Transition Path Sampling

Introduction

Consider a 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) techniques 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 timescale of milliseconds, for instance. 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 quickly after. The input trajectory can be produced using a high temperature MD simulation, or a metadynamics simulation. TPS selects a point in this trajectory, called the 'shooting point'. From this point, then 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. If the simulation reaches the state (A or B) that was reached at the corresponding end of the original trajectory (the front for a backward shooting move, the end for a forward shooting move), the shooting move is labelled as successful. The idea is that 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. After several shooting moves, new paths can be constructed that go from A to B, and are decorrelated from the original trajectory.

In this tutorial the useage of the TIPSI package (version 0.1) is demonstrated. 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 dipeptide alanine, and samples a transition between its two conformations. The second example concerns a transition which is actually a rare event: the flipping of DNA basepairs from the standard Watson-Crick pairing to the non-standard Hoogsteen pairing (Fig. **??**). This example uses a provided result of a metadynamics simulation as input.

TIPSI relies on GROMACS version 4.5.4 to do molecular simulation. 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. 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. Periodic dodecahedron box with 1nm distance between the wall and the peptide. Uses 100mM NaCl solution. We do an em run, add salt, do an em again, then a posre run. 50000/0.002 steps of dt 0.002.

Dipeptide Alanine

(Short introduction and image.)

Preparatory Analysis

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 MD simulation of the molecular system first.

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://dgoldsb@bitbucket.org/dgoldsb/tipsi-tutorial.git
```

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

```
bin documents dipeptala-inputdnabr-equilibratedpath dnabr-input output 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 tutorial are included in `dipeptala-input`, and output will be written into `output`. The `documents` directory contains supporting reading material.

We enter the directory `dipeptala-input` and explore, which shows the following 8 files:

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

The `.gro` file contains the structure of the molecule we simulate, in our case dipeptide analine. 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.twp
```

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 posre -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 showdipept.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 `showdipept.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. We will also use this trajectory to define the stable states.

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

TIPSI also requires a `.tpr` file in which the MD simulation conditions that you are interested in are defined. 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` as follows:

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

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

LAGE TEMP TOCH WEL, MAAR KORT, 10 nanosec, is het echt stabiel?

Running TIPSI

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 ome collective variable. In this example, we use the radius of gyration and the dihedral angle over atoms (5, 7, 9, 15).

NOTE: applied -1 here to run in TIPSI

[Protein] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

[Dihedral1] 4 6 8 14

[Dihedral2] 16 14 8 6

First, to analyze our initial run, go to the output folder of the high temperature run. Now, run

```
g_gyrate -f highT.xtc -s highT.tpr -o gyr_highT.xvg
```

to generate a dataset containing the radius of gyration for each frame. Then, define all dihedral angles by running

```
mk_angndx -s highT.tpr -n angle.ndx -type dihedral
```

followed by

```
g_angle -f highT.xtc -od angles.ndx -ov dihed_highT.xvg
-type dihedral
```

with option 8 (5, 7, 9, 15) 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 gyr_highT.xvg 1 dihed_highT.xvg
1 0.2 0.3 -180 -40 100 100 Gyration Dihedral 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 a radius of gyration and dihedral angle that are visited a lot by the simulation. If we repeat this with the lowT results, we see that there are few transitions between the states. 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.

Go to the `./bin` directory, and look at the `tps.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. Make sure to fill in your own directory in the parameter file for the location of the `.gro` file, as we enter the full path here, not the relative path.

DO WE STILL?

We included a group of atoms in out parameter file, namely `protein`. We must define this custom group in an `.ndx` file to pass this group to TIPSI. Go to the input directory, and run

```
make_ndx -f lowT.tpr
```

If you open this file with `more`, you will see that groups are defined in blocks, and that the name of the block is specified by a header in square brackets. The produced file allows you to use any of the GROMACS standard groups. Custom groups can be made manually, and are referred to by their header. The index file of choice is included in the `run_ tipsi` file. We edit this file slightly to contain just the groups that we are interested in: the protein group, and the group for the dihedral angle. Open `index.ndx`, and delete the blocks in index.ndx that are not the protein group, and rename the file `index-pr-dh.ndx`. Then, add

```
[ Dihedral ]
   5    7    9   15
```

and save the file in the input directory.

ONLY THE DIHEDRALS, DONT INCLUDE THE FILE IN THE REPO

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
```

BRIEFLY STATE WHAT IS WHAT

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`, 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

The

```
Properties are in extra file

If you feel brave, you can check if there were no warnings or errors by opening mer
```

```
% TODO: Treeplot
% TODO: Analyze if they are decorrelated
% TODO: Calculate average pathlength

% TODO: MAYBE OPS

\newpage

\subsection*{DNA Baseroll}

(Short introduction with picture)

\subsubsection*{Preparatory Analysis}

We again log into the Carbon cluster. Make sure you are connected to the UvA networ
%
\begin{lstlisting}
ssh user@carbon.science.uva.nl
```

SAY HOW TO DO REGULAR MD, BUT ALL WE NEED IS TPR We enter the directory `input` and explore, which shows the following 8 files:

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

The `.gro` file contains the structure of the molecule we simulate, in our case dipeptide analine. 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.

Already added solvent, did an energy minimaztion and contrained run. Now that we have relaxed the solvent we can run the MD simulation. Typically when preparing for TIPSI we run at two temperatures, room temperature (298K) and a higher temperature (400K). Here we do room only, otherwise denatures completely. The simulation at room temperature shows the stable state at the temperature we are interested in. At the higher temperature, the system will transition more easily to the second stable state. This simulation will provide the initial transition trajectory, and also helps us define the two stable states. We run these by entering:

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

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

Running TIPSI

* glycosidic angle: 483, 484, 486, 499 * bp (waterstofbrug die altijd gevormd wordt bij baseparing): 275, 492 * wc (waterstofbrug die gevormd wordt bij WC paring): 276, 495 * hg (waterstofbrug die gevormd wordt bij Hoogsteen paring): 276, 489

STATES ARE ALREADY DEFINED, GIVE THE 3 CONDITIONS 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 ome collective variable. In this example, we use the radius of gyration and the dihedral angle over atoms (5, 7, 9, 15).

First, to analyze our initial run, go to the output folder of the high temperature run. Now, run

```
g_gyrate -f highT.xtc -s highT.tpr -o gyr_highT.xvg
```

to generate a dataset containing the radius of gyration for each frame. Then, define all dihedral angles by running

```
mk_angndx -s highT.tpr -n angle.ndx -type dihedral
```

followed by

```
g_angle -f highT.xtc -od angles.ndx -ov dihed_highT.xvg
-type dihedral
```

with option 8 (5, 7, 9, 15) 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 gyr_highT.xvg 1 dihed_highT.xvg
1 0.2 0.3 -180 -40 100 100 Gyration Dihedral GyrDihed
```

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 a radius of gyration and dihedral angle that are visited a lot by the simulation. If we repeat this with the lowT results, we see that there are few transitions between the states. 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.

Go to the `./bin` directory, and look at the `tps.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. Make sure to fill in your own directory in the parameter file for the location of the `.gro` file, as we enter the full path here, not the relative path.

We included a group of atoms in out parameter file, namely `protein`. We must define this custom group in an `.ndx` file to pass this group to TIPSI. Go to the input directory, and run

```
make_ndx -f lowT.tpr
```

If you open this file with `more`, you will see that groups are defined in blocks, and that the name of the block is specified by a header in square brackets. The produced file allows you to use any of the GROMACS standard groups. Custom groups can be made manually, and are referred to by their header. The index file of choice is included in the `run_ tipsi` file. We edit this file slightly to contain just the groups that we are interested in: the protein group, and the group for the dihedral angle. Open `index.ndx`, and delete the blocks in index.ndx that are not the protein group, and rename the file `index-pr-dh.ndx`. Then, add

```
[ Dihedral ]
   5    7    9    15
```

and save the file in the input directory.

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`, 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

resname resid 9DT, 4DA

```
# Dirty index file written in vim, should suffice
[bp]
275 492
[wc]
```

```
276 495
[hg]
276 289
[glycosydic angle]
483 484 486 499
[bp1]
275
[bp2]
492
[wc1]
276
[wc2]
495
[hg1]
276
[hg2]
489
```

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 1
```

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`. 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. 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,:])
```

Calculator options

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. The index files read into TIPSI are 1-indexed, however.
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.