Transition Path Sampling Tutorial

Introduction

Consider a molecular system that can be in two stable states, which we label as state A and state B. An example is a protein complex, which can be in a folded or an unfolded state. To observe the way in which the system transitions from state A to state B, we can apply Molecular Dynamics (MD) methods to simulate the system for an amount of time. Traditional MD methods can have trouble capturing a transition, however, if this transition is a **rare event**. MD simulations can be done up to an order of microseconds, which is not sufficient to capture transitions that happen at a tbeyond that timescale. This problem can be solved by performing a metadynamics simulation. For this method, we introduce a bias potential that drives the system from state A to state B. In some cases, we can also force a transition by increasing the temperature of a traditional MD simulation, which can help proteins unfold, for example. Both these methods will provide a transition path, but not one under normal conditions.

If we are interested in observing the transition under normal conditions, we can apply **transition path sampling** (TPS), a rare event sampling method. TPS methods require a single transition path as an input, and can produce many new, independent ones. The input trajectory can be produced using a high temperature MD simulation, or a metadynamics simulation.

TPS is a Monte Carlo sampling method of the path space, meaning that it samples MD trajectories that connect the states A and B in an unbiased fashion. These two states are assumed to be stable, or at least semi-stable, so that the system is unlikely to exit the state once it enters it. TPS selects a point in this trajectory, called the 'shooting point'. From this point a 'shooting move' is made. In a shooting move, a new MD simulation is started with the shooting point as a starting state. A shooting move can be backward in time or forward in time in the case of one-way shooting, or both in the case of two way shooting. This shooting move replaces the original trajectory in the case of two-way shooting, and the part before or after the shooting point in the case of one-way shooting (depending on the direction of the shooting move). A shooting move can be accepted or rejected. If the new trajectory does not connect the two states A and B, the new shooting move is instantly rejected. If the two states are connected, the Metropolis acceptance rule

$$P_{acc} = \min(1, \frac{N^{(o)}}{N^{(n)}}) \tag{1}$$

where P_{acc} denotes the probability of accepting the new path, $N^{(o)}$ is the length of the original path, and $N^{(n)}$ is the length of the new path. To avoid accepting paths that are too long, a second stage is introduced in the acceptance process: a maximum pathlength $N_{\max}$ is defined before the shooting procedure starts, and defined as

$$N_{\max} = \frac{N^{(o)}}{\xi} \tag{2}$$

where ξ is a uniformly distributed random number between 0 and 1. New paths longer than the maximum length are rejected by default. If the shooting move is rejected, a new move is performed.

The two stable states A and B, as well as the transition region I, are defined using order parameters. These order parameters are properties of the molecular system that can be expressed numerically for every timestep. For instance, it is possible to define the stable states using the distance between two specific atom (to test whether a hydrogen bond exists), by examining the radius of gyration to examine the compactness of the molecule, or by tracking a dihedral angle.

The idea is that (hopefully) the shooting point is chosen in such a way that the 'committor probability' is 0.5: if you start an MD simulation at this frame, there is a 50% chance to reach both state A and state B. A new shooting point is then picked from the trajectory of the shooting move, and the process is repeated. Generally the shooting point is chosen at random, although methods are available to steer the shooting point towards a point where the committor probability is a half. After several shooting moves, new paths can be constructed that go from A to B, and are decorrelated from the original trajectory. A TPS algorithm that performs a forward and backward shooting move

every step is labeled as a two-way shooting algorithm, whereas those that do a forward or backward shooting are labelled one-way shooting algorithms. TIPSI belongs to the latter category, and only does one move per step, randomly picking forward or backward at each attempt. The one-way shooting procedure is used in TIPSI as it improves the acceptance ratio of the shooting moves.

In this tutorial the useage of the TIPSI (Transition Interface/Path Sampling Identifier) package version 0.1 is demonstrated, a script that performs TPS using the GROMACS MD engine [1, 5]. TIPSI is an application of rare event sampling, designed to sample many possible transition paths in a molecular system with two stable states, A and B, where transitions between the two states are extremely rare. In this tutorial, two examples will be discussed. First, we will provide a tutorial designed to have short computational times. This tutorial concerns alanine dipeptide, and samples a transition between its two conformations. These two states are easily visually distinguishable, but the transition is not actually a rare event. The second example concerns a transition which is a rare event: the flipping of DNA basepairs from the standard Watson-Crick pairing to the non-standard Hoogsteen pairing. This transition typically does not occur even if the system is simulated for microseconds. This example uses a provided result of a metadynamics simulation as input.

TIPSI relies on GROMACS version 4.5.4 to do molecular simulation. For this tutorial, GIT is required to pull the repository with the demonstration files. For visualizing structures, the VMD package is recommended. This tutorial is written assuming you work on a Linux system. We assume some knowledge in working with Linux, as well as a basic understanding of how to work with GROMACS. There will be a cheatsheet in the appendix of this tutorial for those less familiar with either. Finally, it is assumed you are working on Carbon, the clustercomputer of the Computational Chemistry group at the Universiteit van Amsterdam. Both use the `TIP3P` water model. The forcefield used in the first tutorial is the `AMBER99SBILDN`, the send uses another `AMBER` forcefield included in the repository.

Dipeptide Alanine

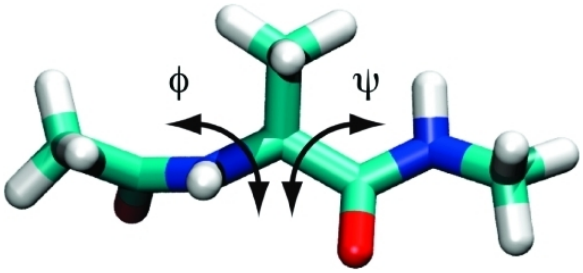

Figure 1: Alanine dipeptide, dihedrals labeled ϕ and ψ, from [2].

Alanine dipeptide is a small organic molecule, consisting of the alanine amino acid with two minimal peptide bonds. This molecule has two stable states, defined by the configuration of the dihedral angles ϕ and ψ (Fig. 1). In this first part of the tutorial, we will generate transition paths between the two states using TIPSI. To run an analysis TIPSI, we need to supply an intial trajectory to TIPSI in which the transition takes place at least once. We also need to define the stable states between which the transition takes place in a quantitative way. To get an initial trajectory, we do a preparatory MD analysis of the molecular system first.

Preparatory Analysis

First, we log into the Carbon cluster. Make sure you are connected to the UvA network or VPN, and enter

```
ssh user@carbon.science.uva.nl
```

When logged into the account, pull the tutorial repository to your system, which can be done by

```
git clone https://github.com/dgoldsb/tipsitutorial
```

which will copy the tutorial directory as a new directory in your current folder. Exploring the folder shows the following subfolders:

```
bin documents dipeptala-input dnabr-input
dnabr-tipsi dicynebts exampleresults README.md
```

The folder `bin` contains some analysis tools we will use later, as well as the `TIPSI` distribution. All input files required for this part of the tutorial are included in `dipeptala-input`, and output will be written into a new folder `output`. The `documents` directory contains supporting reading material.

We enter the directory `dipeptala-input` and explore, which shows the following 9 files:

```
dipeptala.pdb em.mdp  emw.mdp highT.mdp md.mdp
posre.mdp submit_highT submit_md submit_tipsi
```

The `.pdb` file contains the structure of the molecule we simulate, in our case alanine dipeptide. To view this structure, copy the file to your own PC using a secure copy command, and view the structure in VMD. A nice piece of software that can help in copying files is FileZilla, which provides a GUI for secure copy.

To run an MD simulaton we first need to obtain the topology. We can do this by entering

```
pdb2gmx -ignh -f dipeptala.pdb
```

picking option 6 for the `AMBER99SBILDN` forcefield, and option 1 for the `TIP3P` water model subsequently. The option `-ignh` specifies that hydroges should be ignored. Three new file should be created, namely a topology file `topol.top`, a list of atoms for a position constrained run `posre.itp`, and a set of coordinates in GROMACS format `conf.gro`.

Next, we place the protein in a periodic box with a 1 nm distance between the protein and the boundary.

```
editconf -f conf.gro -bt dodecahedron -d 1 -o box.gro
```

We fill up the box with water molecules by running

```
genbox -cs -cp box.gro -p topol.top -o solve.gro
```

and energy minimize this system. The settings for this MD simulation are saved in emw.mdp. Before starting the minimalization process, we combine the coordinates, topology and parameters into a `.tpr` file.

```
grompp -f emw.mdp -c solve.gro -p topol.top -o emw.tpr
```

We run this by entering

```
mdrun -deffnm emw -v
```

The option `-v` requests verbose output. We see that the solvent does not contain ions, thus we add 100 mM of NaCl to the system by replacing water molecules in the solvent. First, we make a new input file `ion.tpr`.

```
grompp -f emw.mdp -c emw.gro -p topol.top -o ion.tpr
```

We add ions to this as follows

```
genion -s ion.tpr -p topol.top -neutral -conc 0.1 -o ion.gro
```

We pick option 13 to replace water molecules with Na^+ and Cl^{-1} ions. For a system this small, only one or several of each will be added. We now energy minimize the system again.

```
grompp -f em.mdp -c ion.gro -p topol.top -o em.tpr
mdrun -deffnm em -v
```

To relax the water and ions around the protein we perform a position constrained run. This implies that all non-hydrogen atoms in the protein are fixed in their position.

```
grompp -f posre.mdp -c em.gro -p topol.top -o posre.tpr
mdrun -deffnm posre -v
```

To visualize the trajectory later, we need to make a snapshot of the system at the start of the MD simulation. IN order to do so, we run

```
trjconv -f posre.gro -s posre.tpr -pbc mol -center  -ur compact -o dipept_show.gro
```

We pick option 1 for our first option to center the protein. The second option depends on your hardware and the length of your simulation. In this case we can pick option 0, as the system is small. If your simulation is long, the system is big, or your RAM is limited, option 1 is preferred to ignore solvent molecules. The output `dipept_show.gro` can be visualized in VMD.

Now that we have relaxed the solvent we can run an MD simulation. We want to ensure that the transition between the two states takes place, so it is typically not a bad idea to run at a higher temperature (400K). The resulting trajectory will serve as an input for TIPSI, and will be used to find many more transitions.

```
grompp -f highT.mdp -c posre.gro -p topol.top -o highT.tpr
qsub submit_highT
```

We will also run the simulation at room temperature, but for a shorter duration, in order to define the stable states. The `.tpr` file in which the MD simulation conditions that you are interested in are defined is also an input for TIPSI. These conditions will be used for all simulation run in the TPS process. We want these simulations to take place at room temperature (298K). We generate the `.tpr` and run the simulation as follows:

```
grompp -f md.mdp -c posre.gro -p topol.top -o md.tpr
qsub submit_md
```

The output will be put in the `output` directory.

Running TIPSI

Before all else, we should check if the right files are flagged as executable. The file `/documents/executables.txt` contains a list of all files in the `/bin/tipsi/`-directory that should be flagged as executable. Executable files should show green when using the `ls` command in this directory. Making files executable is done with `chmod +x file`.

To tell TIPSI how to define the stable states, we have to set up a parameter (`.par`) file. To find if there are stables states between which a transition can take place, we need to express our trajectory in terms of some collective variable. In this example, we use two dihedrals, ϕ and ψ. We create a an index file defining these dihedrals by going to the output directory of the room temperature simulation, and run

```
vim angle.ndx
```

after which we copy-paste the following:

```
[ Phi ]
5 7 9 15

[ Psi ]
7 9 15 17
```

After we save and quit (`:wq` in vim), we analyze our simulation using the dihedrals describedz in the index file. We run

```
g_angle -f md.xtc -od angle.ndx -ov dihed_phi.xvg -type dihedral
```

with option 0 and

```
g_angle -f md.xtc -od angle.ndx -ov dihed_psi.xvg -type dihedral
```

with option 1 to generate a dataset of the dihedral angles. We can visualize how often each combination of these two variables is visited by running

```
python ../../bin/generate2Dbins.py dihed_phi.xvg 1 dihed_psi.xvg
1 -180 180 -180 180 100 100 Phi Psi Dihedrals\ Landscape
```

This script will generate a set of plots in `.pdf` format. Transfer the plots to your computer and view them. You will see that there are two combinations of the dihedral angles that are visited a lot by the simulation. We see that the area between these two states are not visited often. As such, they can be considered semi-stable, and are our A and B state for TIPSI. Note down the ranges for both variables at which the molecule is in a semi-stable state. Be sure that the system does not go outside this range while still being in the stable configuration, as this is problematic for TIPSI.

Go to the `bin` directory, and look at the `tps-dipept.par` file. In this file, we define the maximum length of a transition path, as well as when the system is in state A, state B, or inbetween (I). Confirm that we confirmed the state in accordance with the range that you noted down. There is little documentation so far on how to include other variables in the parameter file, so some alternative options are included in an appendix.

We included groups of atoms in out parameter file, namely `phi` and `psi`. We must define this custom group in an `.ndx` file to pass this group to TIPSI. If explore the `run_tipsi-dipept` file, we see that the `angle.ndx` file is taken as input. To make sure that this file is copied to the node, we make a copy of the file in the `dipeptala-input` directory. In `run_tipsi-dipept` , we can also set the number of processors used by TIPSI, which should match the number we assigned in the submit script (8 processors), as well as the number of shooting moves performed, which in this case is set to 10. As a sidenote, by generating an index file with GROMACS, you can use any of the GROMACS standard groups.

Finally, we need to copy our initial trajectory to our input directory. Change your directory to your high temperature MD output, and exectute

```
cp highT.trr ../../input-dipeptala/highT.trr
```

Now that we set up the parameter file, we need to edit our run script to take the correct input. Open the file `run_ tipsi-dipept`, and check if the `.tpr` file (for the simulation setting) and the `.trr` file (for the trajectory) are set correctly. Run TIPSI by entering

```
qsub submit_tipsi
```

Analyzing Results from TIPSI

Running TIPSI takes some time. While it runs, go to the directory `exampleresults/DATA`. This folder contains some output generated in TIPSI, bar the output trajectories. Explore the directory, you should see

```
0 1 2
```

This implies that three shooting moves have been done. Now go to the directory `0/0`, and do

```
more 0-0.dat
```

Can you confirm that a transition took place? Now move up two directories, and go to the directory `2`. Explore this directory, you should see

```
0 1
```

This means there have been two attempts at a shooting move until one was accepted. Move to the directory `0`, and execute

```
more 2-0-FW.dat
```

This is a forward shooting move, as indicated by `FW`, and thus advances in time. Can you see why this shooting move was not accepted? Finally, move up one directory, and go to the directory `1`. This is a backward trajectory, as indicated by the `BW` flag. Execute

```
more 2-1-BW.dat
```

TIPSI does backward shooting moves by taking a negative timestep, can you confirm this? Can you see why this trajectory was accepted? The total trajectory, if accepted, is also described in a `dat` file. Execute

```
more 2-1.dat
```

Can you identify which parts of the trajectory come from which shooting move?

The `bin/match_paths.sh` script should put together all the trajectories automatically before the output is copied from the node to you own directory. All paths can be found in the `output/jobname/DATA` directory. Copy a complete trajectory to your own PC to confirm that the trajectory is correct, by visualizing it in VMD. You can compare your result to the trajectory in the `exampleresults` folder.

The `bin/match_paths.sh` script also output a file that contains some data on the TPS process. We can process this information by running

```
python ../../../bin/processTIPSImerge.py metainfo.csv
```

This has to be done on your own machine, however, as an image is outputted to the screen, and Carbon runs outdated Python libraries. The output will detail the average path length of the accepted paths, number of decorrelated paths, the shooting tree, and the ratio between forward shooting moves and backward shooting moves. Did the shooting process take place as you had expected? If you feel brave, you can check if there were no warnings or errors by opening `mergelog.txt` and searching for WARNING/Warning/warning and error.

DNA Baseroll

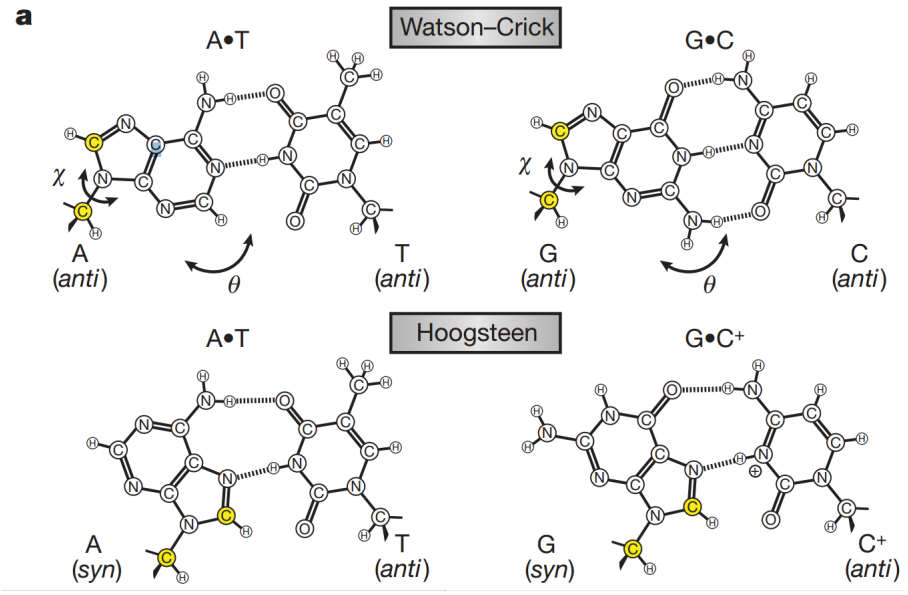

Figure 2: Hoogsteen and Watson-Crick pairing, from [3].

For the second example, we examine the case of the 'DNA baseroll', based on a project by Jocelyne Vreede and David Swenson (the original presentation is included in the `documents` directory) [4]. This is an actual rare event, in which a DNA basepair transition from the Watson-Crick configuration to the Hoogsteen pairing, or vice versa. These two configurations are illustrated in Fig. 2. This event typically does not occur when running a normal MD simulation for a microsecond, but using TIPSI we will generate several new transitions.

Preparation

We again log into the Carbon cluster. Make sure you are connected to the UvA network or VPN, and enter

```
ssh user@carbon.science.uva.nl
```

We enter the directory `dna-br` and explore, which shows the following files:

```
conf.gro run.mdp submit_md topol_DNA_chain_A.itp
topol_DNA_chain_B.itp topol_DNA.itp topol_Ion2.itp topol.top top
```

First, we need to install a custom topology, which we do by running

```
export GMXLIB=./top
```

We can generate the `tpr` file necessary to run TIPSI by executing

```
grompp -f run.mdp -c conf.gro -p topol.top -o md.tpr
```

after which we move this file to our directory for TIPSI input

```
mv md.tpr ../dnabr-tipsi/
```

Normally it is necessary to run a regular MD simulation to confirm that the two states are stable, but in this example this is a given. This can still be done by running the submit script, if you are interested in confirming that the transition does not happen at room temperature in a regular MD simulation.

Running TIPSI

To tell TIPSI how to define the stable states, we have to set up a parameter (`.par`) file. The stable states are in this case define by whether certain hydrogen bonds exist. In Fig. 2, we can see the hydrogen bonds formed in both pairings. We are dealing with an A-T basepair, so there should be two hydrogen bonds in either configuration. One hydrogen bond is always present, and is between atoms 275 and 492. The hydrogen bond unique in the Watson-Crick pairing is between atoms 276 and 495, and the bond unique to the Hoogsteen pairing is between 276 and 289.

Go to the `bin` directory, and look at the `tps-dnabr.par` file. In this file, we define the maximum length of a transition path, as well as when the system is in state A, state B, or inbetween (I). Confirm that we confirmed the state correctly. Notice that the numbers are not the same! This is because we define the atoms now in a Numpy array. This array is 0-indexed, whereas GROMACS starts counting at 1. For more information on this, read the appendix on TIPSI parameter files.

Now that we set up the parameter file, we need to edit our run script to take the correct input. Open the file `run_tipsi-dnabr`, and check if the `.tpr` file (for the simulation setting) and the `.trr` file (for the trajectory) are set correctly. Run TIPSI by entering

```
qsub submit_tipsi
```

in the `dnabr-tipsi` directory.

Analyzing Results from TIPSI

As in the previous example, you should compare your result to the transition in the `exampleresults` folder to verify that you captured the transition. The basepair of interest can be isolated in VMD by visualizing only `resname DT and resid 9`, and `resname DA and resid 4`. If you use the RMSD trajectory tool in VMD, make sure to change 'protein' to 'nucleic' before hitting 'ALIGN'. You can process the information in the TPS process again by running

```
python ../../../bin/processTIPSImerge.py metainfo.csv
```

on your PC. Is the ratio between backward and forward shooting moves about 0.5? And is the tree similarly shaped as the tree from example 1?

References

[1] M.J. Abraham, D. van der Spoel, E. Lindahl, and B. Hess. Gromacs user manual version 4.5.4, 2010.

[2] CP2K: Open Source Molecular Dynamics. Ramachandran plot for alanine dipeptide, 2014. [Online; accessed July 8, 2016].

[3] Evgenia N Nikolova, Eunae Kim, Abigail A Wise, Patrick J OBrien, Ioan Andricioaei, and Hashim M Al-Hashimi. Transient hoogsteen base pairs in canonical duplex dna. *Nature*, 470(7335):498–502, 2011.

[4] Jocelyne Vreede and David W. H. Swenson. Dna baserolling: A friday afternoon project, 2013.

[5] Tsjerk A. Wassenaar. Transition interface/path sampling identifier, 2014.

Linux Cheatsheet

- **Display current directory**: `pwd`
- **View contents of the current directory**: `ls`
- **Change the directory**: `cd ./your/directory/here`

  Note: the currect directory is referred to as `./`, the directory in which your current directory resides is `../`
- **Copy a directory/file**: `cp ./target ./new/location/`

  Note: the option `-r` (recursive) is required for directories, this implies all the contents should be copied
- **Delete a directory/file**: `rm ./target`

  Note: again the recursive option is required for directories, and mind that deleted directories cannot be recovered.
- **Read a file**: `more ./target` or `head ./target` or `tail ./target`
- **Make a file executable**: `chmod +x ./target`

GROMACS Cheatsheet

- **Check the temperature of a simulation**:

```
gmxdump -s target.md | grep ref_t
```

- **Check aspects of a trajectory**:

```
gmxcheck -f target.trr | more
```

- **Add a forcefield (for DNA baseroll)**:

```
export GMXLIB=./top
```

- **Obtain a centere** `.gro` **of the system**:

```
trjconv -f target.gro -s target.tpr -pbc mol -center -ur compact -o output.gro
```

- **Convert a** `.trr` **to a** `.xtc`:

```
trjconv -f target.xtc -s target.tpr -pbc mol -center -ur compact -o ouput.xtc
```

- **List distance between two atoms or center of mass**:

```
g_dist -f target.trr -s target.tpr -n index.ndx -o pdist.xvg
```

- **List the RMSD**:

```
g_ -f target.xtc -s target.tpr -o rmsd.xvg;
1; 1
```

- **List the free energy**:

```
g_energy -f target.edr -o free_energy.xvg
```

- **List the number of helical hydrogen bonds**:

```
g_hbond -f target.xtc -s target.tpr -hx hbhelix.xvg; cat hbhelix.xvg | grep -v "^\#"| grep -v "
```

- **Track a dihedral**:

```
mk_angndx -s target.tpr -n angle.ndx -type dihedral; g_angle -f target.xtc -od angles.ndx -n di
```

- **List radius of gyration**:

```
g_gyrate -f target.xtc -s target.tpr -o gyrate.xvg
```

TIPSI Cheatsheet

Running parameters

In general, we run TIPSI using the `run tipsi`-script included in the `./bin/tipsi/calculator`-directory. Examining this script reveals that TIPSI takes the following input parameters in the case of this tutorial:

```
./tipsi/tipsi.sh -v -tpr md.tpr -trr 000003.trr -par tps.par
-ndx index.ndx -maxc 10 -np 16 -mdrun "\$MDRUN"
-gmxrc "/share/apps/gromacs/openmpi-intel/gromacs-4.5.4/bin/GMXRC"
```

The parameter `-v` enables verbose output, which is written to the `0`-file in the output directory. In general this is helpful, as it will give some indication of where things go wrong when a run fails. The MD-settings are then read from a tpr-file, entered under `-tpr`. The initial trajectory is loaded in with `-trr`. The parameter file is loaded in with `-par`, and the groups that are used in the parameter file with `-ndx`. The number of processors for MPI is set with `-np`. The number of shooting moves performed is set with `-maxc`. Finally, the correct GROMACS executables are linked to with `-gmxrx` and `-mdrun`. Additional parameters can be found by running TIPSI with the `-h` command.

Making a .par file

A paremeter file contains several pieces of information. First, the maximum frames of a trajectory is defined. Second, the groups have to be defined, as well as the parameters. For the parameter definitions, a number of functions in TIPSI can be used. Many have a corresponding function native to `gromacs` that provides the same output. A list of all functions, and their parameters, will be provided in this appendix. Third, we have to define the states of the system using these parameters. The states are a boolean expression, written in Python-syntax. An example is:

```
state A = (0.25 < wc & wc < 0.35 & 0.38 < hg)
state B = (0.25 < hg & hg < 0.35 & 0.38 < wc)

interface I = (!A) & (!B)
```

It is noteworthy that TIPSI seems to not handle excessive brackets well, so try to keep your statement within a single pair of brackets, and use the priority order of different operators to your advantage. The intermediate state is always defined as not A and not B. If no states are defined, the order parameters will be calculated for every frame. This can be useful to determine the states, as GROMACS in some cases does not calculate the order parameters in the same way as TIPSI. Finally, the parameter file is finished with a set of recrossing rules. These are generally kept the same.

Defining groups

In general, the way to specify a group of atoms in TIPSI is through a numpy array. For example, we can find the radius of gyration of atom 5, 7, 13 and 18 through:

```
init group = numpy.array((5, 7, 13, 18))
par rg  = rgyration(frame[acc,:])
```

Groups can be specified in an index file, which has to be included as a parameter. A standard index file, as generated by `gromacs`, contains a number of standard groups. These include the entire protein, the carbon backbone, and the sidechains. For example, we can find the radius of gyration over the entire protein:

```
par rg  = rgyration(frame$Protein)
```

TIPSI is often flexible when it comes to defining pairs. For example, the calculater function that calculates the periodic distance between two atoms can take both two seperate atoms as an input, as well as on numpy array of two atoms. This is demonstrated below:

```
par dist1 = pdist(frame[5,:],frame[7,:])

init pair = numpy.array((5, 7))
par dist2 = pdist(frame[pair,:])
```

IMPORTANT NOTE: TIPSI starts its index with 0, whereas GROMACS starts with 1. To avoid off-by-one errors, make sure you subtract 1 off every atom number when you write them directly into the parameter file. The index files read into TIPSI are 1-indexed, however, so `ndx` files do not have to be altered.

Calculator options

Below, we show a list of the available functions to define parameters in the .par file. An atom or group of atoms is always defined within a frame. Each frame is a $n \times 3$ array, containing the x/y/z-coordinates of all n atoms in the trajectory. Some functions can take weights as an input argument. For example, the atom mass can be use as a weight in many cases. The weights can be extracted from, for example, the `.itp`-files in the example. The weights can be loaded into `tipsi` by including them as a numpy-array in the parameter file. If a periodic version exists, `box` as an argument. Non-periodic conditions are assumened if `box` is left as `none`. Alternatively, the list can be included in the index-file, as groups in the index files are simply loaded as arrays. A final important note that for distances, it is preferable to input two of the same atoms at once, so `frame[numpy.array((1,1)),:]`, and adding `.min()` after the command. This is because some distance functions need an array input for the atom numbers, regardless of whether it is only one atom. For more information, explore `functions.py` in the `./bin/tipsi/calculator` folder.

- `com(coord, box, weights)`
  Center of Mass: mass of a set of atom numbers `coord` (numpy array or index group).
- `dist(a, b, box, what)`
  Distance: the distance between atom numbers `a` and `b`. The mode `what` can be set to Euclidean or Manhattan.
- `angle(a, b, c)`
  Angle: the angle between atom numbers `a`, `b` and `c`.
- `dihrad(a, b, c, d)`
  Dihedral (radians): the dihedral between atom numbers `a`, `b`, `c` and `d` in radians.
- `dihdeg(a, b, c, d)`
  Dihedral (degrees): the dihedral between atom numbers `a`, `b`, `c` and `d` in degrees.
- `rgyr(x)`
  Radius of gyration: the radius of gyration over a set of atom numbers `x`. It should be noted that, unlike in GROMACS, this is not standard weighted with the atom mass.
- `rmsd(x, y)`
  Root-mean-square-deviation: the RMSD between the groups of atom numbers `x` and `y`.
- `hbonds(D, H, A, mode, box)`
  Hydrogen bonds: returns the number of hydrogen bonds between a group of donors (an array of atom numbers `D`), hydrogens (an array of atom numbers
  texttttH) and acceptors (an array of atom numbers `A`). The `mode` can be set to `dha`, in which case only triplets are considered in the order given. If not, then all possible combinations of donors, hydrogens and acceptors are considered.