InterMol: User's Guide

Shirts Group
University of Virginia
Department of Chemical Engineering

March 12, 2012

Contents

1 Preface **3**

1.1 Overview 3

1.2 Prerequisites 3

1.3 Installation and Development 4

1.4 Compiling the Documentation 4

2 Running InterMol **4**

2.1 Gromacs Modules 4

1 Preface

1.1 Overview

This work has served as preliminary data for an NSF "Software Infrastructure for Sustained Innovation" grant under review. Grant proposal info: NSF OCI-1148410, Project title: "Collaborative Research: SI2-SSE: A general, extensible software framework for automated preparation of molecular simulations"

Increases in computational power have made it possible to simulate complex chemical systems, providing valuable insight and predictive capability that could not be obtained by experiments alone. As computational power increases, a larger and larger burden to perform these calculations is the setup time[1]. To run a simulation a series of several complex tasks must first be completed to prepare a set of molecular coordinates and parameters. While there exist some tools to facilitate in the automation of this process, many of these scripts are dependent on a particular simulation package. To resolve these interoperability limitations, there is an increasing need to bridge the file format gap between simulation packages.

This work serves to facilitate the simple conversion of atomistic representations between molecular simulation packages. There have been many prior examples of scripts to convert between different formats such as `acpype`[2] which converts between the Amber simulation format to the GROMACS. Due to the one way nature of such scripts one would require N^2 total scripts to convert between N different simulation packages. Through the use of a universal abstract representation of classical molecular systems, it is possible to reduce the number of necessary scripts to N in order to convert between N packages. Essentially to convert to or from a particular format one must code a translator to and from the universal abstract. Thus a typical workflow to convert to and from one format to another, for example the Maestro[3] format to the GROMACS would require reading in a Maestro simulation into the abstract representation. Then from the abstract representation convert to a GROMACS compatible simulation.

Using InterMol to bridge the file format gap, it will be possible for previously setup simulations to be ported across simulation packages to take advantage of specific tools offered by one package but not another. This ability to reuse previous simulations will be invaluable as it prevents the unnecessary duplication of previous work. Furthermore, the introduction of InterMol will allow the easy interconversion between file formats and force fields which presents researchers with a novel method to benchmark and compare across simulation packages and force fields. Finally, the ability to use simulation setup pipelines such as Maestro across all simulation packages will vastly reduce the burden for researchers to learn how to build systems of interest. Automation will also facilitate the spread of best practices in molecular simulations to prevent human errors.

1.2 Prerequisites

InterMol is a collection of python scripts which run on Intel based Linux systems running CentOS 5.4 or later. Certain scripts require a recent version of Python; we recommend version 2.6.0 or greater. The package is dependent on the `Simtk.units` package which should be available via svn

[1]Chodera, J. D. et al. *Current opinion in structural biology* **Apr. 2011**, *21*, 150–60.
[2]Sousa Da Silva, A. W. et al.
[3]Maestro, version 9.2, Schrödinger, LLC, New York, NY, 2011.

from the simtk openmm website.

1.3 Installation and Development

There are future plans to create a `distutils` or `ez_setup` compatible installation. However neither the `ez_setup` option nor the standard `setup.py` option is currently available. Instead `InterMol` is currently hosted by simtk, a part of the Simbios project funded by the National Institutes of Health. A bleeding edge copy of `InterMol` can be checked out anonymously from the mmtools svn. If the InterMol package is not checked out to a directory along the `PYTHONPATH` then the location of the InterMol module must be given.

```
svn checkout https://simtk.org/svn/mmtools/ctools
export PYTHONPATH=$PYTHONPATH:/home/mst3k/python-modules
```

For developers who are interested in committing changes to the software, one must contact an administrator for the mmtools group on the Simtk website. Once your account is created and given access, then you may make modifications as necessary.

1.4 Compiling the Documentation

In order to compile the online documentation, the `Sphinx` package must be installed. Furthermore the `autodoc.py` distributed with `Sphinx` should be patched to allow for documentation of private members. The patch can be found in the distribution. The documentation can be compiled using the included Makefile.

```
make html
```

2 Running InterMol

2.1 Gromacs Modules

Currently only the Gromacs Modules have been implemented for InterMol. Before using the modules it preferred to have a working copy of Gromacs available online at `http://www.gromacs.org/`. Gromacs takes advantage of 'C'-style #include and #define directives which can only be read when the force field libraries of Gromacs are available to InterMol. To ensure that these libraries are accessible you must define the `GMXBIN` and `GMXLIB` environment variables which give the directory of the binary as well as the top level directory of the topology files respectively:

```
export GMXBIN=/usr/local/gromacs4.5.3-dev-fep/bin
export GMXLIB=/usr/local/gromacs4.5.3-dev-fep/share/gromacs/top/
```

A few basic test cases have been previously implemented including `testGMX.py` and `testGMX2.py` which can be found in the `testfiles` directory. Additional test cases will be added as development continues.

```
python testGMX.py
```