Amber INDUS Notes

Created by Emily Baum in collaboration with the Bilodeau Group

UVA Department of Chemical Engineering

Last Updated 5/1/2024

Table of Contents

Summary

INDUS simulations are used to bias water molecules out of a control volume to measure free energy of cavitation in systems of interest.This guide will attempt to replicate the example following section 7 of the INDUS Manual, linked here: https://github.com/patellab511/indus/blob/master/manual/plumed_indus_manual.pdf.

The first step to using INDUS is calculating the correct parameters to use when biasing your simulation. To do this, run an unbiased simulation of your system of interest, in this case we will use a box of pure water. This is left as an exercise for the reader. To do this simply, copy the “copy_sim” folder, edit the 03_Prod.in file and delete the “plumed=1” and “plumedfile=plumed.dat” lines, then run the simulation with ./shell.sh. Make sure you are in a conda environment with MDAnalysis so that indus_params.py can run.

Once you have run a simulation of a box of water, you can use the ntilde.py file to calculate the biasing parameters; the formula for these comes from the INDUS manual. These parameters depend on the size and shape of your control volume. We can reproduce the results thus far, finding that $\Delta N^* \sim 5$ and $k \sim 1$ kJ/mol, with an average of about 30 water molecules in the box. We can directly plug in the value for k into the simulations, but we need to iterate over multiple values of N*. N* is the “average coarse grained number of water molecules in the volume,” which is a faster estimator than counting all the atoms themselves. INDUS applies a bias to this coarse grained model of water.

So we iterate our simulation in intervals of $\Delta N^* \sim 5$, from 25 down to 0 (we already have it at 30 from the unbiased simulation). To do this, I put the different simulation parameters into sim_list.txt, which is used by the run_sims script (./run_sims sim_list.txt) to run simulations at each of these bias settings.

At this point, INDUS gets angry. The simulation doesn’t really do much unless the value for N* is negative. Also, the bias seems to stop being applied after only a few nanoseconds. See the probability distribution plot from the example in biased_AT_20. Toward the beginning of the simulation, the average hovers around 0, then “jumps” to a new regime where it explores with an average of around 25. No matter how I change the settings, I can’t get any of the simulations to explore an intermediate range; it always undersamples between 0 and the true average of ~30 waters in the box. Surprisingly, the plumed.out file seems to be generated improperly, as a count from the trajectory using MDAnalysis (indus_params.py) does not match the indus.n output.

The results from WHAM support the error I am getting. The probabilities of N=0 and near N = 30 are predicted fairly well, but the region between them is undersampled. This results in a cubic looking function, rather than the expected parabola.

Dependencies

To use this package, you will need:

- Amber MD software
- An environment with python with MDAnalysis
- PLUMED 2.4.0
- INDUS
- WHAM

INDUS Installation

Installing INDUS is a bit of a pain if you don't know the right sequence of steps. You will want to also download PLUMED 2.4.0 so that you can run the biased simulations with INDUS. Follow the instructions below:

Download PLUMED here: https://github.com/plumed/plumed2/releases/tag/v2.4.0
Download INDUS here: https://github.com/patellab511/indus/tree/master

- Download INDUS and PLUMED 2.4.0 files first
- Patch INDUS into PLUMED before installing PLUMED

To patch INDUS into PLUMED, run the following command:

A shell script to aid you in the patching process is available in the INDUS code repository under `indus/plumed_patch/patch_plumed.sh`. When you run the script, simply pass to it the location of the root directory of the PLUMED repository (*i.e.* the one that contains the `Makefile` and so on):

```
./patch_plumed.sh <location-of-PLUMED-source-code>
```

Now that PLUMED is patched, it needs to be installed. Move to the PLUMED directory, then run the following commands:

- ./configure --disable-mpi --prefix=$HOME/opt
- make -j 4
- make check
    - This checks that the code is compiled correctly
- ./configure --disable-mpi --prefix=$HOME/opt
- make
- make install
- make installcheck
    - This checks that the code is installed correctly

You'll also need to install WHAM, available here: http://membrane.urmc.rochester.edu/

?page_id=126. Follow the directions to build the standard 1-D wham code. You can also use the version installation in this github.

Simulation

All of the necessary input files are available in the copy_sim folder. The tleap.in file creates the water box. 01_Min.in, 02_Heat.in, and 03_Prod.in give all the necessary simulation steps. The INDUS biasing is done through plumed.dat and indus.input. The shell.sh script runs everything in the proper order, as well as creates summary files. The probability distribution is plotted and saved as a time series for WHAM by indus_params.py. Outside of this, run_sims takes in the sim_list.txt for the INDUS biasing parameters and runs all of the simulations in their own folders. Despite the undetermined issue that impacts the sampling of INDUS, the shell scripts and modular setup of the simulations may be useful in future work for automation purposes. The python scripts serve as useful tools to properly calculate the biasing parameters for INDUS and also for using MDAnalysis to ensure that the INDUS output matches the simulation trajectory.

Example

Some results of the simulation are given in the github to follow along with. The .nc files from the simulation are too large to upload, but the remaining files for one of the biased simulations (with AT=20) are given in the biased_AT_20 folder. Additionally, the combined probability distribution results using WHAM are given in wham-release-2.0.11/wham_results. I have given the metadata file along with the command used directly by WHAM, the raw results from WHAM, as well as a python script creating the resulting plot of the WHAM output.