naming particle distribution file.
  + Value 1: <STRING> – Short phrase which will be combined with **RunNumber** and **RunLetter** to use in the name of the binned histogram for particle distribution.
* **HistName**: Sets short phrase to naming energy sample file.
  + Value 1: <STRING> – Short phrase, which will be combined with **RunNumber** and **RunLetter,** to use in the name of the energy/particle count sample file.
* **RunNumber**: Sets a number, which is a part of **DistName** and **HistName** file name.
  + Value 1: <UINT> – Run number to be used in the above file names.
* **RunLetter**: Sets a letter, which is a part of **DistName** and **HistName** file name.
  + Value 1: <CHAR> – Run letter to be used in above file names.
* **SampleFreq**: Controls histogram sampling frequency.
  + Value 1: <UINT> – the number of steps per histogram sample.

Example:  
###############################

# OutHistSettings

################################

DistName dis

HistName his

RunNumber 5

RunLetter a

SampleFreq 200

* **OutEnergy**\***\*\*, OutPressure**\***\*\*, OutMolNumber\*\*, OutDensity\*\*, OutVolume\*\*\*, OutSurfaceTension**\***,**: Enables/disables for specific kinds of file output for tracked thermodynamic quantities  
    
  (\*) = NVT ensemble (**\***) = NPT ensemble and Gibbs ensemble (**\***) = GC ensemble
  + Value 1: <BOOLEAN> – “true” enables message output of block averages via this tracked parameter (and in some cases such as entry, components); “false” disables it.
  + Value 2: <BOOLEAN> – “true” enables message output of a fluctuation into the console file via this tracked parameter (and in some cases, such as entry, components); “false” disables it.

Example:  
######################################  
# ENABLE: BLK AVF., FLUCK.  
######################################  
OutEnergy true true   
OutPressure true true   
OutMolNum true true   
OutDensity true true

OutVolume true true

OutSurfaceTention false false

**Sample Files**

Here’s a sample of how a typical NVT ensemble control file might be expected to look:

#==================================================  
#= GOMC beta 2.00 – NVT Control example =  
#==================================================  
  
#######################################

##=====-------- INPUT --------=====  
#######################################

#########################

# ENABLE, STEP

#########################

Restart false

####################################

# KIND {RESTART, RANDOM, INTSEED}

####################################

PRNG RANDOM

####################################

# FORCE FIELD

####################################

ParaTypeCHARMM true

Parameters ./../common/Par\_MODEL\_NAME.inp

####################################

# INPUT PDB FILES

####################################

Coordinates 0 STEP3\_START\_XX\_PHA\_BOX\_0.pdb

####################################

# INPUT PSF FILES

####################################

Structure 0 STEP3\_START\_XX\_PHA\_BOX\_0.psf

###########################################################

# =======------------- SYSTEM ----------------=========

###########################################################

#############################

# SIMULATION CONDITION

#############################

Temperature 200.00

Potential VDW

LRC true

Rcut 10

Exclude 1-4

#############################

# ELECTROSTATIC

#############################

ElectroStatic true

Ewald false

Tolerance 0.0001

1-4scaling 0.0

################################

# STEPS

################################

RunSteps 25000000

EqSteps 5000000

AdjSteps 1000

################################

# MOVE FREQUENCY

################################

DisFreq 0.80

RotFreq 0.10

IntraSwapFreq 0.10

################################

# PRESSURE CALCULATION

################################

PressureCalc true 10000

################################

# BOX DIMENSION #, X, Y, Z

################################

BoxDim 0 35.00 35.00 35.00

##############################

# CBMC TRIALS

##############################

CBMC\_First 10

CBMC\_Nth 4

CBMC\_Ang 100

CBMC\_Dih 30

###########################################################

# =======--------------- OUTPUT --------------=========

###########################################################

##########################

# OUTPUT FILE NAME

##########################

OutputName AR\_200\_00\_K\_2220\_r5

#####################################

# STATISTICS ENABLE, FREQ.

#####################################

CoordinatesFreq true 1000000

RestartFreq true 1000000

ConsoleFreq true 1000000

BlockAverageFreq true 1000000

########################################

# ENABLE: BLK AVF., FLUCK.

########################################

OutEnergy true true

OutPressure true true

OutMolNum false false

OutDensity false false

Here’s a sample of how a typical Gibbs ensemble control file might be expected to look:

#=============================================================  
#= GOMC beta 2.00 – Gibbs ensemble Control example =  
#=============================================================

#########################

# ENABLE, STEP

#########################

Restart false

####################################

# KIND {RESTART, RANDOM, INTSEED}

####################################

PRNG INTSEED

Random\_Seed 49

####################################

# FORCE FIELD

####################################

ParaTypeCHARMM false

ParaTypeEXOTIC true

Parameters Par\_MODEL\_NAME.inp

####################################

# INPUT PDB FILES

####################################

Coordinates 0 STEP3\_START\_XX\_PHA\_BOX\_0.pdb

Coordinates 1 STEP3\_START\_XX\_PHA\_BOX\_1.pdb

####################################

# INPUT PSF FILES

####################################

Structure 0 STEP3\_START\_XX\_PHA\_BOX\_0.psf

Structure 1 STEP3\_START\_XX\_PHA\_BOX\_1.psf

######################################################################

# =======------------------- SYSTEM --------------------==========

######################################################################

##################################

# GEMC TYPE (DEFULT IS NVT\_GEMC)

##################################

GEMC NPT

Pressure 2.50

#############################

# SIMULATION CONDITION

#############################

Temperature 200.00

Potential SWITCH

LRC false

Rswitch 7

Rcut 10

RcutLow 1.0

Exclude 1-4

#############################

# ELECTROSTATIC

#############################

Ewald false

ElectroStatic false

################################

# STEPS

################################

RunSteps 25000000

EqSteps 5000000

AdjSteps 1000

################################

# MOVE FREQUENCY

################################

DisFreq 0.69

RotFreq 0.10

VolFreq 0.01

SwapFreq 0.20

IntraSwapFreq 0.00

################################

# PRESSURE CALCULATION

################################

PressureCalc true 10000

################################

# BOX DIMENSION #, X, Y, Z

################################

BoxDim 0 35.00 35.00 35.00

BoxDim 1 45.00 45.00 45.00

##############################

# CBMC TRIALS

##############################

CBMC\_First 8

CBMC\_Nth 4

CBMC\_Ang 100

CBMC\_Dih 30

######################################################################

# =======-------------------- OUTPUT --------------------=========

######################################################################

##########################

# OUTPUT FILE NAME

##########################

OutputName AR\_200\_00\_K\_2220\_r5

#####################################

# STATISTICS ENABLE, FREQ.

#####################################

CoordinatesFreq true 1000000

RestartFreq true 1000000

ConsoleFreq true 1000000

BlockAverageFreq true 1000000

##################################

# ENABLE: BLK AVF., FLUCK.

##################################

OutEnergy true true

OutPressure true true

OutMolNum true true

OutDensity true true

Here’s a sample of how a typical grand canonical ensemble control file might be expected to look:

#=============================================================  
# GOMC beta 2.00 – Grand canonical ensemble Control example =  
#=============================================================

#########################

# ENABLE, STEP

#########################

Restart false

####################################

# KIND {RESTART, RANDOM, INTSEED}

####################################

PRNG RANDOM

####################################

# FORCE FIELD

####################################

ParaTypeCHARMM false

ParaTypeEXOTIC true

Parameters Par\_MODEL\_NAME.inp

####################################

# INPUT PDB FILES

####################################

Coordinates 0 STEP3\_START\_XX\_PHA\_BOX\_0.pdb

Coordinates 1 STEP3\_START\_XX\_PHA\_ Reserv\_BOX\_1.pdb

####################################

# INPUT PSF FILES

####################################

Structure 0 STEP3\_START\_XX\_PHA\_BOX\_0.psf

Structure 1 STEP3\_START\_XX\_PHA\_Reserv\_BOX\_1.psf

######################################################################

# =======------------------- SYSTEM --------------------==========

######################################################################

#############################

# SIMULATION CONDITION

#############################

Temperature 200.00

Potential VDW

LRC true

Rcut 10

Exclude 1-4

#############################

# ELECTROSTATIC

#############################

Ewald true

ElectroStatic true

CachedFourier true

Tolerance 0.00001

1-4scaling 0.0

################################

# STEPS

################################

RunSteps 25000000

EqSteps 5000000

AdjSteps 1000

################################

# MOVE FREQUENCY

################################

DisFreq 0.20

RotFreq 0.20

SwapFreq 0.60

################################

# PRESSURE CALCULATION

################################

PressureCalc false

################################

# BOX DIMENSION #, X, Y, Z

################################

BoxDim 0 35.00 35.00 35.00

BoxDim 1 45.00 45.00 45.00

##############################

# CBMC TRIALS

##############################

CBMC\_First 8

CBMC\_Nth 4

CBMC\_Ang 100

CBMC\_Dih 30

####################################

# Mol. Name Chem. Pot.

####################################

ChemPot DME -2230

######################################################################

# =======-------------------- OUTPUT --------------------=========

######################################################################

##########################

# OUTPUT FILE NAME

##########################

OutputName DME\_200\_00\_K\_2220

#####################################

# STATISTICS ENABLE, FREQ.

#####################################

CoordinatesFreq true 1000000

RestartFreq true 1000000

ConsoleFreq true 1000000

BlockAverageFreq true 1000000

HistogramFreq true 100000

################################

# OutHistSettings

################################

DistName dis

HistName his

RunNumber 5

RunLetter a

SampleFreq 200

##################################

# ENABLE: BLK AVF., FLUCK.

##################################

OutEnergy true true

OutPressure false false

OutMolNum true true

OutDensity false false

OutSurfaceTension false false

**GOMC’s Output Files, Terminal Output**

GOMC currently supports several kinds of output:

* STDIO (“console”) Output
* File Output
  + PDB
  + PSF
  + Block Averages

GOMC output units:

|  |  |  |  |
| --- | --- | --- | --- |
| **Properties** | **Units** | **Properties** | **Units** |
| Energy | K | Volume | Å3 |
| Pressure, Pressure Tensor | bar | Density | Kg/m3 |
| Heat of vaporization | KJ/mol | Surface Tension | mN/m |

*Console Output*

A variety of useful information relating to instantaneous statistical and thermodynamic data (move trials, acceptance rates, file I/O messages warnings, and other kinds of information) is printed to the STDIO, which, in Linux, will typically be displayed in the terminal. This output can be redirected into a log file in Linux using the ‘>’ operator.

Statistical and thermodynamic information is provided in console output.

* Energy
  + Intermolecular (LJ)
  + Intramolecular bonded
  + Intramolecular nonbonded
  + Tail corrections
  + Electrostatic real
  + Electrostatic Reciprocal
  + Electrostatic self
  + Electrostatic correction
  + Total electrostatic energy (sum of real, reciprocal, self, and correction)
  + Total Energy (sum of the all energies)
* Pressure, Pressure Tensor (Pxx, Pyy, Pzz).
* Volume
* Total molecule number
* Total Density
* Surface Tension
* Mole fraction of each species

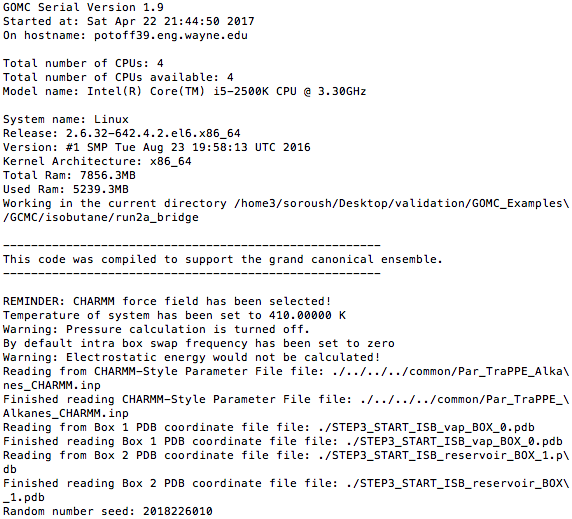
Detailed move, energy, and statistical or thermodynamic information for each simulation box will be printed in three different sections. Each section’s title will start with MTITLE, ETITLE, and STITLE for move, energy, and statistical information, respectively. The instantaneous values for each section will start with MOVE\_#, ENER\_#, and STAT\_# for move, energy, and statistical values, respectively. Where, # is the simulation box number. In addition, if pressure calculation is activated and enabled to print, pressure tensor will be printed in the console output file. This section starts with PRES\_# and print the diagonal value of pressure tensor Pxx, Pyy, and Pzz, respectively. The second element after the title of each section is the step number.

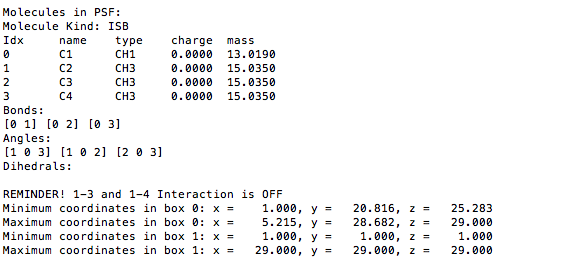
In order to extract the desired information from the console file, “grep” and “awk” commands can be used with a proper title section. For example, in order to extract total energy of the system, the following command needs to be executed in terminal:

grep “ENER\_0” output\_console.log | awk '{print $3}'

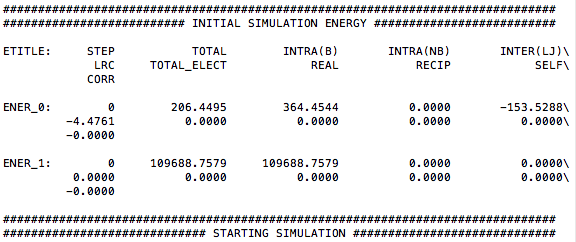
Here, “output\_console.log” is the console output file and “$3” represents the second element of the “ENERGY\_BOX\_0” section.

The first section of this console output typically includes some info relating to the system, CPU, and RAM. In continue, console output includes information regarding the input file (configuration file) reading, force field reading, summary of the topology of the molecule, and minimum and maximum coordinate of molecules. This output is important; it may contain text relating to issues encountered if there was an error in the current run (e.g. a bad parameter, unknown keyword, missing parameters in the configuration file, etc.)

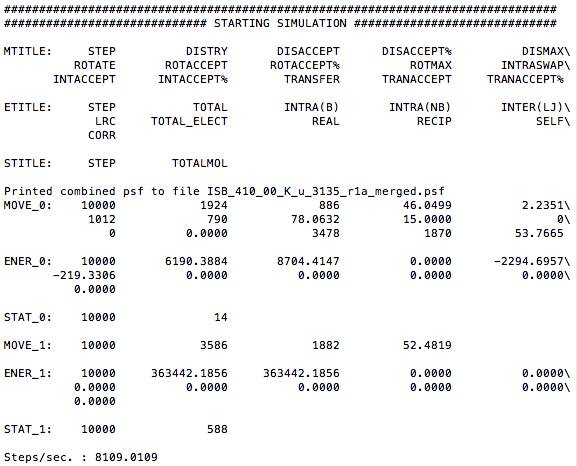




Next, the energy title and initial energy of the system’s starting configuration will print:



After the simulation starts, move, energy, and statistical title, followed by their values for each simulation box, will print:



At the end of the run, timing information and other wrap up info will be printed.

*Note:*  
 Printed energy and statistical values are instantaneous values.

*Note:*  
 It’s important to watch the acceptance rates and adjust the move percentages and CBMC trial amounts to get the desired rate of move acceptance.

*PDB and PSF Files*  
  
The PDB and PSF output (merging of atom entries) has already been mentioned/explained in previous sections. To recap: The PDB file’s **ATOM** entries’ occupancy is used to represent the box the molecule is in for the current frame. All molecules are listed in order in which they were read (i.e. if box 0 has 1..N1 molecules and box 1 has 1..N­2 molecules, then all of the molecules in box 0 are listed first and all the molecules in box 1, i.e. 1..N1, N1+1..N1+N2). PDB frames are written as standard PDBs to consecutive file frames.

To visualize, open the PDB and PSF files outpout by VMD and use the command in the terminal:  
  
  
  
Use the TKConsole to get the current frame to obey the periodicity. Some peculiarities will still be observed, so please review the PDB file if anything looks odd (e.g. bonds will sometimes not show up in later frames). The GOMC team is in the process of improving VMD compatibility, but this may require changes to VMD to allow the files output to be read properly.

pbc join connected

For Gibbs ensemble, the same PSF will be used to visualize either box, i.e.  
  
  
…for visualizing the first box and…

vmd ISB\_T\_330.00\_K\_Run\_0\_MIE\_BOX\_1.pdb ISB\_T\_330.00\_K\_Run\_0\_MIE\_merged.pdb

vmd ISB\_T\_330.00\_K\_Run\_0\_MIE\_BOX\_0.pdb ISB\_T\_330.00\_K\_Run\_0\_MIE\_merged.pdb

... for visualizing the second box.

*Block Output Files*

GOMC tracks a number of thermodynamic variables of interest during the simulation and prints them all in one file for each box.

* Energy
  + Intermolecular (LJ)
  + Intramolecular bonded
  + Intramolecular nonbonded
  + Tail corrections
  + Electrostatic real
  + Electrostatic Reciprocal
  + Total Energy (sum of the all energies)
* Virial
* Pressure
* Surface Tension (using virial method)
* Volume
* Total molecule number
* Total Density
* Mole fraction of each species
* Heat of vaporization

At the beginning of each file, the title of each property followed by their average values is printed. Desired data can be extracted, as explained before, using the “awk” command. For example, in order to extract total density of the system, the following command need to be executed in terminal:

cat Blk\_OUTPUTNAME\_BOX\_0.dat | awk '{print $13}'

Here, “Blk\_OUTPUTNAME\_BOX\_0.dat” is the block-average file for simulation box 0 and “$13” represents the 13th column of the block file.

**Putting it all Together: Running a GOMC Simulation**

It is strongly recommended that you download the test system provided at <http://gomc.eng.wayne.edu/downloads.html> or

<https://github.com/GOMC-WSU/GOMC_Examples/tree/master>

run different simulation types in order to become more familiar with different parameter and configuration files (\*.conf).

To recap the previous examples, a simulation of isobutane will be completed for a single temperature point on the saturated vapor-liquid coexistence curve.

The general plan for running the simulation is:

1. Build GOMC (if not done already)
2. Copy GOMC executable to build directory
3. Create scripts, PDB, and topology file to build the system, plus in.dat file and parameter files to prepare for runtime
4. Build finished PDBs and PSFs using the simulation.
5. Run the simulation in the terminal.
6. Analyze the output.

Please, complete steps 1 and 2; then, traverse to the directory, which should now contain a single file “GOMC\_Serial\_GEMC”.

Next, six files need to be made:

* PDB file for isobutane
* Topology file describing isobutane residue
* Two \*.inp packmol scripts to pack two system boxes
* Two TCL scripts to input into PSFGen to generate the final configuration

isobutane.pdb :

REMARK 1 File created by GaussView 5.0.8  
ATOM 1 C1 ISB 1 0.911 -0.313 0.000 C  
ATOM 2 C2 ISB 1 1.424 -1.765 0.000 C  
ATOM 3 C3 ISB 1 -0.629 -0.313 0.000 C  
ATOM 4 C4 ISB 1 1.424 0.413 -1.257 C  
END

Top\_Branched\_Alkane.inp:

\* Custom top file -- branched alkanes  
\*

MASS 1 CH3 15.035 C !

MASS 3 CH1 13.019 C !

RESI ISB 0.00 ! isobutane – TraPPE  
GROUP  
ATOM C1 CH1 0.00 ! C3  
ATOM C2 CH3 0.00 ! C2-C1  
ATOM C3 CH3 0.00 ! C4  
ATOM C4 CH3 0.00 !  
BOND C1 C2 C1 C3 C1 C4  
PATCHING FIRS NONE LAST NONE  
END

pack\_box\_0.inp:

tolerance 3.0  
filetype pdb  
output STEP2\_ISB\_packed\_BOX\_0.pdb  
  
structure isobutane.pdb  
number 1000   
inside box 0. 0. 0. 68.00 68.00 68.00  
end structure

pack\_box\_1.inp:   
  
tolerance 3.0  
filetype pdb  
output STEP2\_ISB\_packed\_BOX\_1.pdb  
  
structure isobutane.pdb  
number 1000  
inside box 0. 0. 0. 68.00 68.00 68.00  
end structure

build\_box\_0.inp:

psfgen << ENDMOL  
topology ./Top\_Branched\_Alkane.inp  
segment ISB {  
 pdb ./STEP2\_ISB\_packed\_BOX\_0.pdb  
 first none  
 last none  
}

coordpdb ./STEP2\_ISB\_packed\_BOX\_0.pdb ISB

writepsf ./STEP3\_START\_ISB\_sys\_BOX\_0.psf  
writepdb ./STEP3\_START\_ISB\_sys\_BOX\_0.pdb

build\_box\_1.inp:

psfgen << ENDMOL  
topology ./Top\_Branched\_Alkane.inp  
segment ISB {  
 pdb ./STEP2\_ISB\_packed\_BOX\_1.pdb  
 first none  
 last none  
}

coordpdb ./STEP2\_ISB\_packed\_BOX\_1.pdb ISB

writepsf ./STEP3\_START\_ISB\_sys\_BOX\_1.psf  
writepdb ./STEP3\_START\_ISB\_sys\_BOX\_1.pdb

These files can be created with a standard Linux or Windows text editor. Please, also copy a Packmol executable into the working directory.

Once those files are created, run in the terminal:

./packmol < pack\_box\_0.inp  
./packmol < pack\_box\_1.inp

This will create the intermediate PDBs.

Then, run the PSFGen scripts to finish the system using the following commands:

VMD < ./build\_box\_0.inp  
VMD < ./build\_box\_1.inp

This will create the intermediate PDBs.

To run the code a few additional things will be needed:

* A GOMC Gibbs ensemble executable
* A control file
* Parameter files.

Enter the control file (**in.conf**) in the text editor in order to modify it. Example files for different simulation types can be found in previous section.

Once these four files have been added to the output directory, the simulation is ready.

Assuming the code is named GOMC\_Serial\_GEMC, run in the terminal using:

./GOMC\_Serial\_GEMC in.conf > out\_ISB\_T\_330.00\_K\_RUN\_0.log &

For running GOMC in parallel, using openmp, run in the terminal using:

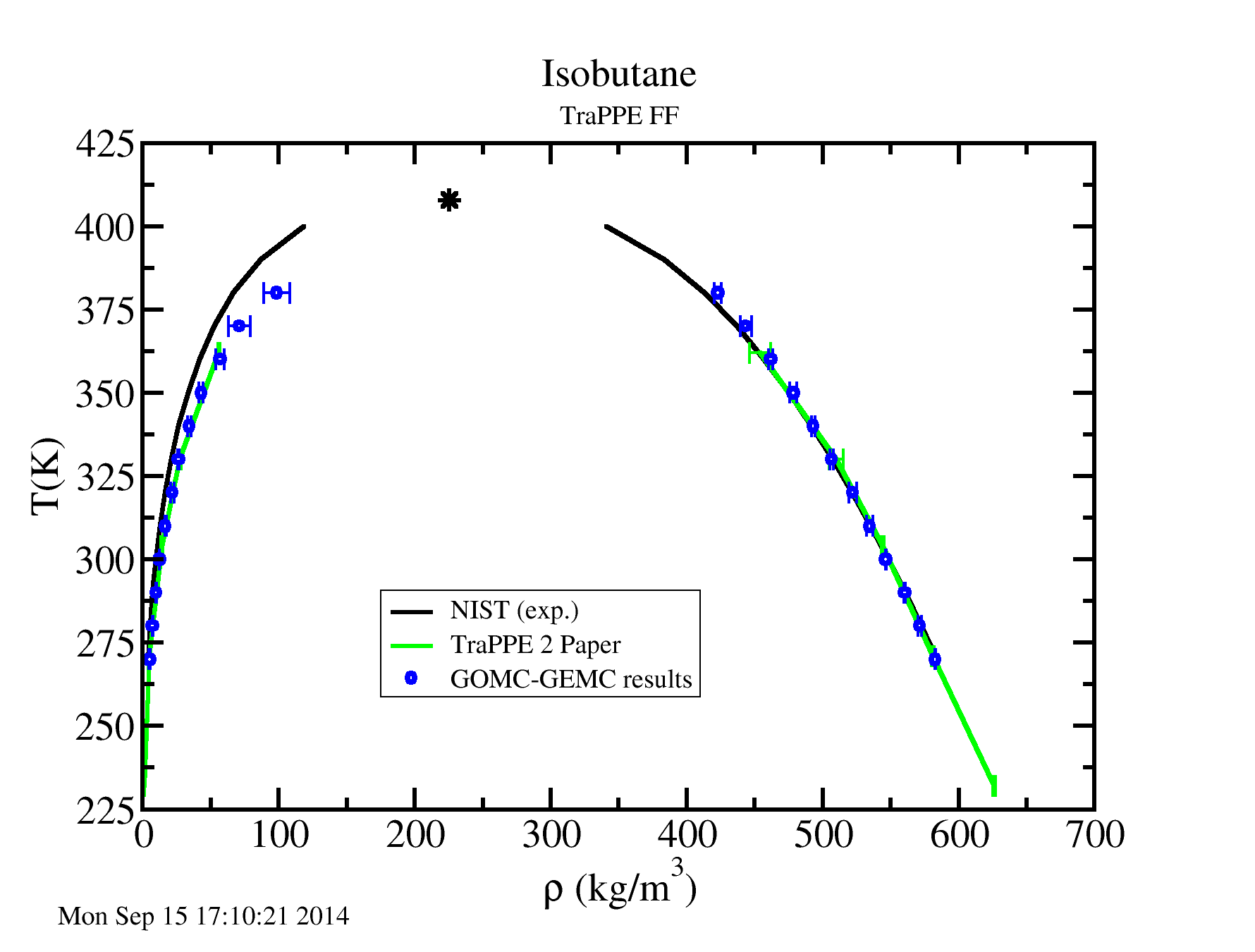
./GOMC\_Serial\_GEMC +p4 in.conf > out\_ISB\_T\_330.00\_K\_RUN\_0.log &

Here, 4 defines the number of processors that will be used to run the simulation in parallel.

Progress can be monitored in the terminal with the tail command:

tail –f out\_ISB\*log

Congratulations! You have examined a single-phase coexistence point on the saturated vapor-liquid curve using GOMC operating in the Gibbs ensemble.



Repeating this process for multiple temperatures will allow you to obtain the following results.

**Intermolecular Energy and Virial function:**

* **VDW**: This option calculates potential energy without any truncation.
* Potential calculation: Interactions between atoms can be modeled with an n − 6 potential, a Mie potential in which the attractive exponent is fixed. The Mie potential can be viewed as a generalized version of the 12-6 Lennard-Jones potential,

where rij, εij, and σij are, respectively, the separation, well depth, and collision diameter for the pair of interaction sites i and j. The constant Cn is a normalization factor such that the minimum of the potential remains at −εij for all nij. In the 12-6 potential, Cn reduces to the familiar value of 4.

* Virial calculation: Virial is basically the negative derivative of energy with respect to distance, multiplied by distance.

Using n-6 LJ potential defined above:

Note:

This option only evaluates the energy up to specified Rcut distance. Tail correction to energy and pressure can be specified to account for infinite cutoff distance.

* **SHIFT**: This option forces the potential energy to be zero at Rcut distance.
  + Potential calculation: Interactions between atoms can be modeled with an n − 6 potential,

where rij, εij, and σij are, respectively, the separation, well depth, and collision diameter for the pair of interaction sites i and j. The constant Cn is a normalization factor such that the minimum of the potential remains at −εij for all nij. In the 12-6 potential, Cn reduces to the familiar value of 4.

* Virial calculation: Virial is basically the negative derivative of energy with respect to distance, multiplied by distance.

Using SHIFT potential function defined above:

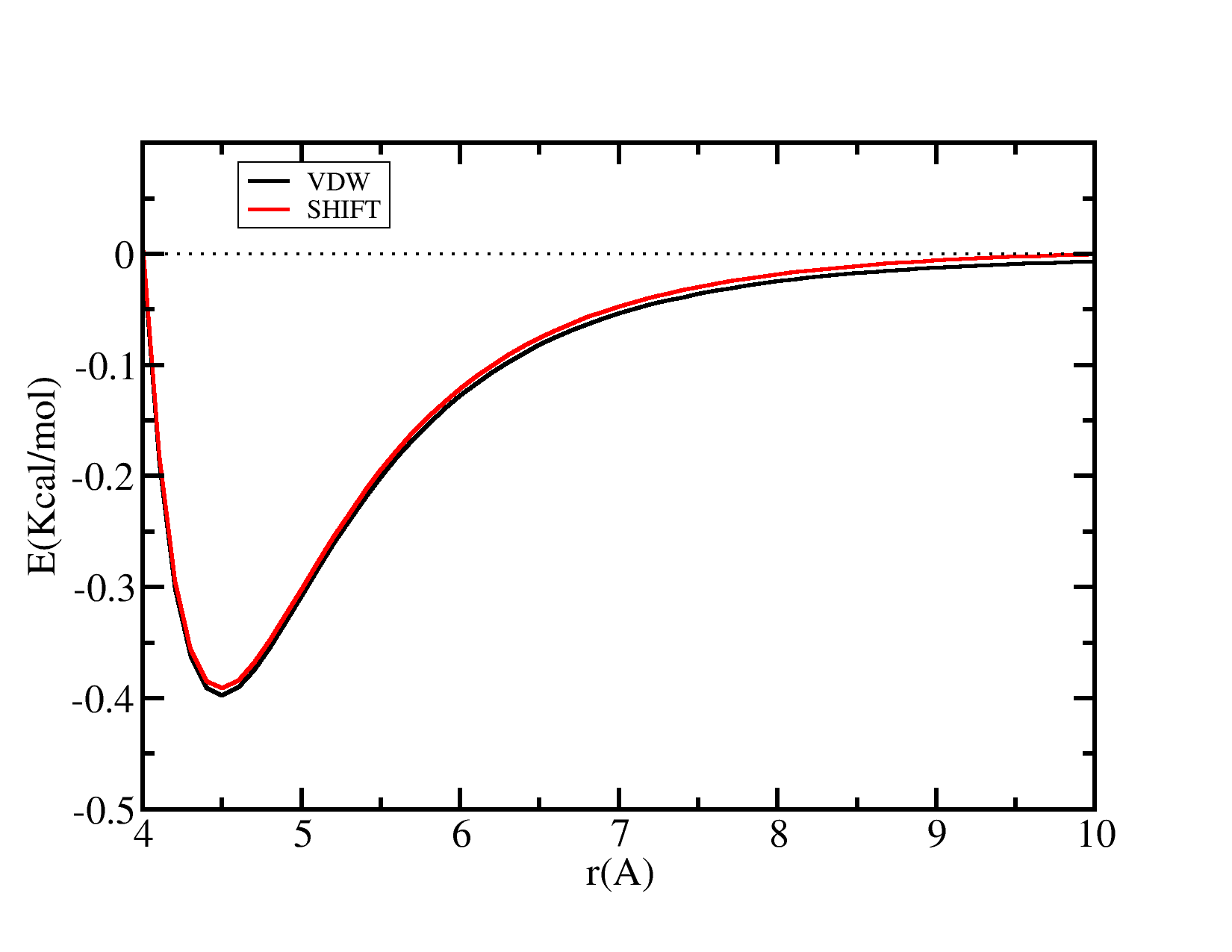


Figure 1: Graph of van der Waals potential with and without the application of the SHIFT function. With the SHIFT function active, the potential by force was reduced to 0.0 at the Rcut distance. With the SHIFT function, there is a discontinuity where the potential is truncated.

* **SWITCH**: This option in CHARMM or EXOTIC force field smoothly forces the potential energy to be zero at Rcut distance and starts modifying the potential at Rswitch distance.
  + Potential calculation: Interactions between atoms can be modeled with an n − 6 potential,

where rij, εij, and σij are, respectively, the separation, well depth, and collision diameter for the pair of interaction sites i and j. The constant Cn is a normalization factor used such that the minimum of the potential remains at −εij for all nij. In the 12-6 potential, Cn reduces to the familiar value of 4.

The factor FE is defined as:

* Virial calculation: Virial is basically the negative derivative of energy with respect to distance, multiplied by distance.

Using the SWITCH potential function defined above:

The factor FW is defined as:

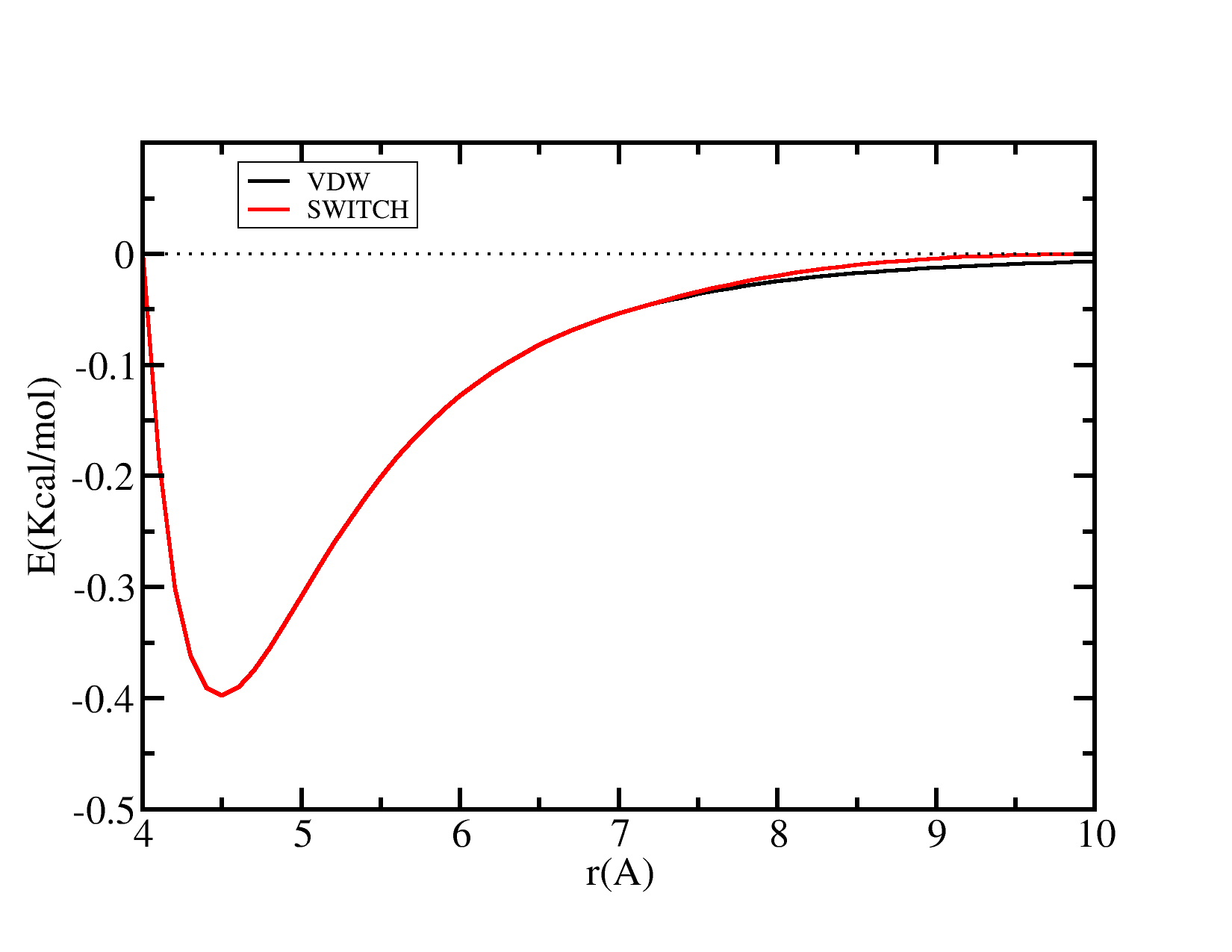


Figure 2: Graph of van der Waals potential with and without the application of the SWITCH function. With the SWITCH function active, the potential is smoothly reduced to 0 at the Rcut distance.

* **SWITCH**: This option in MARTINI force field smoothly forces the potential energy to be zero at Rcut distance and starts modifying the potential at Rswitch distance.
  + Potential calculation: Interactions between atoms. Rswitch is set to 0.0 for coulomb interaction.

The factor is defined as:

* Virial calculation: Virial is basically the negative derivative of energy with respect to distance, multiplied by distance.

Using the SWITCH potential function defined above:

The factor is defined as:

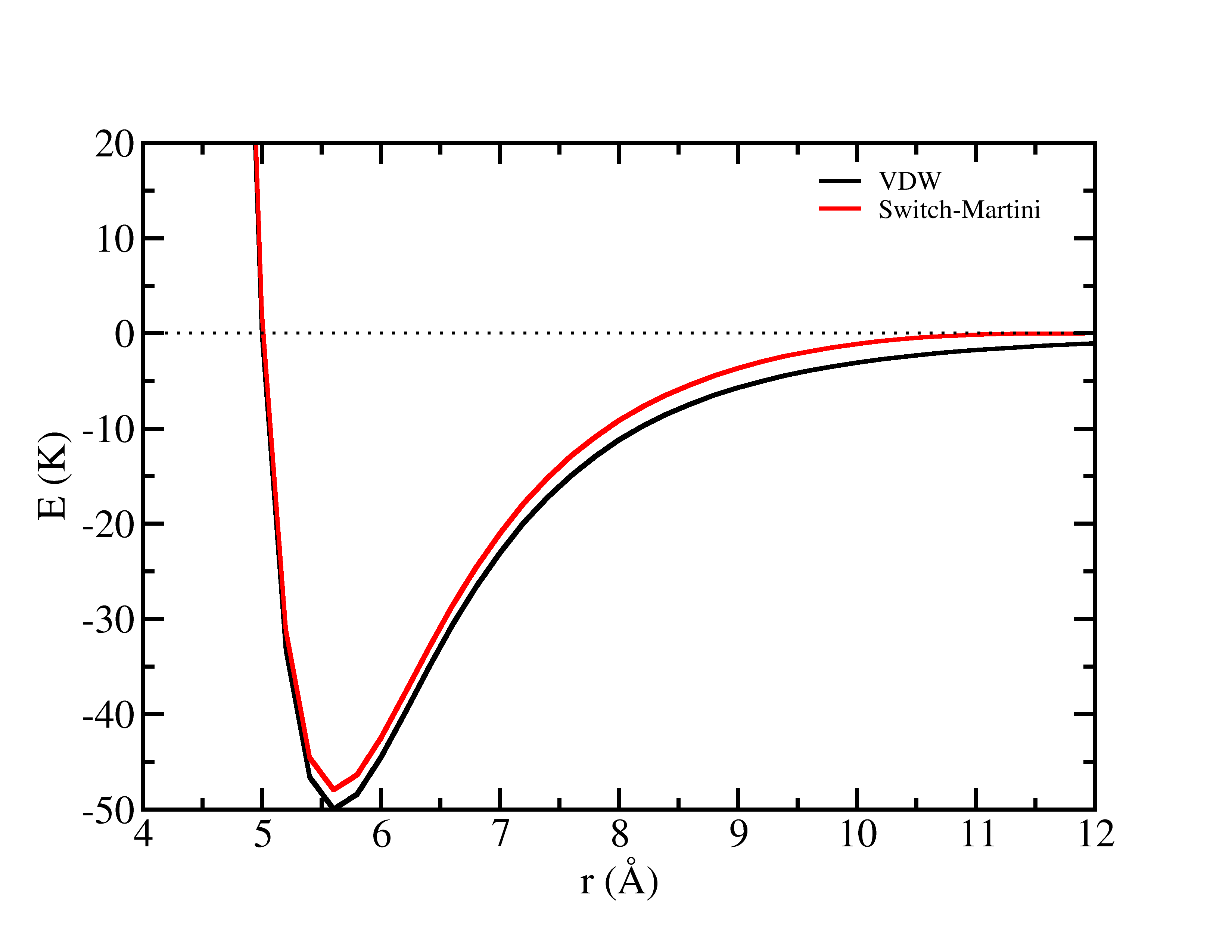
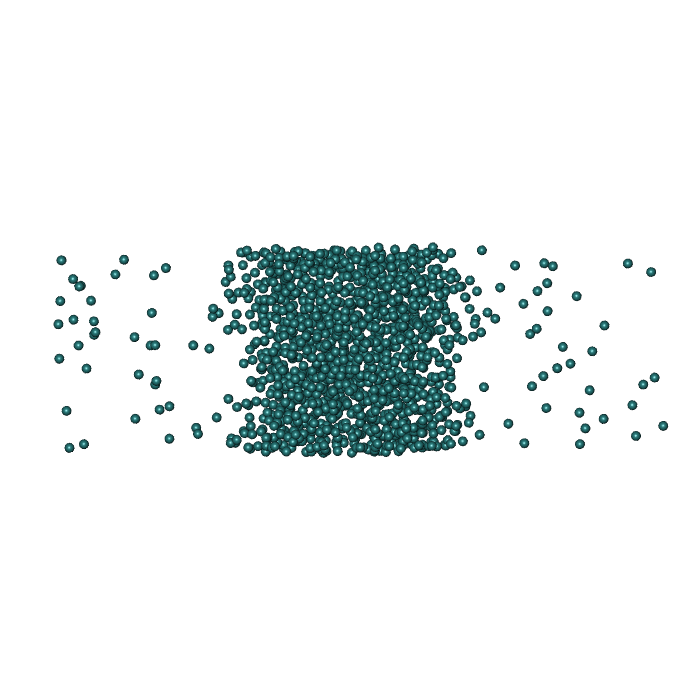


Figure 3: Graph of van der Waals potential with and without the application of the SWITCH function in MARTINI force field. With the SWITCH function active, the potential is smoothly reduced to 0.0 at the Rcut distance.

* **SurfaceTension**: Surface tension will be calculated using virial method according to following equation:



**How to Get Help/Technical Support**

Please, send an email to ***gomc@eng.wayne.ed***u