GOMC

USER'S MANUAL

version 0.96

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
   a. Required compilers, libraries, programs (this includes psfgen and tcl which are needed to build the PSF files)
   b. Supported operating systems
   c. Serial
   d. GPU
4. Input file formats
   a. PDB, PSF, parameter, control file (define each part of the files here)
5. Output file formats
   a. Working with VMD
6. System setup
   a. Building initial coordinates with Packmol
   b. Creating a PSF file
   c. Creating the control file
7. Running the simulation
8. Examples
9. How to get help/technical support.
10. Summary of algorithms used in the code

Tutorial Overview

This document will introduce a new user to how to download, compile and run the GOMC molecular simulation code. A working knowledge of statistical physics is recommended as a perquisite to understanding this tutorial.

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

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

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

Introduction (what is GOMC?)

Monte Carlo (MC) simulations are a kind of simulations that are driven by stochastic processes. The "GO" stands for GPU Optimized, as 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 both canonical and Gibbs ensemble simulations. Support for GCMC and GEMC-NPT will be added shortly. GOMC uses widely used simulations file types (PDB, CHARMM-style parameter file, PSF). GOMC includes configurational bias algorithms for both linear and branched systems. Support for cyclic molecules and charged systems is being added.

How to get the software

Currently the latest public code builds, the project logo, manual, and other resources can be obtained via the website:

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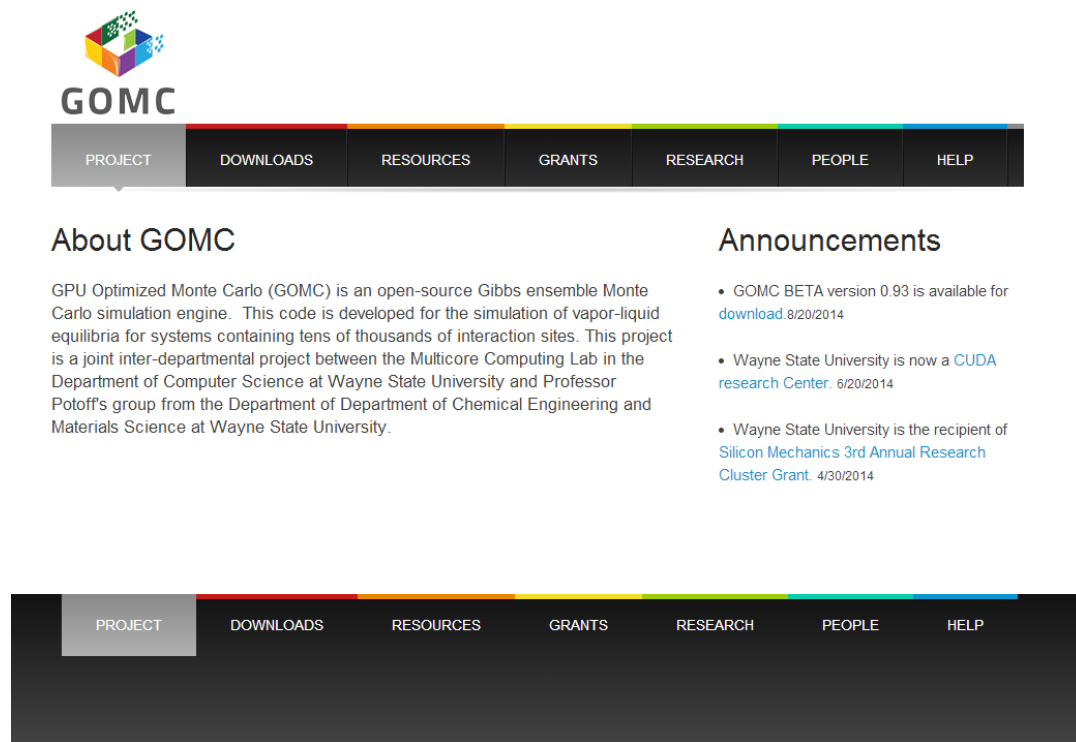
The code can be found under the download tab, just below and to the right of the logo. When new betas (or eventually release builds) are announced, they will replace the prior code under the downloads tab. An announcement will be posted on the front page to notify users.

Currently version control is handled through an internal SVN system maintained by the developers at Wayne State University. The posted builds are "frozen" versions of the code that have been validated for a number of systems and ensembles. Eventually the project will be hosted as a Git repository to allow for greater collaboration and a faster means of spotting new releases.

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 may compile on recent versions of **OS X**, but that platform is not officially supported.

Required Software Prequisites

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

1. C++03 Compliant Compiler
    a. Linux/OS X
        i. icc (Intel c++ compiler)
           Type…

           ```
           icc --version
           ```

           …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).

        ii. g++ (GNU GCC)
            Type…

            ```
            g++ --version
            ```

            …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.
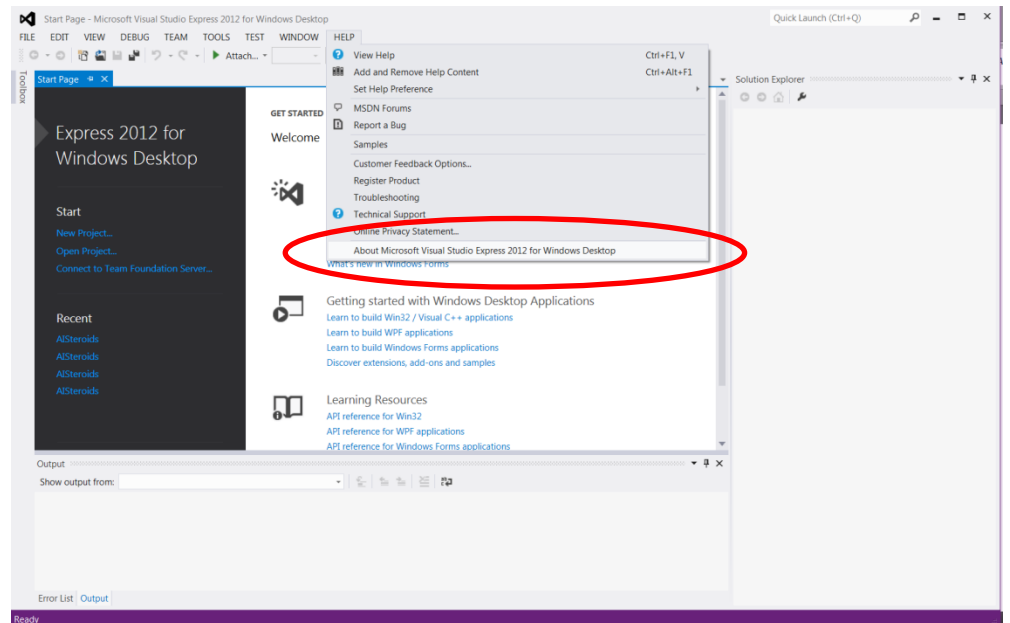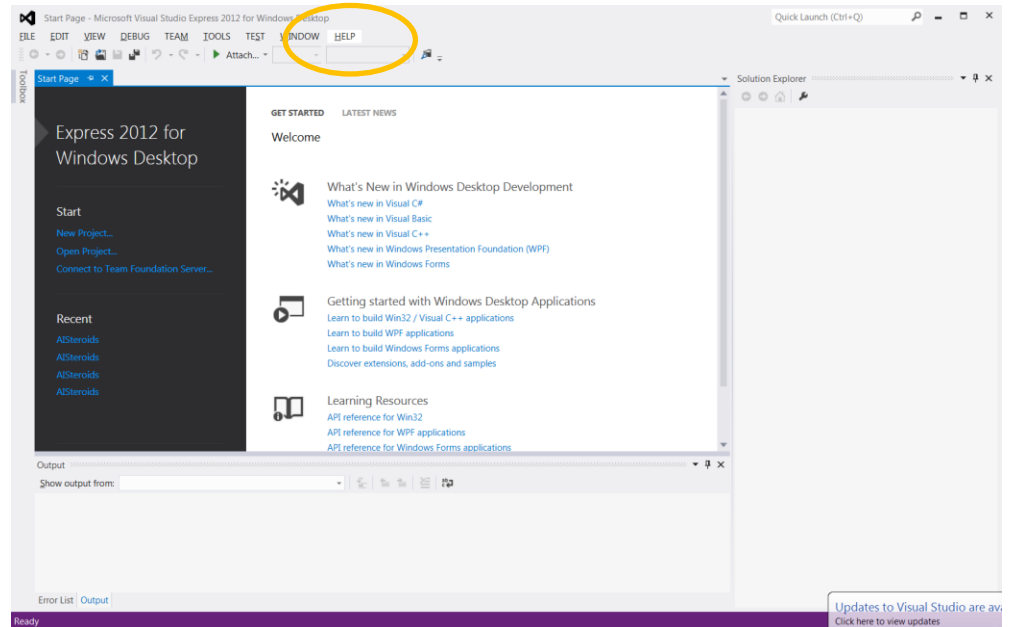
b. Windows
    i. Visual Studio

Microsoft's Visual Studio 2010 or later is recommended.

Check version:

Help (top tab) → *About Microsoft Visual Studio*

2. cmake (if compiling on Linux)

   To check if cmake is installed

   ```
   which cmake
   ```

   To check version number:

   ```
   cmake --version
   ```

   Here's an example from one of our systems:

3. nvcc/CUDA libs

   The GPU builds of the code requires NVIDIA's CUDA 6.0 or newer...
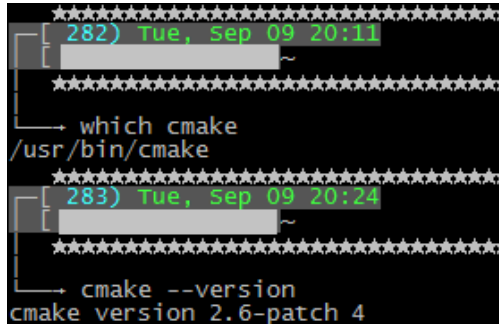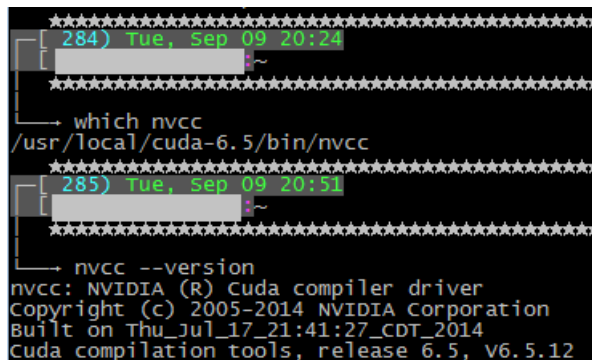
   To check if nvcc is installed

   ```
   which nvcc
   ```

   To check version number:

   ```
   nvcc --version
   ```

   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.

To download CUDA visit NVIDIA's webpage:
https://developer.nvidia.com/cuda-downloads
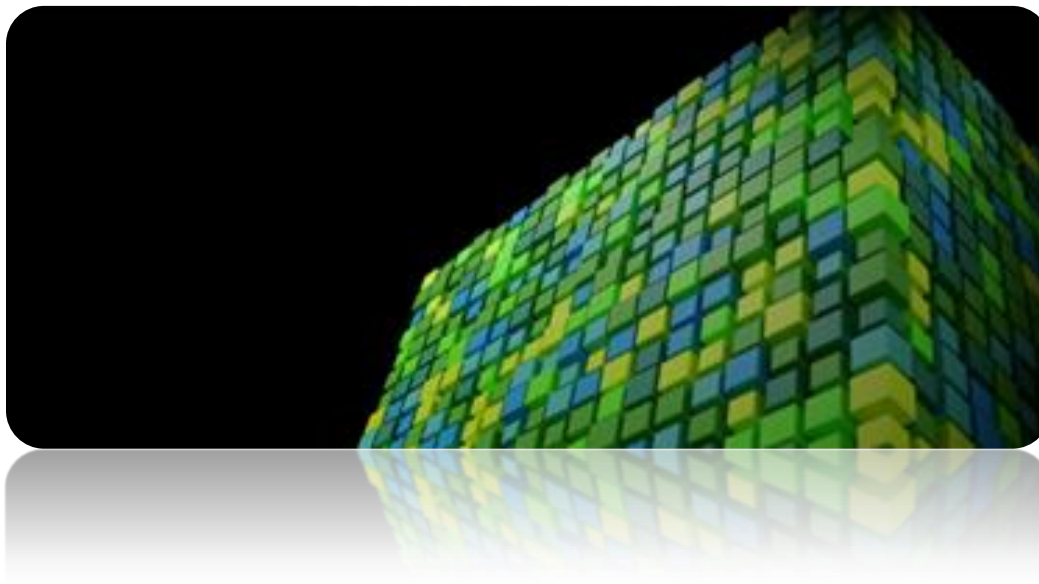
CUDA is required to compile the GPU executable in both Windows and Linux. Please refer to CUDA Developer webpages to select an appropriate version for the desired platform.

To install CUDA in Linux *root*/*sudo* privileges are generally required. In Windows, administrative access is required.

Recommended Software Tools

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 source and binaries are available free to download. It comes with a robust scripting engine capable of running python and tcl scripts.

More info can be found out here:

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

GOMC uses the same fundamental file types – PDB (coordinates) and PSF (topology) as VMD, although it uses some special tricks to obey certain rules of those file formats.

One useful purpose of VMD is visualization of your systems.
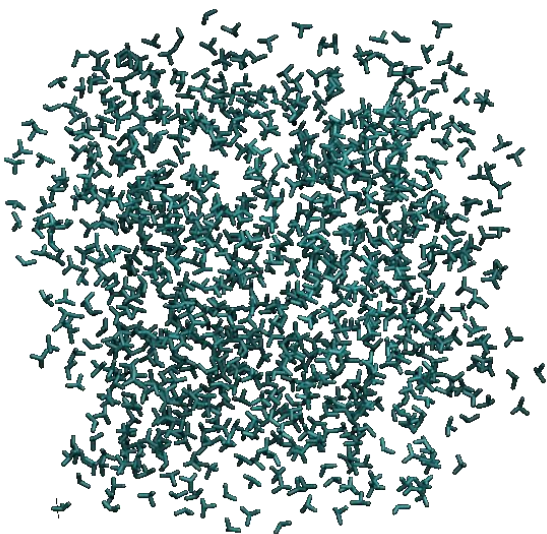
(A system of united atom isobutane molecules is seen above.)

The most critical part of VMD, though 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.
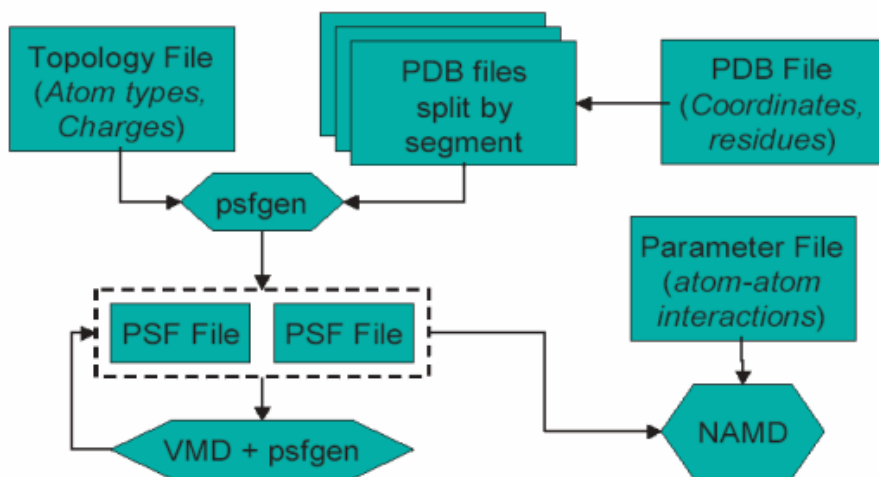
FIGURE: An overview of the PSFGen file generation process and its relationship to VMD/NAMD

To read more about PSFGen, please see:

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 its 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 automatically come with Fortran compilers, so this may need to be installed.

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

WARNING
Another more seriously 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 2 to 3 Angstroms smaller than the simulation box size. This prevents hard overlaps over the periodic boundary.

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, although 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. This makes it ideal for visualizing much of the thermodynamic data from the GOMC engine. Here is an example of the results of simulations of saturated VLE densities of linear alkanes produced with Grace:
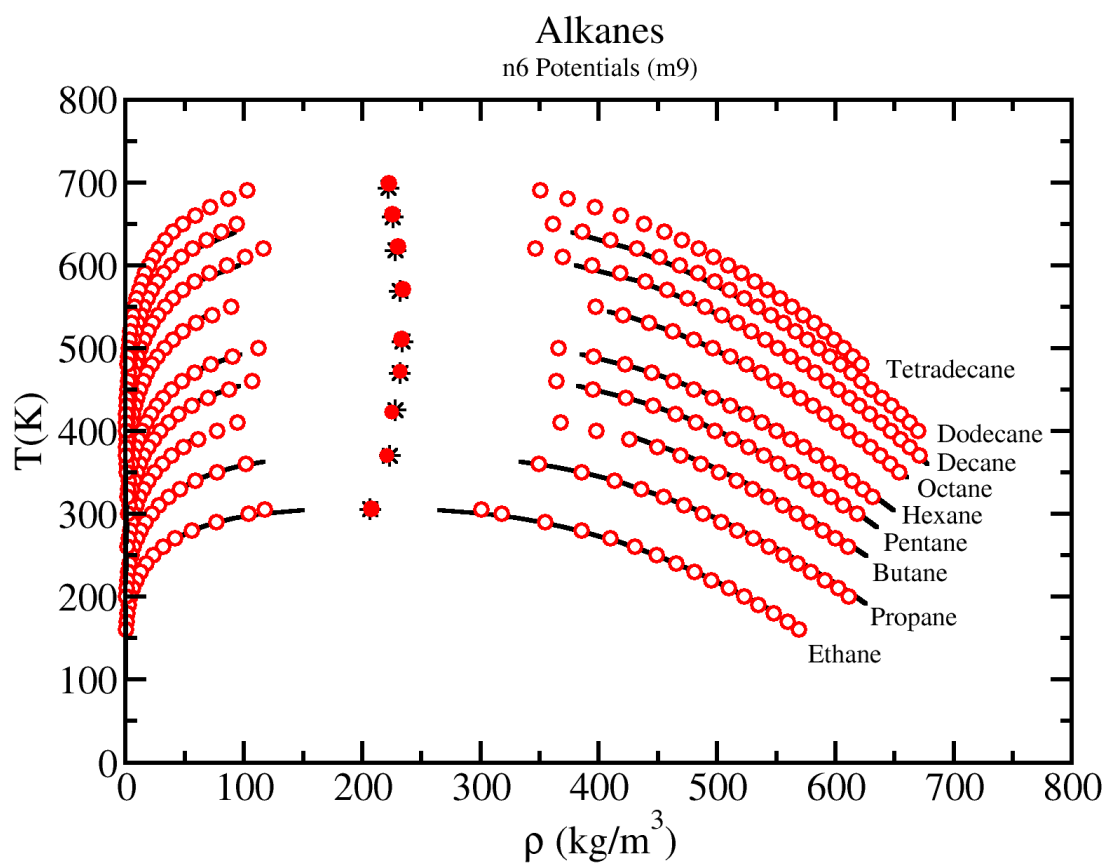
Cygwin

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

https://cygwin.com/
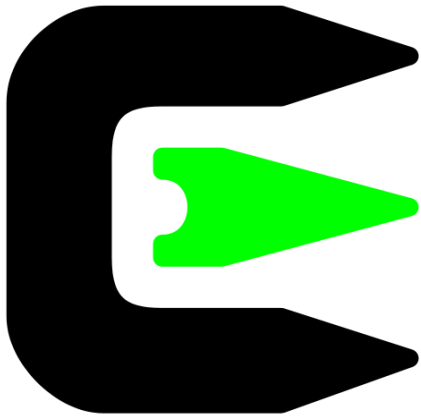
The software is free and open source, licensed under the GNU General Public License version 3. Its primary maintainers are Red Hat Inc. and NetApp. One of the most impressive abilities of Cygwin is its ability to launch a full Windows-compatible X-server Window, which allows convenient visualization 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 which contains 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/

Here's an example of what the downloaded, code looks like when unzipping in Peazip…
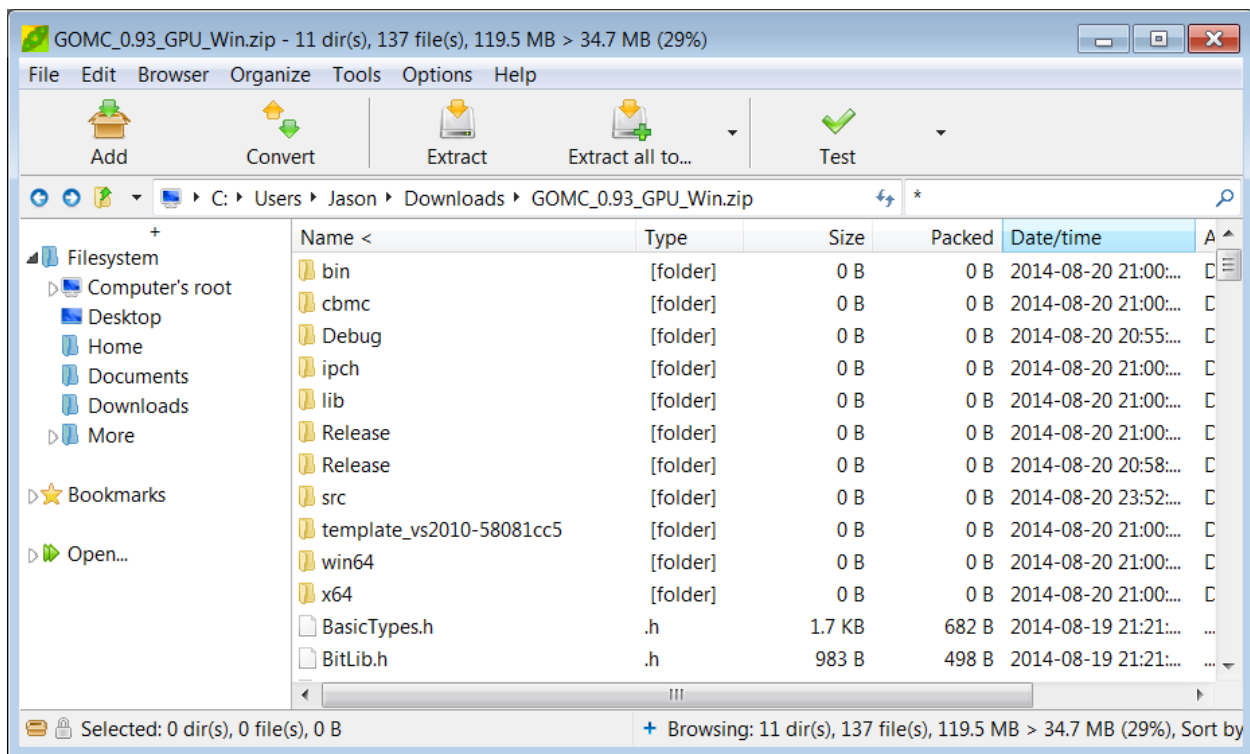
In Linux the GPU and Serial codes are compressed using gzip and tar (*.tar.gz). To extract, simply move to the desire folder and type:

```
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. To do that, open the extracted folder, and then double click on the solution file of the desired Visual Studio version. After the solution is opened in Visual Studio, you can go to the "Build" menu and select "Build solution" to compile the code. You can compile either with release mode or 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, just go into the directory of the project, and then type

```
make
```

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

The default compute capability is set to 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.

```
# CUDA code generation flags
GENCODE_SM10    := -gencode arch=compute_10,code=sm_10
GENCODE_SM20    := -gencode arch=compute_20,code=sm_21
GENCODE_SM30    := -gencode arch=compute_30,code=sm_30
GENCODE_SM35    := -gencode arch=compute_35,code=sm_35
GENCODE_FLAGS   :=  $(GENCODE_SM30)
```

To run the program, run the executable "GOMC.out". To run this, the system's LD_LIBRARY_PATH will need to be configured to support CUDA (more on this later).

Serial Code:

For the serial code – which uses cmake for compilation – simply go to the base directory and type:

```
./metamake.sh
```

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

```
cd bin
```

…and type:

```
make
```

Two executables – GOMC_Serial_GEMC (Gibbs ensemble) and GOMC_Serial_NVT (NVT 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, simply change the value after "CMAKE_BUILD_TYPE:STRING=" from "Release" to "Debug"…

```
# This is the CMakeCache file.
# For build in directory: [REDACTED]/Serial_code/bin
# It was generated by CMake: /usr/bin/cmake
# You can edit this file to change values found and used by cmake.
# If you do not want to change any of the values, simply exit the editor.
# If you do want to change a value, simply edit, save, and exit the editor.
# The syntax for the file is as follows:
# KEY:TYPE=VALUE
# KEY is the name of a variable in the cache.
# TYPE is a hint to GUI's for the type of VALUE, DO NOT EDIT TYPE!.
# VALUE is the current value for the KEY.

########################
# EXTERNAL cache entries
########################

//Path to a program.
CMAKE_AR:FILEPATH=/usr/bin/ar

//Choose the type of build, options are: None Debug Release RelWithDebInfo
// MinSizeRel
CMAKE_BUILD_TYPE:STRING=Release

//Enable/Disable color output during build.
CMAKE_COLOR_MAKEFILE:BOOL=ON

//CXX compiler.
CMAKE_CXX_COMPILER:FILEPATH=/usr/bin/c++
```

To…

```
# This is the CMakeCache file.
# For build in directory: [REDACTED]/Serial_code/bin
# It was generated by CMake: /usr/bin/cmake
# You can edit this file to change values found and used by cmake.
# If you do not want to change any of the values, simply exit the editor.
# If you do want to change a value, simply edit, save, and exit the editor.
# The syntax for the file is as follows:
# KEY:TYPE=VALUE
# KEY is the name of a variable in the cache.
# TYPE is a hint to GUI's for the type of VALUE, DO NOT EDIT TYPE!.
# VALUE is the current value for the KEY.

########################
# EXTERNAL cache entries
########################

//Path to a program.
CMAKE_AR:FILEPATH=/usr/bin/ar

//Choose the type of build, options are: None Debug Release RelWithDebInfo
// MinSizeRel.
CMAKE_BUILD_TYPE:STRING=Debug

//Enable/Disable color output during build.
CMAKE_COLOR_MAKEFILE:BOOL=ON

//CXX compiler.
CMAKE_CXX_COMPILER:FILEPATH=/usr/bin/c++
```

And retype the command:

```
make
```

The outputted 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.

Input File Formats

A typical GOMC GEMC directory has the following:

- GOMC executable
- in.dat (proprietary control file)
- One (NVT ensemble) or two PDB files (Gibbs ensemble)
- One (NVT ensemble) or two PSF files (Gibbs ensemble)
- A CHARMM-style parameter file
- An exotic-style parameter file (may be empty, but must be present)

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 are 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, so for NVT ensemble simulations one file is expected, for Gibbs 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, etc. Note, packmol typically leaves the following remark:

`REMARK original generated coordinate pdb file`

…at the top of the file. Note 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

Crystallographic and Coordinate Transformation Section

This section describes the geometry of the crystallographic experiment and the coordinate system transformations.

CRYST1

Overview

The CRYST1 record presents the unit cell parameters, space group, and Z value. If the structure was not determined by crystallographic means, CRYST1 simply provides the unitary values, with an appropriate REMARK.

Record Format

```
COLUMNS        DATA  TYPE     FIELD         DEFINITION
-------------------------------------------------------------
 1 -  6        Record name    "CRYST1"
 7 - 15        Real(9.3)      a             a (Angstroms).
16 - 24        Real(9.3)      b             b (Angstroms).
25 - 33        Real(9.3)      c             c (Angstroms).
34 - 40        Real(7.2)      alpha         alpha (degrees).
41 - 47        Real(7.2)      beta          beta (degrees).
48 - 54        Real(7.2)      gamma         gamma (degrees).
56 - 66        LString        sGroup        Space  group.
67 - 70        Integer        z             Z value.
```

Details

- If the entry describes a structure determined by a technique other than X-ray crystallography, CRYST1 contains a = b = c = 1.0, alpha = beta = gamma = 90 degrees, space group = P 1, and Z = 1.
- The Hermann-Mauguin space group symbol is given without parenthesis, e.g., P 43 21 2. Please note that the screw axis is described as a two digit number.
- The full International Table's Hermann-Mauguin symbol is used, e.g., P 1 21 1 instead of P 21.
- For a rhombohedral space group in the hexagonal setting, the lattice type symbol used is H.
- The Z value is the number of polymeric chains in a unit cell. In the case of heteropolymers, Z is the number of occurrences of the most populous chain.

As an example, given two chains A and B, each with a different sequence, and the space group P 2 that has two equipoints in the standard unit cell, the following table gives the correct Z value.

Here's 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 the cell is cubic.

NOTE:
Non-cubic cells are not yet 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….

19

Coordinate Section

The Coordinate Section contains the collection of atomic coordinates as well as the MODEL and ENDMDL records.

ATOM

Overview

The ATOM records present the atomic coordinates for standard amino acids and nucleotides. They also present the occupancy and temperature factor for each atom. Non-polymer chemical coordinates use the HETATM record type. The element symbol is always present on each ATOM record; charge is optional.

Changes in ATOM/HETATM records result from the standardization atom and residue nomenclature. This nomenclature is described in the Chemical Component Dictionary (ftp://ftp.wwpdb.org/pub/pdb/data/monomers).

Record Format

```
COLUMNS        DATA  TYPE    FIELD        DEFINITION
-------------------------------------------------------------------------------------
 1 -  6        Record name   "ATOM  "
 7 - 11        Integer       serial       Atom  serial number.
13 - 16        Atom          name         Atom name.
17             Character     altLoc       Alternate location indicator.
18 - 20        Residue name  resName      Residue name.
22             Character     chainID      Chain identifier.
23 - 26        Integer       resSeq       Residue sequence number.
27             AChar         iCode        Code for insertion of residues.
31 - 38        Real(8.3)     x            Orthogonal coordinates for X in Angstroms.
39 - 46        Real(8.3)     y            Orthogonal coordinates for Y in Angstroms.
47 - 54        Real(8.3)     z            Orthogonal coordinates for Z in Angstroms.
55 - 60        Real(6.2)     occupancy    Occupancy.
61 - 66        Real(6.2)     tempFactor   Temperature  factor.
77 - 78        LString(2)    element      Element symbol, right-justified.
79 - 80        LString(2)    charge       Charge  on the atom.
```

Details

- ATOM records for proteins are listed from amino to carboxyl terminus.

- Nucleic acid residues are listed from the 5' to the 3' terminus.

- Alignment of one-letter atom name such as C starts at column 14, while two-letter atom name such as FE starts at column 13.

- Atom nomenclature begins with atom type.

- No ordering is specified for polysaccharides.

- Non-blank alphanumerical character is used for chain identifier.

- The list of ATOM records in a chain is terminated by a TER record.

- If more than one model is present in the entry, each model is delimited by MODEL and ENDMDL records.

- AltLoc is the place holder to indicate alternate conformation. The alternate conformation can be in the entire polymer chain, or several residues or partial residue (several atoms within one residue). If an atom is provided in more than one position, then a non-blank alternate location indicator must be used for each of the atomic positions. Within a residue, all atoms that are associated with each other in a given conformation are assigned the same alternate position indicator. There are two ways of representing alternate conformation- either at atom level or at residue level (see examples).

- For atoms that are in alternate sites indicated by the alternate site indicator, sorting of atoms in the ATOM/HETATM list uses the following general rules:

```
      - In the simple case that involves a few  atoms or a few residues with alternate sites,
        the coordinates occur one after  the other in the entry.
      - In the case of a large heterogen groups  which are disordered, the atoms for each conformer
        are listed together.
```

- Alphabet letters are commonly used for insertion code. The insertion code is used when two residues have the same numbering. The combination of residue numbering and insertion code defines the unique residue.

- If the depositor provides the data, then the isotropic B value is given for the temperature factor.

- If there are neither isotropic B values from the depositor, nor anisotropic temperature factors in ANISOU, then the default value of 0.0 is used for the temperature factor.

- Columns 79 - 80 indicate any charge on the atom, e.g., 2+, 1-. In most cases, these are blank.

- For refinements with program REFMAC prior 5.5.0042 which use TLS refinement, the values of B may include only the TLS contribution to the isotropic temperature factor rather than the full isotropic value.

20

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

Other important entries are the residue name, atom name, and chainID. Numbering is important primarily in that it represents an inconvenience in packing/loading large systems. Looking at the previous example:

```
REMARK    1_____2_____3_____4_____5_____6_____7_____8
REMARK  89012345678901234567890123456789012345678901234567890123456789012345678 90
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 Undocument PDB Format Conventions:

- While it is only 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 21 unused character to add a fourth letter to the residue (molecule name). This extension is currently support, 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 outputted PDBs from 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 the box a particle is in, in our outputted PDBs (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.

_____

Past either of these and an "overflow" solution is hit, where it will print stars to the respective field. This file should still be readable, but is consider unsafe in terms of portability, as it will work for GOMC and NAMD, but may fail for other codes.

_____

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

With that overview of the format in mind, here's how a PDB file is typically built.

First, 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
```

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:

1. Pack both boxes to a single "middle" density, which is an average of the liquid and vapor densities.
2. 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.
3. Pack one box to the predicted liquid density and the other to the vapor density.

T_c (K)	407.7	P_c (bar)	36.5	ρ_c (kg/m³)	223.77	
T_b (K)	262	MW (g/mol)	58.1222	Avog.	6.022E+23	Conv. kg/A³
T_m (K)	140	cm3--> A3	1E+24	N (molec.)	500	2.072E-08
FROM NIST	(click to browse)					
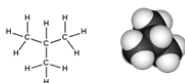
SATURATED VLE

BOTH			VAPOR											LIQUID										
T(K)	P (bar)	γ (N/m)	ρ (kg/m³)	v (m³/kg)	U (kJ/mol)	H (kJ/mol)	S (J/mol·K)	C_v (J/mol·K)	C_P (J/mol·K)	C_sound (m/s)	μ_JT (K/bar)	η (μPa·s)	λ (W/m·K)	ρ (kg/m³)	v (m³/kg)	U (kJ/mol)	H (kJ/mol)	S (J/mol·K)	C_v (J/mol·K)	C_P (J/mol)	C_sound (m)	μ_JT (K/ba)	η (μPa·s)	λ (W/m·K)
230	0.24387	0.0183	0.75229	1.3293	27.019	28.903	135.31	71.137	80.055	189.56	4.9534	5.8209	0.0104	627.39	0.0016	6.1547	6.157	36.415	83.944	120.96	1209.5	-0.048	342.120	0.1163
240	0.40218	0.017058	1.1969	0.8355	27.705	29.658	134.44	73.796	82.988	192.25	4.3661	6.0634	0.0113	616.95	0.0016	7.3758	7.3795	41.612	85.714	123.4	1151.2	-0.045	298.320	0.1121
250	0.6335	0.015827	1.8253	0.54785	28.406	30.423	133.88	76.548	86.099	194.52	3.8843	6.3045	0.0122	606.28	0.0016	8.6216	8.6277	46.697	87.581	126	1093.4	-0.042	262.040	0.1079
260	0.95885	0.014608	2.6848	0.37247	29.12	31.196	133.58	79.396	89.409	196.32	3.4866	6.5453	0.0131	595.37	0.0017	9.8936	9.903	51.687	89.541	128.76	1036.1	-0.039	231.630	0.1039
270	1.4017	0.013404	3.8285	0.2612	29.846	31.974	133.5	82.338	92.94	197.59	3.1567	6.7873	0.014	584.17	0.0017	11.193	11.207	56.592	91.532	131.7	979.29	-0.036	205.870	0.0999
280	1.9876	0.012215	5.3164	0.1881	30.581	32.754	133.61	85.375	96.722	198.29	2.8823	7.0328	0.015	572.64	0.0017	12.522	12.543	61.427	93.728	134.86	922.77	-0.032	183.830	0.096
290	2.744	0.011044	7.217	0.13856	31.326	33.535	133.87	88.503	100.79	198.35	2.6539	7.2844	0.016	560.71	0.0018	13.883	13.911	66.201	95.946	138.26	866.47	-0.027	164.800	0.0922
300	3.7	0.0098917	9.6096	0.10406	32.076	34.314	134.26	91.721	105.2	197.74	2.4641	7.5457	0.017	548.32	0.0018	15.276	15.315	70.927	98.243	141.94	810.25	-0.022	148.220	0.0886
310	4.8858	0.0087616	12.589	0.07944	32.83	35.086	134.74	95.03	110.03	196.37	2.3074	7.8217	0.0182	535.39	0.0019	16.705	16.758	75.615	100.62	145.38	753.97	-0.016	133.640	0.0851
320	6.3335	0.0076559	16.269	0.06147	33.584	35.847	135.29	98.434	115.39	194.17	2.1794	8.1184	0.0194	521.81	0.0019	18.171	18.242	80.276	103.06	150.46	697.45	-0.009	120.670	0.0818
330	8.0761	0.0065779	20.795	0.04809	34.335	36.592	135.89	101.91	121.44	191.05	2.0778	8.4448	0.0207	507.43	0.002	19.68	19.772	84.925	105.59	155.52	640.46	0.0007	109.010	0.0786
340	10.148	0.0055315	26.361	0.03794	35.075	37.313	136.51	105.41	128.42	186.9	2.0032	8.8129	0.0222	492.07	0.002	21.234	21.354	89.574	108.21	161.39	582.74	0.0124	98.394	0.0756
350	12.587	0.0045214	33.23	0.03009	35.797	37.998	137.11	108.95	136.89	181.59	1.9577	9.2411	0.0239	475.47	0.0021	22.84	22.994	94.242	110.93	168.44	523.94	0.0279	88.587	0.0727
360	15.433	0.0035537	41.791	0.02393	36.486	38.632	137.65	112.72	148.11	174.96	1.9413	9.7575	0.026	457.23	0.0022	24.506	24.702	98.954	113.8	177.37	463.6	0.0494	79.371	0.0699
370	18.727	0.0026364	52.652	0.01899	37.123	39.19	138.06	117.03	164.55	166.77	1.9526	10.409	0.0285	436.72	0.0023	26.245	26.494	103.75	116.86	189.58	401.01	0.0815	70.526	0.0674
380	22.513	0.0017812	66.867	0.01496	37.677	39.635	138.25	122.11	191.67	156.73	1.9914	11.278	0.0319	412.85	0.0024	28.079	28.396	108.68	120.26	208.53	335.05	0.1341	61.731	0.0649
390	26.869	0.0010065	86.625	0.01154	38.083	39.886	138.04	128.27	246.92	144.37	2.0583	12.546	0.0368	383.26	0.0026	30.057	30.464	113.88	124.42	245.34	263.93	0.2344	52.770	0.0628
400	31.856	0.00034895	118.39	0.00845	38.148	39.712	136.88	136.83	439.11	128.9	2.1293	14.761	0.0465	341.03	0.0029	32.319	32.862	119.75	130.78	369.01	184.38	0.4926	42.498	0.0619

TraPPE 2 Results

T(K)	P(bar)	ρ_vap (kg/m³)	ρ_liq (kg/m³)	Z
232	0.45	1.38	626	0.98
271	1.76	4.8	581	0.949
304	5.40	14.1	544	0.88
330	10.39	27.3	510	0.81
362	20.00	55.9	454	0.69

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/

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. 0. 0. 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:

```
./packmol < pack_isobutane.inp
```

A more efficient way than packing individual simulations by hand is to embed a packmol script in a larger script, which reads in a table of comma or space separated values for the packing box sizes, as calculated via a spreadsheet.

Here's an example snippet from tcl:

```
#(20) Pack copies of base model in first box.
exec packmol << "
    tolerance 3.0
    filetype pdb
    output STEP2_${compound_shorthand}_packed_BOX_0.pdb

    structure STEP1_${compound_shorthand}.pdb
    number ${num_per_box}
    inside box 1. 1. 1. ${pack_l}. ${pack_l}. ${pack_l}.
    end structure"

#(21) Pack copies of base model in second box.
exec packmol << "
    tolerance 3.0
    filetype pdb
    output STEP2_${compound_shorthand}_packed_BOX_1.pdb

    structure STEP1_${compound_shorthand}.pdb
    number ${num_per_box}
    inside box 1. 1. 1. ${pack_v}. ${pack_v}. ${pack_v}.
    end structure"
```

When packing large files, packmol will adjust for this by incrementing the chain letter. This will cause issues with PSFGen. The solution is to use "grep" in Linux (or a similar method in Windows) to separate the PDB into several separate PDB files, one per chain. Each collection of residues of a specific kind will typically end up in a file together using this methodology.

The PDB output packmol is fed into PSFGen, along with a topology file. PSFGen will output a final PDB, which will be nearly identical to the packmol generated file. The only observable difference is generally the addition of a "segname", the aforementioned field VMD/NAMD insert into an unspecified set of columns in ATOM entries. The two PDB files generated by PSFGen (or one if an NVT ensemble simulation is being prepped) will serve as the coordinates files for a new simulation.

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 laid out in a series of sections. At the top of each section there is typically a line with a numeric value. 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 in the bonds, angles, dihedrals and impropers.

A few important Notes/Warnings:

- The PSF file format is a highly redundant file format. For example 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) are unsupported or legacy, hence 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 to do this. For example the script:

```
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 consider one of the integral file types. It will be briefly discussed in the following section.

As with the packmol script, its often useful to pack the PSFGen script in a tcl or python script. Here's an example:

```
        #(20) Pack copies of base model in first box.
        exec packmol << "
            tolerance 3.0
            filetype pdb
            output STEP2_${compound_shorthand}_packed_BOX_0.pdb

            structure STEP1_${compound_shorthand}.pdb
            number ${num_per_box}
            inside box 1. 1. 1. ${pack_l}. ${pack_l}. ${pack_l}.
            end structure"

        #(21) Pack copies of base model in second box.
        exec packmol << "
            tolerance 3.0
            filetype pdb
            output STEP2_${compound_shorthand}_packed_BOX_1.pdb

            structure STEP1_${compound_shorthand}.pdb
            number ${num_per_box}
            inside box 1. 1. 1. ${pack_v}. ${pack_v}. ${pack_v}.
            end structure"

        #(22) Run tcl script to build the final PDB/PSF for box 0
        exec psfgen << "
            topology ./Top_Branched_Alkanes.inp

            segment ${compound_shorthand} {
                pdb ./STEP2_${compound_shorthand}_packed_BOX_0.pdb
                first none
                last none
            }

            coordpdb ./STEP2_${compound_shorthand}_packed_BOX_0.pdb ${c\
ompound_shorthand}

            writepsf ./STEP3_START_${compound_shorthand}_sys_BOX_0.psf
            writepdb ./STEP3_START_${compound_shorthand}_sys_BOX_0.pdb
            "

        #(23) Run tcl script to build the final PDB/PSF for box 1
        exec psfgen << "
            topology ./Top_Branched_Alkanes.inp

            segment ${compound_shorthand} {
                pdb ./STEP2_${compound_shorthand}_packed_BOX_1.pdb
                first none
                last none
            }

            coordpdb ./STEP2_${compound_shorthand}_packed_BOX_1.pdb ${c\
ompound_shorthand}

            writepsf ./STEP3_START_${compound_shorthand}_sys_BOX_1.psf
            writepdb ./STEP3_START_${compound_shorthand}_sys_BOX_1.pdb
            "
```

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
       9 ISB  3    ISB  C1    CH1    0.000000        13.0190          0
      10 ISB  3    ISB  C2    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       9      10       9      11
       9      12      13      14      13      15      13      16
      17      18      17      19      17      20      21      22
      21      23      21      24      25      26      25      27
      25      28      29      30      29      31      29      32
      33      34      33      35      33      36      37      38
   ....
    3977    3980    3981    3982    3981    3983    3981    3984
    3985    3986    3985    3987    3985    3988    3989    3990
    3989    3991    3989    3992    3993    3994    3993    3995
    3993    3996    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
      10       9      12      10       9      11      11       9      12
```

27

```
    14        13        16        14        13        15        15        13        16
    18        17        20        18        17        19        19        17        20
    22        21        24        22        21        23        23        21        24
    26        25        28        26        25        27        27        25        28
  ....
  3990      3989      3992      3990      3989      3991      3991      3989      3992
  3994      3993      3996      3994      3993      3995      3995      3993      3996
  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 (with their masses), plus a list of residue information (charges, composition, and topology). It is essentially a non-redundant lookup table equivalent to the PSF file.

This is followed by a series of residues, which tell PSFGen what's bonded to what. 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 containing pairs of bonded atoms (typically up to 3 per line)

28

- 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    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
```

Note the keyword END must be used to terminate this file and keywords related to the autogeneration 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/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 requires two forcefield files:

- CHARMM (Chemistry at HARvard Molecular Mechanics) compatible parameter file
- "exotic" parameter file

If one of these files is missing an error will be thrown.

Both forcefields are whitespace separated files, with sections preceded by a tag. When a known tag (representing a kind of molecular interaction in the model) is encountered, reading of that section of the forcefield 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 contains a widely used model used in Monte Carlo and molecular dynamics simulations for describing energies. The exotic file contains certain models of molecular interaction which are not included in CHARMM forcefields. The currently supported interactions include:

- `DIHEDRALS_POWER_SERIES` : Cosine power series-style dihedral potentials (used for TraPPE fluoralkanes)
- `DIHEDRALS_QUAD_CIS_TRANS` configurationally dependent quadratic torsional potential (used for TraPPE alkenes)
- `NONBONDED_MIE` n-6 potential. The Lennard-Jones potential (12-6) is a subset of this potential. Manipulating the repulsive exponent gives you another means of tuning your forcefield.

For a general overview of the CHARMM forcefield, see:

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

Here's the basic CHARMM contributions:

$U_{CHARMM} = U_{bonded} + U_{non-bonded}$

where U_{bonded} consists of the following terms,

$U_{bonded} = U_{bond} + U_{angle} + U_{UB} + U_{dihedral} + U_{improper} + U_{CMAP}$

with

$$U_{bond} = \sum_{bonds} K_b(b - b^0)^2,$$

$$U_{angle} = \sum_{angles} K_\theta(\theta - \theta^0)^2,$$

$$U_{UB} = \sum_{Urey-Bradley} K_{UB}(b^{1-3} - b^{1-3,0})^2$$

$$U_{dihedral} = \sum_{dihedrals} K_\varphi((1 + \cos(n\varphi - \delta)))$$

$$U_{improper} = \sum_{impropers} K_\omega(\omega - \omega^0)^2, \text{ and}$$

$$U_{CMAP} = \sum_{residues} u_{CMAP}(\Phi, \Psi)$$

$U_{non-bonded}$ consists of two terms,

$U_{non-bonded} = U_{LJ} + U_{elec}$

with

$$U_{LJ} = \sum_{nonb.pairs} \varepsilon_{ij} \left[\left(\frac{r_{ij}^{min}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{ij}^{min}}{r_{ij}} \right)^6 \right],$$

$$U_{elec} = \sum_{nonb.pairs} \frac{q_i q_j}{\epsilon r_{ij}}.$$

✔ **Fully supported**

◯ **Read, but not supported**

✖ **Not currently Supported or read**

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

- BONDS
  - Quadratic expression describing bond stretching based on bond length (*b*) in Angstrom
  - Typically is ignored as bonds are rigid for Monte Carlo sims.
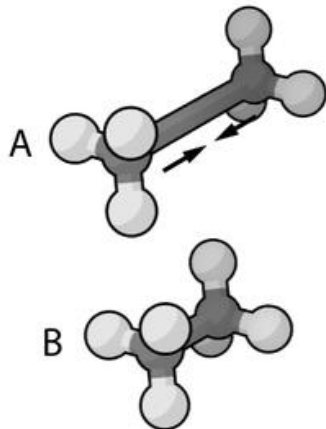
[Image Courtesy of Wikimedia Commons]

- ANGLES

   o Describes the conformational behavior of an angle (ϑ) between three atoms , one of which is shared branch point to the other two.
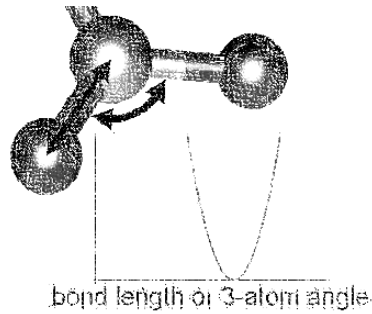
Image Courtesy of Wikimedia Commons

- DIHEDRALS Describes crankshaft like rotation behavior about a central bond in a series of three consecutive bonds (rotation is given as φ)
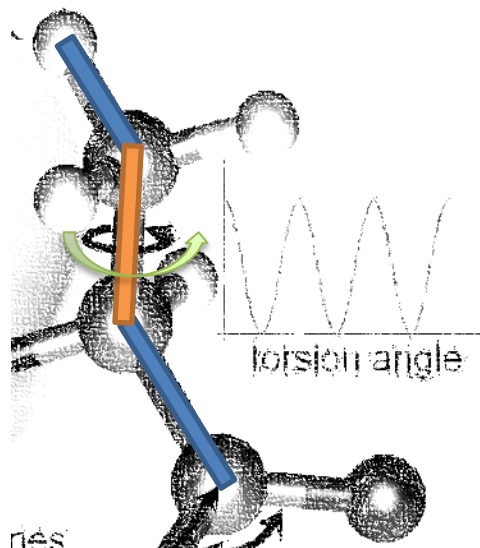
- NONBONDED This section should only be used if the NONBONDED_MIE section is not being used in the exotic parameter file. This section describes 12-6 (Lennard-Jones) nonbonded interactions.
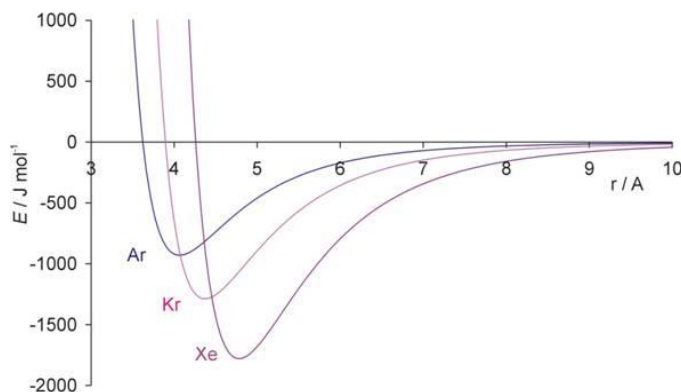
Image Courtesy of the Univ. of Bristol

"Exotic" parameter file

The most frequently used section of the exotic files in the Mie potential section, 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_ij     n    description
CH4      161.00      3.740      14   ! Potoff, et al. '09
CH3      121.25      3.783      16   ! Potoff, et al. '09
CH2       61.00      3.990      16   ! Potoff, et al. '09
```

Note the exotic file uses σ, *not* the R_{min} used by CHARMM, although the units (Angstroms) are the same. The energy in the exotic file are expressed in Kelvin (K), as this is the standard convention in the literature.

CHARMM parameter file

Most parts of the model are listed in the CHARMM compliant parameter file. Angles and bonds (if applicable) are always found in the CHARMM file. Currently supported sections of the CHARMM

compliant file include BONDS, ANGLES, DIHEDRALS, IMPROPERS, NONBONDED. Other sections such as CMAP are not currently read or supported.

BONDS (aka "bond stretching) are one key sections of the CHARMM-compliant file. Units for the K_b variable in this section are in kcal/mol (for $K_{b)}$; the b_0 section (which represents the default 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.; (9999999999 if no stretching)
!
!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 forcefields (e.g. TraPPE) typically treat molecules as rigid constructs, Kb is set to a large value -- 9999999999

Next comes an ANGLES (bond bending) section. θ and θ_0 are commonly measured in degrees and K_θ is measured in kcal/mol/K. These values are often given in the literature in Kelvin (K). To convert Kelvin to kcal/mol/K, multiply by the Boltzmann constant -- k_b, 0.0019872041 kcal/mol

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
```

```
!Kub: kcal/mole/A**2 (Urey-Bradley)
!SO: A
!
! Ktheta (kcal/mol) = Ktheta (K) * Boltz. const.
!
!atom types        Ktheta      Theta0    Kub(?)  SO(?)
CH3 CH1 CH3        62.100125   112.00 ! TraPPE 2
```

Some CHARMM ANGLES section entries include Urey-Bradley potentials (K_{ub}, b_{ub}), in addition to the standard quadratic angle potential. The constants related to this potential function are currently read in, 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's 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 as this kind are often determined solely by the two middle atoms in the middle of the dihedral, according to the literature

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

The final section of the CHARMM style parameter file is the NONBONDED. In order to use TraPPE this section of the CHARMM compliant file will be critical. Here's an example with our isobutane potential models.

```
NONBONDED nbxmod  5 atom cdiel fshift vatom vdistance vfswitch -
!cutnb 14.0 ctofnb 12.0 ctonnb 10.0 eps 1.0 e14fac 1.0 wmin 1.5
!
!V(Lennard-Jones) = Eps,i,j[(Rmin,i,j/ri,j)**12 - 2(Rmin,i,j/ri,j)**6]
!
!epsilon: kcal/mole, Eps,i,j = sqrt(eps,i * eps,j)
!Rmin/2: A, Rmin,i,j = Rmin/2,i + Rmin/2,j
!
! Rmin = sig * (2^(1/6)) / 2 ; eps (kcal/mol) = eps (K) * Boltz. const.
!                              Boltzmann = 0.0019872041 kcal / (mol * K)
!
!atom   ignored   epsilon      Rmin/2        ignored  eps,1-4  Rmin/2,1-4
CH3     0.0       -0.194745992 2.10461634058 ! TraPPE 1
CH1     0.0       -0.019872040 2.62656119304 ! TraPPE 2
Control File
End
```

Here's the finished Exotic file for isobutane (basically, empty, as we have no dihedrals, but with comments left in, to prepare for future reuse):

```
* Parameter file for GO-MC
*
*


DIHEDRALS_POWER_SERIES
!
!V(dihedral) = Kchi * cos(chi - delta)^n
!
!Kchi: kcal/mole
!n: power
!delta: shift
!
!atom types        Kchi    n    delta            description


DIHEDRALS_QUAD_CIS_TRANS
!
!V(dihedral) = Kchi(1 + cos(n(chi) - delta))
!
! Kchi: kcal/mole
!n: 0 - cis; 1 - trans
!delta: shift
!
!atom types        Kchi     n    delta            description


NONBONDED_MIE
!
!V(mie) = 4*eps*((sig_ij/r_ij)^n-(sig_ij/r_ij)^6)
!
!atom    eps        sig_ij    n    description
```

And here's the finished CHARMM format Parameter file for CHARMM isobutane.

```
* Parameter file for GO-MC
* Contains parameters for isobutane (TraPPE 2 FF)
*
```

```
BONDS
!
!V(bond) = Kb(b - b0)**2
!
!Kb: kcal/mole/A**2
!b0: A
!
! Kb (kcal/mol) = Kb (K) * Boltz. const.; (9999999999 if no stretching)
!
!atom type        Kb              b0          description
CH3 CH1          9999999999      1.540    ! TraPPE 2
ANGLES
!
!V(angle) = Ktheta(Theta - Theta0)**2
!
!V(Urey-Bradley) = Kub(S - S0)**2
!
!Ktheta: kcal/mole/rad**2
!Theta0: degrees
!Kub: kcal/mole/A**2 (Urey-Bradley)
!S0: A
!
! Ktheta (kcal/mol) = Ktheta (K) * Boltz. const.
!
!atom types        Ktheta        Theta0    Kub(?)   S0(?)
CH3 CH1 CH3        62.100125     112.00 ! TraPPE 2

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
```

```
NONBONDED nbxmod  5 atom cdiel fshift vatom vdistance vfswitch -
!cutnb 14.0 ctofnb 12.0 ctonnb 10.0 eps 1.0 e14fac 1.0 wmin 1.5
!
!V(Lennard-Jones) = Eps,i,j[(Rmin,i,j/ri,j)**12 - 2(Rmin,i,j/ri,j)**6]
!
!epsilon: kcal/mole, Eps,i,j = sqrt(eps,i * eps,j)
!Rmin/2: A,  Rmin,i,j = Rmin/2,i + Rmin/2,j
!
! Rmin = sig * (2^(1/6)) / 2 ; eps (kcal/mol) = eps (K) * Boltz. const.
!                             Boltzmann = 0.0019872041 kcal / (mol * K)
!
!atom  ignored   epsilon      Rmin/2       ignored  eps,1-4  Rmin/2,1-4
CH3    0.0      -0.194745992  2.10461634058 ! TraPPE 1
CH1    0.0      -0.019872040  2.62656119304 ! TraPPE 2
```

Note, the dihedrals section contains no values for this file, but was inserted none the less to demonstrate an empty section, similar to the unused sections of the exotic file. Alternatively empty tagged sections can simply be left off (as is done for IMPROPERS in this case).

Control File (in.dat):

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

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.

- InState: Determines whether to restart, and if so what step to restart from.
  - Value 1: <BOOLEAN> – true if restart, false otherwise
  - Value 2: <ULONG> – step to restart from

    Example:

```
##################################
# enable, step
##################################
InState      true    1000000
```

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

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

    - RESTART: Used for restarting a previous simulation, this option loads in the Mersenne Twister's state from a saved file.

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

    - 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.

      Example:
      ```
      ##################################
      # kind {RESTART, RANDOM, INTSEED}
      ##################################
      PRNG       INTSEED        5
      ```

  - o Value 2: ULONG or UINT: If "INTSEED" command is used (see above example).

- **InParam:** Provides the filenames of the CHARMM-compatible and "exotic" style parameter files (the forcefield) to read in.

  - o Value 1: <STRING> – Defines parameter file in CHARMM format.
  - o Value 2: <STRING> – Defines parameter file in EXOTIC format.

  Example:

```
#####################################
# charmm param, exotic param
#####################################
InParam     Par_CHARMM.inp  Par_Exotic.inp
```

- **InPDB:** Gives the filenames of the PDB files (coordinates) for each box in the system (one entry for NVT ensemble, two for GEMC)
  - Value 1: <STRING> – Defines pdb file name of first box.
  - Value 2 (*GEMC only*): <STRING> – Defines pdb file name of second box.

  Example:
```
#####################################
# pdbs for 1st and 2nd box
#####################################
InPDB STEP3_START_ISB_sys_BOX_0.pdb STEP3_START_ISB_sys_BOX_1.pdb
```

- **InPSF:** Gives the filenames of the PSF (topology) files for each box in the system (one entry for NVT ensemble, two for GEMC)
  - Value 1: <STRING> – Defines psf file name of first box.
  - Value 2 (*GEMC only*): <STRING> – Defines psf file name of second box.

  Example:
```
#####################################
# psfs for 1st and 2nd box
#####################################
InPSF STEP3_START_ISB_sys_BOX_0.psf STEP3_START_ISB_sys_BOX_1.psf
```

System Settings for During Run Setup

This section contains all the variables not involved with the output of data during the simulation or with 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 Gibb 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 w/ constant mole number, volume and temperature.

    Example :
    ```
    ###############################
    #GEMC     type      Pressure (bar)
    ###############################
    GEMC      NVT
    ```

  - NPT: Run simulation w/ constant mole number, pressure and temperature.

    Example:
    ```
    ###############################
    #GEMC     type      Pressure (bar)
    ###############################
    GEMC      NPT       15.5
    ```

- Value 2: \<DOUBLE\> -- Constant pressure in bars to run simulation at, if NPT ensemble is selected (see above example).

- **System**: This section defines the parameters that are required to calculate energy.
  - Value 1: \<BOOLEAN\> – whether to use long range corrections (values: "true" or "false")
  - Value 2: \<DOUBLE\> – constant temperature of simulation in Kelvin
  - Value 3: \<DOUBLE\> – the distance to truncate the Lennard-Jones potential at
  - Value 4: \<DOUBLE\> – scaling for all 1-4 nonbonded interactions

    Example:
    ```
    ###############################################################
    # SYSTEM VARS    #LRC     #Temp (K)   #cutoff (A)    #1-4 Inter.
    ###############################################################
    System           true     270.00      10             0.0
    ```

    Note/Warning:
    This isn't fully CHARM compliant. In CHARMM the 1-4 interaction scaling is supposed to be handled by having a separate 1-4 parameter in the nonbonded section. In the future the code's support for the NONBONDED_MIE and NONBONDED sections of the forcefield will be updated to incorporate these parameters. The constant will then be transititioned to an electrostatics 1-4 scaling terms, along the lines of what NAMD currently does.

- **Steps**: Intervals for key step events
    - **Value 1: <ULONG>** – total number of steps to run for (one move is performed for each step) (cycles = this value / number of molecules in the system)
    - **Value 2: <ULONG>** – estimate of the number of steps necessary to equilibrate the system ; averaging will begin at this step.
    - **Value 3: <ULONG>** – 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 A^3 to swap, etc.)
    Example:

```
###############################################################
# STEPS PER...   #total        #till Equil   #Per adjustment
###############################################################
Steps           1000000       10000000      1000
```

- **Percent:** Fractional percentage of each move (should add up to 1.0)
    - Value 1: <DOUBLE> – % displace
    - Value 2: <DOUBLE> – % rotate
    - Value 3: <DOUBLE> – (FOR Gibb ensemble runs only) % of volume swaps
    - Value 4: <DOUBLE> – (FOR Gibb ensemble runs only) % of molecule swaps

    Example:

```
#################################################
# MOVE %               #disp. #rot.  #vol.  #mol.
#################################################
Percent               0.790  0.10   0.01   0.10
```

- **Box**: Defines the axis lengths of simulation box. This tag may occur multiple times (It occurs once for NVT, but twice for Gibbs 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

    - Example (NVT ensemble):

```
################################
# box #, x, y, z
################################
BoxDim  0    40.00 40.00 40.00
```

    Example (Gibbs ensemble):

```
####################################
# box #, x, y, z
####################################
BoxDim  0    106.60 106.60 106.60
BoxDim  1    176.60 176.60 176.60
```

- **Growth**: (FOR Gibb ensemble runs only, currently) Number of CBMC trials to perform during certain stages of the regrowth
  - Value 1: <INTEGER> – Number of initial insertion sites to try
  - Value 2: <INTEGER> – Number of trials for subsequent regrown positions in topology

  Example:
  ```
  ##############################
  # growth tr.  first   nth
  ##############################
  Growth          10     4
  ```

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.

- **OutNameUnique**: Unique name for simulation used to name the block output files.
  - **Value 1**: <STRING> – unique phrase to identify this system

  Example:
  ```
  ##########################################
  # statistics filename add
  ##########################################
  OutNameUnique   T_270.00_K
  ```

- **OutState**: controls output of PDB file (coordinates) and PRNG state.
  - Value 1: <BOOLEAN> –"true" enables dumping these files; "false" disables dumping.
  - Value 2: <ULONG> – steps per dump event

  Example:
  ```
  #######################
  # enable, frequency
  ```

```
#########################
OutState    true    1000000
```

- **OutPDB**: Name of the output PDB, if PDB dumping is enabled (one files for NVT, two files for Gibbs ensemble).
    - Value 1: <STRING> – name of PDB file to output to for first box.
    - Value 2: <STRING> – (FOR Gibb ensemble runs only) name of output pdb file for second box

    Example:
    ```
    #########################
    # Box 0, Box 1 PDBs
    #########################
    OutPDB      C1T_T_270.00_K_0.pdb C1T_T_270.00_K_1.pdb
    ```

    Note/Warning:
    The PDB file contains an entry for every **ATOM**, in all boxes read in. 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).

- **OutPSF, OutSeed**: Names of PSF and seed (PRNG state/frame) files (respectively).
    - Value 1: <STRING> – File name

    Example:
    ```
    ##############################################
    # (merged PSF, seed)
    ##############################################
    OutPSF      C1T_T_270.00_K_Run_1_MIE_merged.psf
    OutSeed     C1T_T_270.00_K_Run_1_MIE_seedOut.dat
    ```

    Note/Warning:
    The PSF file contains an entry for every **ATOM**, in all boxes read in. It also contains all the topology for every molecule in both boxes, corresponding to the merged PDB format.

- **OutConsole**: Controls the output to STDIO ("the console") of messages such as acceptance statistics, averages, and run timing info.
    - Value 1: <BOOLEAN> – "true" enable message printing; "false" disables dumping.
    - Value 2: <ULONG> – number of steps per print

Example:
```
############################
# enable, frequency
############################
OutConsole   true   10000
```

- **OutBlockAverage, OutFluctuation, OutHistogram**: Controls the output of block averages, fluctuation logs, and histograms (respectively). 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); fluctuations are a log of instantaneous values for those quantities; histograms are a binned listing of observation frequency for a specific thermodynamic variable
  - Value 1: <BOOLEAN> – "true" enable message printing; "false" disables it.
  - Value 2: <ULONG> – number of steps per file output

  Example:
  ```
  ##################################
  # enable, frequency
  ##################################
  OutBlockAverage   true   100000
  OutFluctuation    true   10000
  OutHistogram      true   100000
  ```

  Note/Warning:
  Fluctuation logs and histograms are not currently supported. However, they are included here for future support. As a result, these tags can be ignored at present, but will be necessary for future functionality.

- **OutEnergy, OutPressure, OutMolNumber*, OutDensity*, OutVolume***: Enables/disables for specific kinds of file output for tracked thermodynamic quantities

  (*) = Gibbs ensemble only (as these quantities don't fluctuate in NVT)

  - Value 1: <BOOLEAN> – "true" enable message output of block averages of this tracked parameter (and in some cases such as entry, components); "false" disables it.
  - Value 2: <BOOLEAN> – "true" enable message output of a fluctuation log for this tracked parameter (and in some cases such as entry, components); "false" disables it.
  - Value 3: <BOOLEAN> – "true" enable message output of a histogram of observation frequency for values of this tracked parameter (and in some cases such as entry, components); "false" disables it.

Example:

```
###################################
# enable: blk avg., fluct., hist.
###################################
OutEnergy          true    true    true
OutPressure        true    true    true
OutMolNum          true    true    true
OutDensity         true    true    true
OutVolume          true    true    true
```

Note/Warning:

Fluctuation logs and histograms are not currently supported. However, they are included here for future support. As a result, these tags can be ignored at present, but will be necessary for future functionality. Unlike the previous enable section, these values do require three entries per line.

Also, more tracked parameters, such as move acceptance, move constant values, chemical potential, sampled geometry, etc. may be added in the future.

Sample Files

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

```
#===============================================
#          GOMC beta 0.94 — NVT Control example    =
#===============================================
#

################################################################
###    INPUT
###################################
#
################################
# enable, step
################################
InState      true    1000000
##################################
# kind {RESTART, RANDOM, INTSEED}
##################################
PRNG        RANDOM
##################################
# charmm param, exotic param
```

```
######################################
InParam     Par_CHARMM_LJ_BENCH_FF.inp  Par_Exotic_LJ_BENCH_FF.inp
######################################
# pdbs for 1st and 2nd box
######################################
InPDB STEP3_START_LJ_sys_BOX_0.pdb STEP3_START_LJ_sys_BOX_1.pdb
######################################
# psfs for 1st and 2nd box
######################################
InPSF STEP3_START_ISB_sys_BOX_0.psf STEP3_START_ISB_sys_BOX_1.psf


################################################################
###    SYSTEM
######################################
#
################################################################
# SYSTEM VARS    #LRC      #Temp (K)  #cutoff (A)   #1-4 Inter.
################################################################
System          true     0.85        10            0.0
################################################################
# STEPS PER...  #total        #tiil Equil   #Per adjustment
################################################################
Steps          1000000       10000000      1000
######################################
# MOVE %              #disp. #rot.
######################################
Percent              1.00    0.00
######################################
# box #, x, y, z
######################################
BoxDim  0    40.00 40.00 40.00


################################################################
###    OUTPUT
######################################
#
##########################################
# statistics filename add
##########################################
OutNameUnique  T_0.85_Density_0.800
#########################
# enable, frequency
#########################
OutState   true    1000000
#########################
```

48

```
# Box 0
##########################
OutPDB      LJ_T_0.85_Density_0.800_Run_1_MIE.pdb
###############################################
# (PSF, seed)
###############################################
OutPSF      LJ_T_0.85_Density_0.800_Run_1_MIE.psf
OutSeed     LJ_T_0.85_Density_0.800_Run_1_MIE_seedOut.dat
##############################
# enable, frequency
##############################
OutConsole  true  10000
##################################
# enable, frequency
##################################
OutBlockAverage    true    100000
####################################
# enable: blk avg., fluct., hist.
####################################
OutEnergy          true     true     true
OutPressure        true     true     true
```

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

```
#========================================================
#         GOMC beta 0.94 – Gibbs ensemble Control example      =
#========================================================
#

#####################################################################
###    INPUT
######################################################
#
##################################
# enable, step
##################################
InState     true    1000000
##################################
# kind {RESTART, RANDOM, INTSEED}
##################################
PRNG      RANDOM
##################################
# charmm param, exotic param
##################################
InParam     Par_CHARMM_TraPPE_2_FF.inp  Par_Exotic_TraPPE_2_FF.inp
##################################
```

```
# pdbs for 1st and 2nd box
#################################
InPDB STEP3_START_ISB_sys_BOX_0.pdb STEP3_START_ISB_sys_BOX_1.pdb
#################################
# psfs for 1st and 2nd box
#################################
InPSF STEP3_START_ISB_sys_BOX_0.psf STEP3_START_ISB_sys_BOX_1.psf


############################################################
###     SYSTEM
######################################
#
############################################################
# SYSTEM VARS    #LRC      #Temp (K)  #cutoff (A)   #1-4 Inter.
############################################################
System            true      330.00      10            0.0
############################################################
# STEPS PER...  #total        #till Equil  #Per adjustment
############################################################
Steps           1000000        10000000      1000
##################################################
# MOVE %                 #disp. #rot.  #vol.   #mol.
##################################################
Percent                 0.626  0.10   0.005    0.269
#################################
# box #, x, y, z
#################################
BoxDim  0    71.49 71.49 71.49
BoxDim  1    71.49 71.49 71.49
############################
# growth tr.  first   nth
############################
Growth          53       8


############################################################
###      OUTPUT
######################################
#
#########################################
# statistics filename add
#########################################
OutNameUnique   T_330.00_K
########################
# PDB box 0, 1
########################
```

```
OutPDB ISB_T_330.00_K_Run_1_MIE_BOX_0.pdb ISB_T_330.00_K_Run_1_MIE_BOX_1.pdb
##############################################
# (merged PSF, seed)
##############################################
OutPSF      ISB_T_330.00_K_Run_1_MIE_merged.psf
OutSeed     ISB_T_330.00_K_Run_1_MIE_seedOut.dat
#############################
# enable,frequency
#############################
OutConsole   true  10000
###################################
# enable, frequency
###################################
OutBlockAverage    true    100000
######################################
# enable: blk avg., fluct., hist.
######################################
OutEnergy         true    true    true
OutPressure       true    true    true
OutMolNum         true    true    true
OutDensity        true    true    true
OutVolume         true    true    true
```

GOMC's Output Files, Terminal Output

GOMC currently supports several kinds of output:

- ❖ STDIO ("console") Output
- ❖ File Output
    - ○ PDB
    - ○ PSF
    - ○ Block Averages

Console Output

A variety of useful information relating to variable averages, acceptance rates, file I/O messages warnings, and other kinds of information is printed formatted 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

The first section of this console output typically includes some intro info relating to the forcefield read. This output is important as it may contain text relating to issues encountered if there was an error in the current run (e.g. a bad parameter, etc.

```
GOMC Serial Version 0.96
Started at: Thu Sep  4 04:15:30 2014
On hostname: ▮▮▮▮▮▮▮▮▮▮▮▮▮▮

Reading from GO-MC Configuration File file: ./in.dat

538Finished reading GO-MC Configuration File file: ./in.dat

Reading from CHARMM-Style Parameter File file: ./Par_Mie_CHARMM_style.inp

Finished reading CHARMM-Style Parameter File file: ./Par_Mie_CHARMM_style.inp

Reading from Exotic Parameter File file: ./Par_Mie_Exotic_style.inp

Finished reading Exotic Parameter File file: ./Par_Mie_Exotic_style.inp

Reading from Box 1 PDB coordinate file file: ./STEP3_START_ISB_sys_BOX_0.pdb

Finished reading Box 1 PDB coordinate file file: ./STEP3_START_ISB_sys_BOX_0.pd\
b

Reading from Box 2 PDB coordinate file file: ./STEP3_START_ISB_sys_BOX_1.pdb

Finished reading Box 2 PDB coordinate file file: ./STEP3_START_ISB_sys_BOX_1.pd\
b

Min. Box Size: 8000.1
STARTING SIMULATION!!
```

Next some info on the system's starting configuration will print:

```
------------------------
--    =========     --
--  === BOX 0  ===   --
--    =========     --
------------------------

Volume: 365373 A^3
Tot # mol: 1000

Molecule Type ISB
--------------------
Number: 1000 ;
Density: 0.264161 g/ml

Energy (in K): total: 54787.7
inter: -45328.3 ; inter (tail corr.): -82693.4
intra (bonded): 182809 ; intra (nonbonded): 0


------------------------
--    =========     --
--  === BOX 1  ===   --
--    =========     --
------------------------

Volume: 365373 A^3
Tot # mol: 1000

Molecule Type ISB
--------------------
Number: 1000 ;
Density: 0.264161 g/ml

Energy (in K): total: 54787.7
inter: -45328.3 ; inter (tail corr.): -82693.4
intra (bonded): 182809 ; intra (nonbonded): 0


------------------------
--    =========     --
--  === SYSTEM ===   --
--    =========     --
------------------------

Energy (in K): total: 109575
inter: -90656.5 ; inter (tail corr.): -165387
intra (bonded): 365619 ; intra (nonbonded): 0
```

For most of the run info on the variables and move statistics will be printed, e.g.:

```
STEP 9999 of 40000000 (0.0249975% done)


-----------------------
--   ==========    --
--  === BOX 0  ===    --
--   ==========    --
-----------------------

Volume: 361095 A^3
Tot # mol: 946

Molecule Type ISB
-------------------
Number: 946 ;
Density: 0.252856 g/ml

Energy (in K): total: -491491
inter: -713432 ; inter (tail corr.): -74880.3
intra (bonded): 296821 ; intra (nonbonded): 0

Displace (Box 0) -- tries: 3329; # Accept: 1366; % Accept : 41.0333; Max Amt.: 0.919088
Rotate (Box 0) -- tries: 420; # Accept: 241; % Accept : 57.381; Max Amt.: 3.12267
Molecule Transfer (Box 0 -> Box 1) -- tries: 1265; # Accept: 562; % Accept : 44.4269
Volume Transfer (Box 0 -> Box 1) -- tries: 82; # Accept: 54; % Accept : 65.8537; Max Amt.: 6834.38


-----------------------
--   ==========    --
--  === BOX 1  ===    --
--   ==========    --
-----------------------

Volume: 369650 A^3
Tot # mol: 1054

Molecule Type ISB
-------------------
Number: 1054 ;
Density: 0.275204 g/ml

Energy (in K): total: -619525
inter: -855439 ; inter (tail corr.): -90802.5
intra (bonded): 326716 ; intra (nonbonded): 0

Displace (Box 1) -- tries: 3226; # Accept: 1293; % Accept : 40.0806; Max Amt.: 0.879687
Rotate (Box 1) -- tries: 463; # Accept: 275; % Accept : 59.3952; Max Amt.: 3.14159
Molecule Transfer (Box 1 -> Box 0) -- tries: 1215; # Accept: 508; % Accept : 41.8107
Volume Transfer (Box 1 -> Box 0) -- tries: 0; # Accept: 0; % Accept : 0; Max Amt.: 300

-----------------------
--   ==========    --
--  === SYSTEM ===    --
--   ==========    --
-----------------------

Energy (in K): total: -1.11102e+06
inter: -1.56887e+06 ; inter (tail corr.): -165683
intra (bonded): 623538 ; intra (nonbonded): 0

Steps/sec. : 178.955
```

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.

At the end of the run timing information and other wrapup info will print, such as any failures encountered in the energetic consistency checks:

```
Simulation Time (total): 207330sec.
GOMC Serial Version 0.96
Completed at: Sat Sep  6 13:56:26 2014
On hostname:                    edu
```

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 in the current frame. All molecules are listed in order they were read in. i.e. if box 0 has $1..N_1$ molecules and box 1 has $1..N_2$ molecules, then all of the molecules in box 0 are listed first, then all the molecules in box 1, i.e. $1..N_1$, $N_1+1..N_1+N_2$. PDB frames are written as standard PDBs to consecutive file frames.

To visualize you'll need to open the PDB and PSF outputted files in VMD and use the command:

> pbc join connected

In the TKConsole in order to get the current frame to obey the periodicity. Some peculiarities are 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 outputted to be properly read.

For Gibbs ensemble, the same PSF will be used to visualize either box, i.e.

> 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 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

... for visualizing the second box.

Block Output Files

GOMC tracks a number of thermodynamics variables of interest during the simulation, some independent, others dependent on each other. The variables are tracked by this module:

- Density (g/ml)
- Energy
    - Intermolecular
    - intramolecular bonded
    - Intramolecular nonbonded
    - tail corrections

- o   total interactions
- Virial coefficient
- Intermolecular
- tail corrections
- Pressure (bar)
- total

It is convenient to average the block file using simple shell utilities in Linux:

```
alias avg 'awk -v start_ln=¥!:2 -v column=¥!:3 '"'"'BEGIN {n=0; sum=0; std_dev=¥
0} NR>start_ln { sum+=$column; result[n]=$column; n++} END { avg=sum/n; sum=0; ¥
for(x=1;x<=n;x++){ sum+=((result[x]-avg)^2)}; std_dev=sqrt(sum/n); print avg" "¥
std_dev }'"'"' ¥!:1'
```

```
alias avgd 'awk -v start_ln=¥!:2 -v column=¥!:3 '"'"'BEGIN {n=0; sum=0; std_dev¥
=0} NR>$start_ln { result[n]=$column*1000; sum+=result[n]; n++} END { avg=sum/n¥
; sum=0; for(x=1;x<=n;x++){ sum+=((result[x]-avg)^2)}; std_dev=sqrt(sum/n); pri¥
nt avg" "std_dev }'"'"' ¥!:1'
```

To call this, make sure its directory is in your path and type "./

On the command file the name of the file goes first.

```
avg Blk_Pressure_T_330.00_K_Run_0_MIE.dat  100 2
```

This should display the average, skipping the first 100 lines of the file. Adjust the # of skipped lines as appropriate based on when the system is found to be equilibrated, and adjust the fourth parameter for the desired box (with 1 being box 0, 2 being box 1).

Putting it all Together: Running a GOMC Simulation

To recap the previous examples, a simulation of isobutane will be completed for a single temperature point on the saturated vapor liquid coexistence curve. Please refer to the attached reference directories if you get stuck.

Files for this simulation will be put in the empty folder "1-Isobutane-UserGenerated". If errors are encountered during the building of the system please refer to the folders "2-Isobutane-Reference/1_ScriptsComplete/" and "1-Isobutane-Reference/2_FullyBuilt/" for the proper files and results at various stages of the build.

The general plan for running the simulation is:

1. Build GOMC (if not done already)
2. Copy GOMC GEMC 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 and then traverse to the directory, which should now have a single file "GOMC_Serial_GEMC" in it.

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
*
1  1
!
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. The resulting folder can be compared against the "Expected_Results/Build_Scripts_and_Model" directory in the sample code folder.

Once those files are created, run:

```
./packmol < pack_box_0.inp
./packmol < pack_box_1.inp
```

This will create the intermediate PDBs. Please examine them for correctness (the results obtained can be compared against the "Expected_Results/PDB_Intermediates" directory in the sample code folder.

Then run the PSFGen scripts to finish the system

```
./build_box_0.inp
./build_box_1.inp
```

Note, you probably will have to first make your scripts executable using:

```
chmod a+x build_box_*.inp
```

This will create the intermediate PDBs. Please examine them for correctness (the results obtained can be compared against the "Expected_Results/Complete_PDB_and_PSF" directory in the sample code folder.

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.dat**) in the text editor

```
#==========================================================
#        GOMC beta 0.94 - Gibbs ensemble Control example    =
#==========================================================
#

############################################################
###    INPUT
```

```
#####################################
#
###############################
# enable, step
###############################
InState      true    1000000
#####################################
# kind {RESTART, RANDOM, INTSEED}
#####################################
PRNG      INTSEED    2
#####################################
# charmm param, exotic param
#####################################
InParam      Par_CHARMM_TraPPE_2_FF.inp  Par_Exotic_TraPPE_2_FF.inp
#####################################
# pdbs for 1st and 2nd box
#####################################
InPDB STEP3_START_ISB_sys_BOX_0.pdb STEP3_START_ISB_sys_BOX_1.pdb
#####################################
# psfs for 1st and 2nd box
#####################################
InPSF STEP3_START_ISB_sys_BOX_0.psf STEP3_START_ISB_sys_BOX_1.psf


###############################################################
###    SYSTEM
#####################################
#
###############################################################
# SYSTEM VARS    #LRC      #Temp (K)  #cutoff (A)    #1-4 Inter.
###############################################################
System           true     330.00      10             0.0
###############################################################
# STEPS PER...  #total        #till Equil   #Per adjustment
###############################################################
Steps           1000000      10000000      1000
#################################################
# MOVE %              #disp. #rot.  #vol.   #mol.
#################################################
Percent             0.626  0.10   0.005    0.269
###############################
# box #, x, y, z
###############################
BoxDim  0   71.49 71.49 71.49
BoxDim  1   71.49 71.49 71.49
###############################
```

```
# growth tr.   first   nth
###########################
Growth         53      8


#################################################################
###     OUTPUT
#######################################
#
###########################################
# statistics filename add
###########################################
OutNameUnique   T_330.00_K
#########################
# PDB box 0, 1
##########################
OutPDB ISB_T_330.00_K_Run_1_MIE_BOX_0.pdb ISB_T_330.00_K_Run_1_MIE_BOX_1.pdb
#################################################
# (merged PSF, seed)
#################################################
OutPSF      ISB_T_330.00_K_Run_1_MIE_merged.psf
OutSeed     ISB_T_330.00_K_Run_1_MIE_seedOut.dat
##############################
# enable, frequency
##############################
OutConsole   true  10000
####################################
# enable, frequency
####################################
OutBlockAverage    true    100000
######################################
# enable: blk avg., fluct., hist.
######################################
OutEnergy          true    true    true
OutPressure        true    true    true
OutMolNum          true    true    true
OutDensity         true    true    true
OutVolume          true    true    true
```

Exotic parameters for isobutane:

```
* Parameter file for GO-MC
*
*


DIHEDRALS_POWER_SERIES
!
!V(dihedral) = Kchi * cos(chi - delta)^n
!
!Kchi: kcal/mole
!n: power
!delta: shift
!
!atom types          Kchi      n      delta              description


DIHEDRALS_QUAD_CIS_TRANS
!
!V(dihedral) = Kchi(1 + cos(n(chi) - delta))
!
! Kchi: kcal/mole
!n: 0 - cis; 1 - trans
!delta: shift
!
!atom types          Kchi      n      delta              description


NONBONDED_MIE
!
!V(mie) = 4*eps*((sig_ij/r_ij)^n-(sig_ij/r_ij)^6)
!
!atom    eps          sig_ij      n      description
```

CHARMM format Parameter file for CHARMM isobutene:

```
* Parameter file for GO-MC
* Contains parameters for isobutane (TraPPE 2 FF)
*
```

```
BONDS
!
!V(bond) = Kb(b - b0)**2
!
!Kb: kcal/mole/A**2
!b0: A
!
! Kb (kcal/mol) = Kb (K) * Boltz. const.; (9999999999 if no stretching)
!
!atom type      Kb              b0          description
CH3 CH1         9999999999      1.540    ! TraPPE 2
ANGLES
!
!V(angle) = Ktheta(Theta - Theta0)**2
!
!V(Urey-Bradley) = Kub(S - S0)**2
!
!Ktheta: kcal/mole/rad**2
!Theta0: degrees
!Kub: kcal/mole/A**2 (Urey-Bradley)
!S0: A
!
! Ktheta (kcal/mol) = Ktheta (K) * Boltz. const.
!
!atom types        Ktheta       Theta0    Kub(?)   S0(?)
CH3 CH1 CH3        62.100125    112.00 ! TraPPE 2

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
```

```
NONBONDED nbxmod  5 atom cdiel fshift vatom vdistance vfswitch -
!cutnb 14.0 ctofnb 12.0 ctonnb 10.0 eps 1.0 e14fac 1.0 wmin 1.5
!
!V(Lennard-Jones) = Eps,i,j[(Rmin,i,j/ri,j)**12 - 2(Rmin,i,j/ri,j)**6]
!
!epsilon: kcal/mole, Eps,i,j = sqrt(eps,i * eps,j)
!Rmin/2: A, Rmin,i,j = Rmin/2,i + Rmin/2,j
!
! Rmin = sig * (2^(1/6)) / 2 ; eps (kcal/mol) = eps (K) * Boltz. const.
!                              Boltzmann = 0.0019872041 kcal / (mol * K)
!
!atom  ignored    epsilon      Rmin/2        ignored  eps,1-4  Rmin/2,1-4
CH3    0.0       -0.194745992  2.10461634058 ! TraPPE 1
CH1    0.0       -0.019872040  2.62656119304 ! TraPPE
```

Once these four files have been added to the output directory the simulation should be ready to run. The files and directory can be compared to "Expected_Results/Ready_To_Run" directory in the sample code folder.

Assuming the code is named GOMC_Serial_GEMC run using:

```
./GOMC_Serial_GEMC > out_ISB_T_330.00_K_RUN_0.log &
```

Progress can be monitored with the tail command:

```
tail -f out_ISB*log
```

When the run is done, please compare the results to those found in "Expected_Results/Simulation_Output" directory in the sample code folder. Use the "avg" alias, added in the previous section to determine the equilibrated block averages.

Congratulations, a single phase coexistence point on the saturated vapor liquid curve has been examined using GOMC operating in the Gibbs ensemble.

This process can be repeated additional times to produce new results (saving the old results aside) to produce averages. It is recommended that at least three simulations be run per point, although for larger/longer simulations sometimes it is appropriate to run only a single simulation.

Isobutane
TraPPE FF

Mon Sep 15 17:10:21 2014
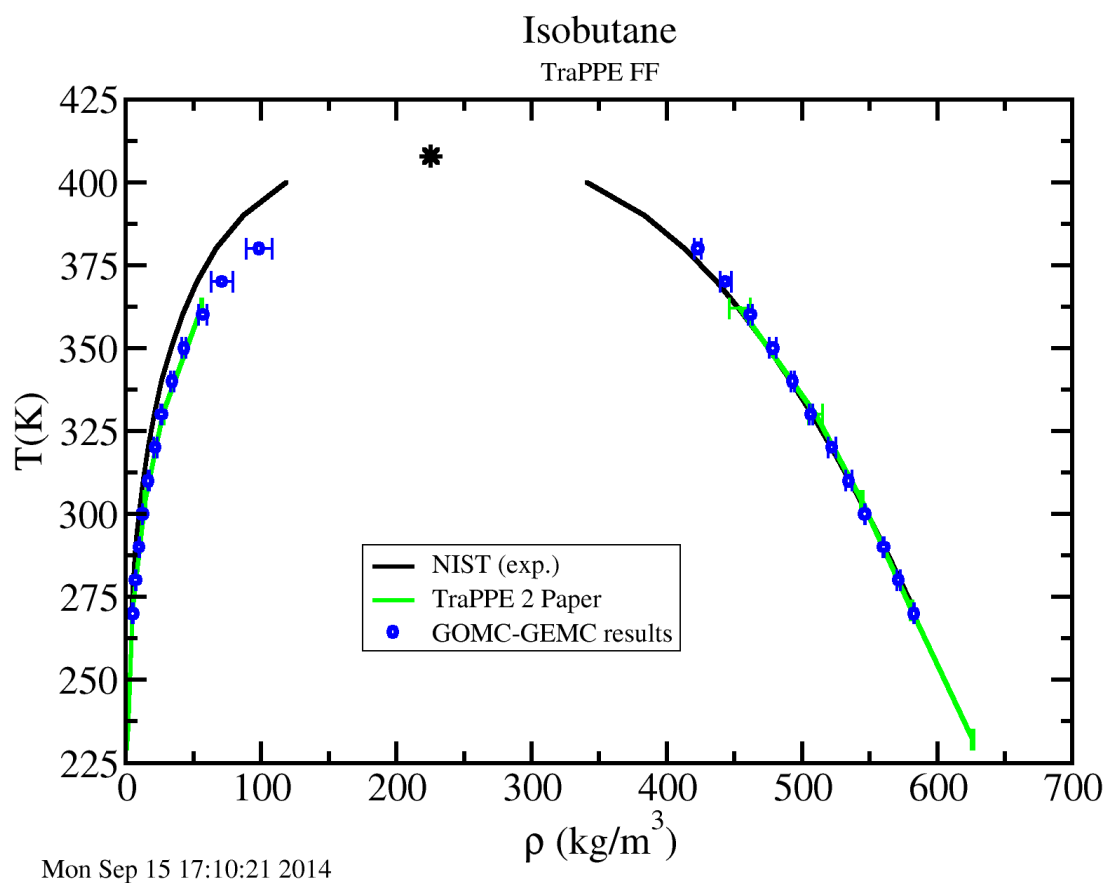
Repeating this process for multiple temperatures will allow you to obtain the following results. This data is contained in the folder "Expected_Results/TraPPE_Isobutane_VLE_Graphs".

Summary of Algorithms in the Code

NVT Ensemble

Gibbs Ensemble

Configurational Bias

GPU Algorithms/Optimizations

How to Get Help/Technical Support

Please send an email to **gomc@eng.wayne.edu**