Chapter 1

TPS using Open Path Sampling

Thursday 27-06-2024

1.1 Introduction

This chapter will introduce you to OpenPathSampling (OPS), as well as several other libraries for (bio)molecular simulation and for scientific computing in general. You will perform path sampling, analyze a complete transition path sampling run, and analyze a committor simulation. You'll also get familiar with using a Jupyter notebook to interactively run simulation and analysis. You can run this tutorial on your own computer, and requires installing a few things. You can only run it on your own computer using Linux, Virtual Machine Linux, macOS, or the Linux Subsystem for Windows. Note that OpenPathSampling does **not** support Windows straightforwardly; if you're using a Windows machine, you should use the Windows Subsystem for Linux or use the Virtual Machine Linux.

1.2 Installation

The easiest way to install OPS is with conda. Conda is a powerful package and environment management system. If you do not already have a highly customized Python environment, we recommend starting by installing conda, either in the full anaconda distribution or the smaller-footprint miniconda (`https://docs.anaconda.com/miniconda/`). Note: OPS does not work in Windows, install in Windows Subsystem for Linux or the Virtual Machine Linux instead.

Once you have installed conda and it is in your path, download the 'ops.yml' file required to install ops.

Use the link at: `https://github.com/CECAM-LTS-MAP/OPS-tutorial/blob/main/ops.yml`

create a new environment:

```
conda env create -f ops.yml --name ops_env
conda activate ops_env
```

Installation instructions are also available online. Note that specific versions of python and libraries are required for OPS, which are included in the `ops.yml` file.

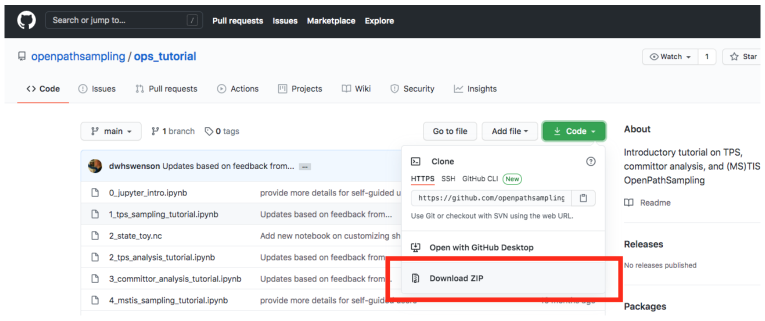
`https://github.com/dwhswenson/ops_tutorial/blob/main/README.md`

1.3 Obtaining the tutorial materials

Go to

`https://github.com/openpathsampling/ops_tutorial`

Download the files and extract them into a local directory:



dwhswenson Updates based on feedback from...

```
0_jupyter_intro.ipynb                provide more details for self-guided u
1_tps_sampling_tutorial.ipynb        Updates based on feedback from...
2_state_toy.nc                       Add new notebook on customizing sh
2_tps_analysis_tutorial.ipynb        Updates based on feedback from
3_committor_analysis_tutorial.ipynb  Updates based on feedback from
4_mstis_sampling_tutorial.ipynb      provide more details for self-guid
```

`https://github.com/openpathsampling`

Use Git or checkout with SVN using the web URL.

Introductory tutorial on TPS, committor analysis, and (MS)TIS OpenPathSampling

No releases published

Make sure to download all files in the tutorial repository.

You will also want to download the precomputed results (for analysis), which can be obtained from

`https://figshare.com/s/01302bc7a39ec7648ea1`

Extract them and save them in the same directory as the Jupyter notebooks.

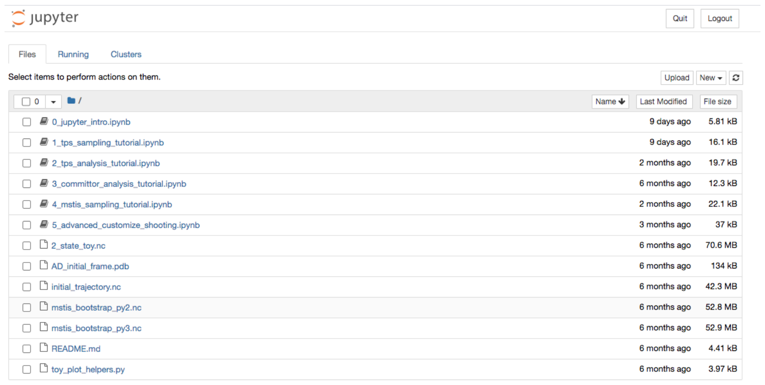
1.4 Launching the Jupyter notebook

Jupyter notebooks are a very convenient way to work with code like OPS. If you've used Mathematica or similar software, they take the same philosophy. You have "cells" where you can input code, and you run each of those cells. You can also have special cells to add text, so you can document what the notebook does. The big difference between Mathematica and Jupyter is that, whereas Mathematica only works with its own language, Jupyter works with many different languages. In this tutorial, we'll be using Python – everything you do in this is technically Python programming, but the libraries we use make it much more straightforward.

Open a terminal. Change to the directory with the ops_tutorial materials. Type

```
jupyter notebook
```

This will usually open a web browser. If it doesn't, it will give you a link to copy-paste into a web browser. In the browser, you should see something similar to this:



Select items to perform actions on them.

Name	Last Modified	File size
0_jupyter_intro.ipynb	9 days ago	5.81 kB
1_tps_sampling_tutorial.ipynb	9 days ago	16.1 kB
2_tps_analysis_tutorial.ipynb	2 months ago	19.7 kB
3_committor_analysis_tutorial.ipynb	6 months ago	12.3 kB
4_mstis_sampling_tutorial.ipynb	2 months ago	22.1 kB
5_advanced_customize_shooting.ipynb	3 months ago	37 kB
2_state_toy.nc	6 months ago	70.6 MB
AD_initial_frame.pdb	6 months ago	134 kB
initial_trajectory.nc	6 months ago	42.3 MB
mstis_bootstrap_py2.nc	6 months ago	52.8 MB
mstis_bootstrap_py3.nc	6 months ago	52.9 MB
README.md	6 months ago	4.41 kB
toy_plot_helpers.py	6 months ago	3.97 kB

You can click on a notebook name to open it in a new tab. Once in a notebook, you can run each cell by typing shift-return. The first tutorial notebook (number 0) gives more information on using Jupyter notebooks.

1.5 The tutorial

The tutorial is in four parts. The first part introduces you to Jupyter notebooks. The second part shows you how to do transition path sampling in OpenPathSampling, and the third part shows you how to analyze a much longer sampling run. In the fourth and final part, you'll analyze the results of a committor simulation. There are extra notebooks at the end, to do multiple state transition path sampling, customize shooting moves and to do transition interface sampling. These are not required to do now, and have no questions in them. They may be interesting for people who want to learn more about path sampling methods. In all these notebooks, most of the steps that are expected to take a long time (potentially longer than a minute) are marked with %%time at the top of the cell. This is an Jupyter/IPython feature that automatically checks how long the cell took to run (but it is used here as a reminder that this cell is expected to take a while to run). The only cell that takes a long time and isn't marked with that is the `sampler.run(40)` line in the sampling tutorial. That gives a time- remaining estimate as soon as it is running.

Start with the **0_jupyter_intro.ipynb** tutorial to get familiar with how to use a Jupyter notebook. This shouldn't take you more than a few minutes. Then start on the sampling tutorial **1_tps_sampling_tutorial.ipynb.** Once you have the notebook up, it should be pretty straightforward. There are many links within this text; these are for more information if you are interested. You'll need to add some lines, marked with the text `YOUR TURN`. In particular, you'll need to: define the collective variable psi, much like the collective variable phi define the volume for the state alpha_R , much like the volume for the state C_7eq Once you're performing the sampling (running the simulation), you can load the file in the analysis tutorial in **2_tps_analysis_tutorial.ipynb.** Again, for the most part you can just step through part of the analysis tutorial. There are a few tasks marked as `YOUR TURN`, including one advanced task (optional, details on this were not covered in this course). Be sure to answer the questions that are asked. That will also take some time, so doing both at once will help you finish the whole tutorial more quickly.

The line:

```
path_dens_counter = path_density.histogram([s.active[0].trajectory for s in flexible.steps])
```

will take a bit of time to run. This is a good time to take a brief break.

Once that histogram has been generated, the rest is straightforward. You'll need to identify trajectories trajA and trajB which belong to each transition channel. After the nglview widgets have loaded, you can rotate them by dragging, pan by dragging with a right click, or zoom with a scroll. Double-clicking will make the 3D view go full screen (hit esc to exit).

The last cells in the notebook show how to visualize trajectories from the precomputed data. In order to save disk space, those details have been stripped from the data you have, so you won't be able to run those cells.

Finally, you'll analyze a committor simulation in **3_committor_analysis_tutorial.ipynb.** The file you analyze is based on taking 1000 snapshots from a 10000 move TPS simulation, and running each snapshot 10 times. One of the advantages of OPS simulations is that the shot stops as soon as it enters a state. This makes it much more efficient, especially for short paths.

The main analysis, with the line

```
analyzer = paths.ShootingPointAnalysis(steps=simulation_storage.steps,
             states=[C_7eq, alpha_R])
```

will take a bit longer.

In this notebook, you'll need to create a histogram for the psi variable, which is analogous to the histogram for the phi variable that is demonstrated in the notebook. Don't forget to create the hash function, which is made in an earlier cell for phi ! You'll also try to define the transition state, based on the 50% committor frames.

The tutorial contains more notebooks, illustrating examples of path sampling applications. Of particular interest is **4_mstis_sampling_tutorial.ipynb**, which shows how to set up and run multiple state transition interface sampling for a simple toy model.