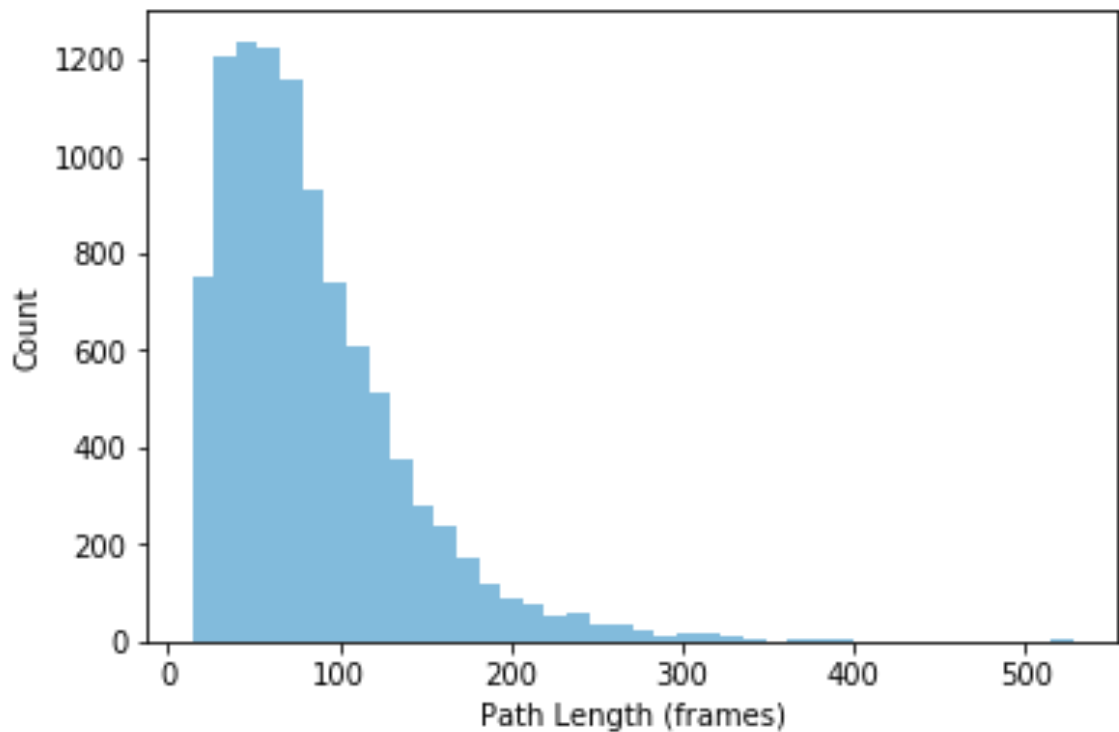
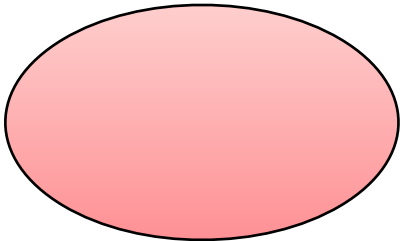


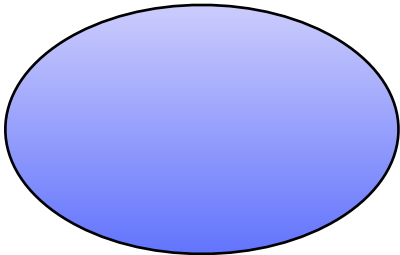
- Advantage: Much shorter trajectories
- Disadvantage: Need to monitor the MD while running

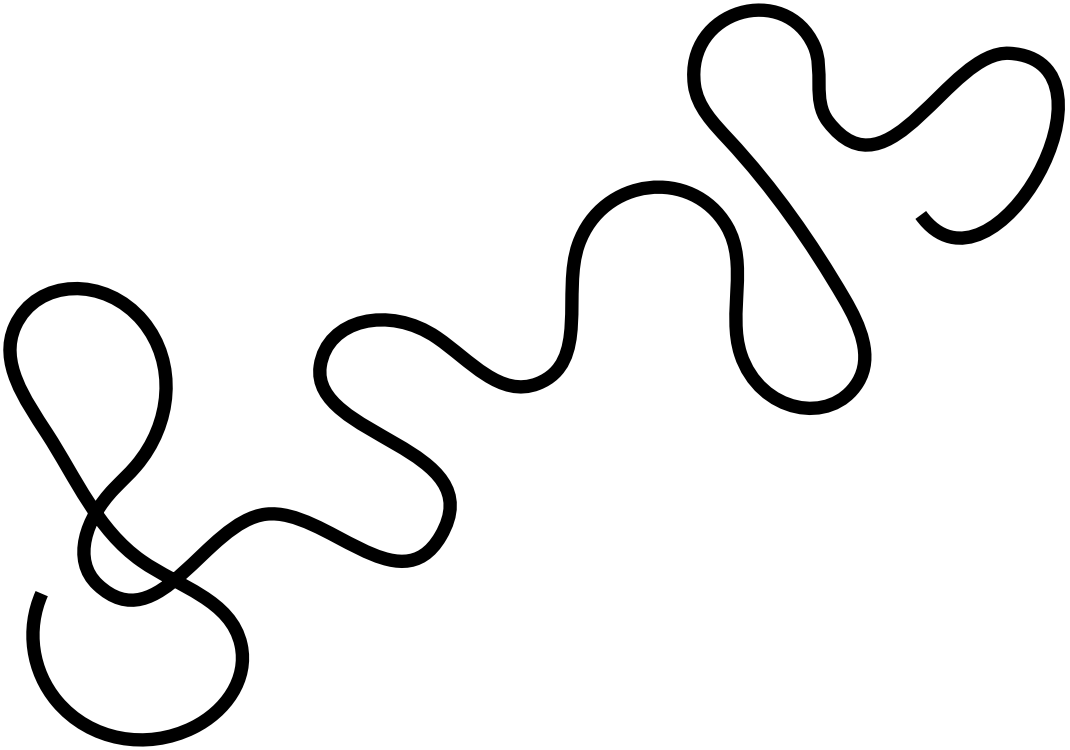
Flexible Length Shooting

Assume a working states: Trajectories stop when they enter.





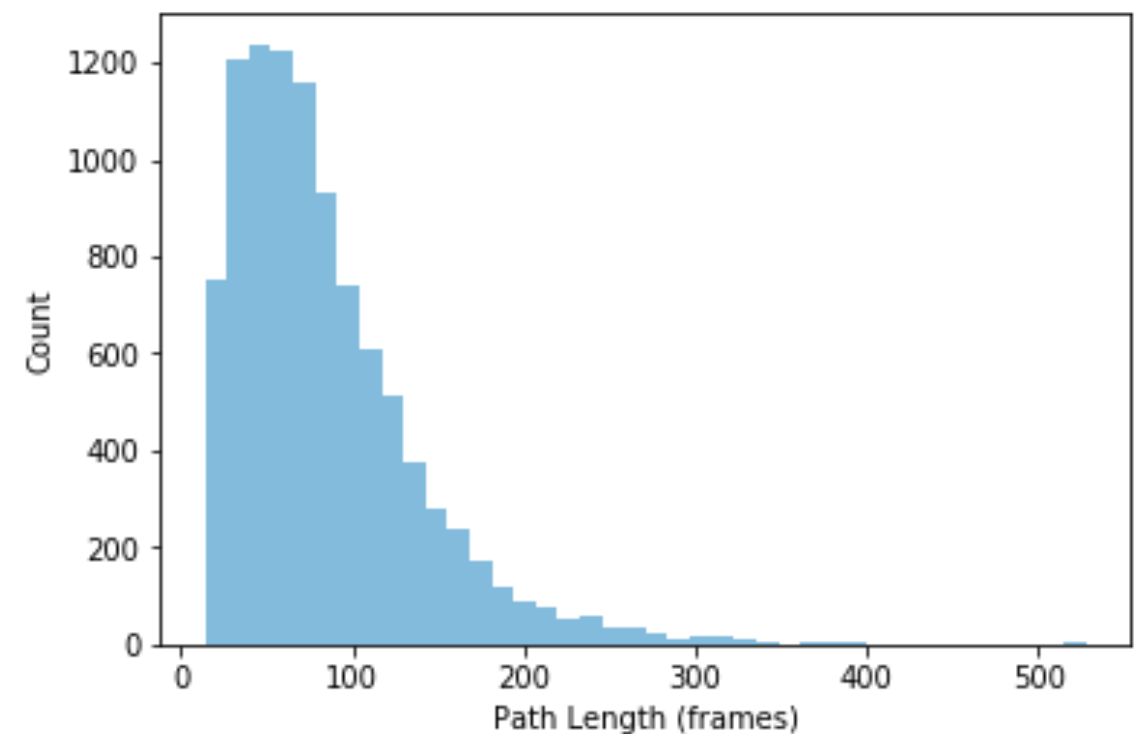
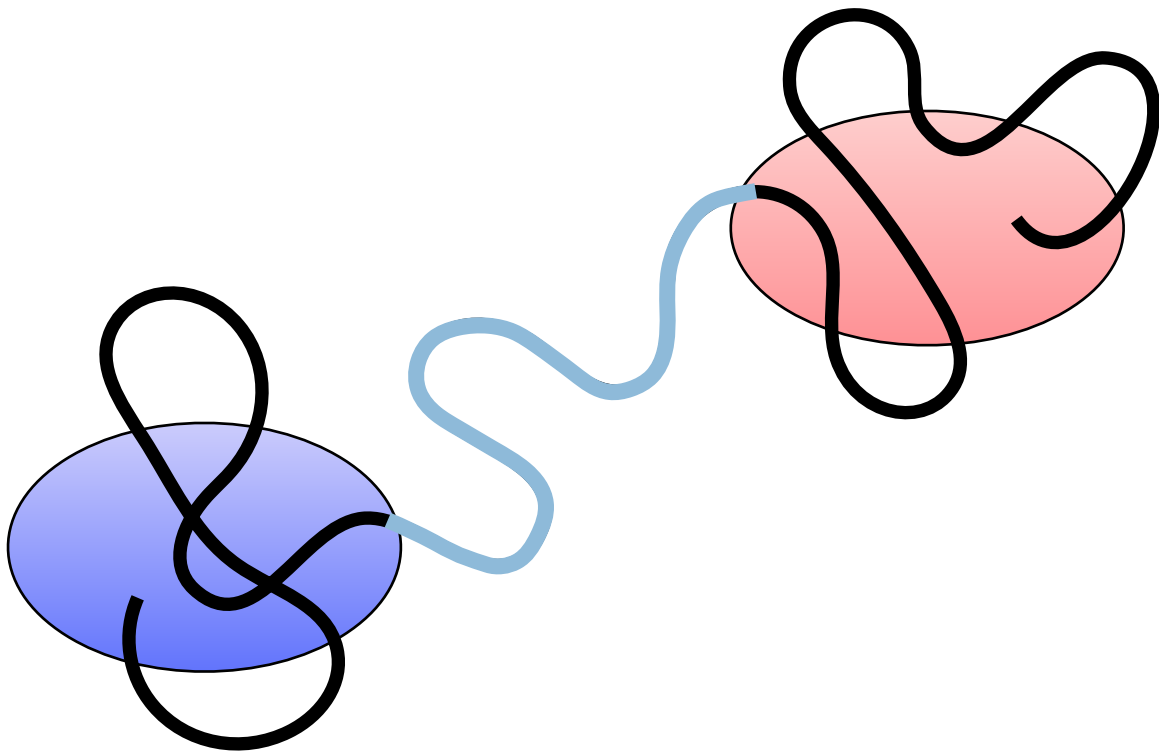






Flexible Length Shooting

Assume absorbing states: Trajectories stop when they enter.



- Advantage: Much shorter trajectories
- Disadvantage: Need to monitor the MD while running

OpenPathSampling



<http://openpathsampling.org>

Twitter: @pathsampling

Development at: <http://github.com/openpathsampling/>

Part of the Omnia consortium: <http://omnia.md>

A Python library for path sampling simulations

- ✓ **Easy to use:** Beginners can quickly learn to use it
- ✓ **Easy to extend:** Advanced users can use it to develop new methods
- ✓ **Independent of dynamics engine:** Useful in many fields and to the broadest audience

```
import openpathsampling as paths
in_file = paths.AnalysisStorage("input_file.nc")
init_traj = in_file.trajectories[0]
engine = in_file.engines[0]
dist = in_file.cvs['my_distance']
stateA = paths.CVRangeVolume(dist, 0.0, 1.0)
stateB = paths.CVRangeVolume(dist, 3.0, float('inf'))
ensemble = paths.TPSEnsemble(stateA, stateB)
shooting_mover = paths.OneWayShootingMover(ensemble)
init_samp = paths.Sample(
    replica=0,
    trajectory=ensemble.split(init_traj)[0],
    ensemble=ensemble)
out_file = paths.Storage("output.nc", "w", init_traj[0])
tps_calc = paths.PathSampling(
    storage=out_file,
    engine=engine,
    move_scheme=paths.LockedMoveScheme(shooting_mover),
    globalstate=paths.SampleSet([init_samp]))
tps_calc.run(1000)
```

Prinz, DWHS, Bolhuis, Chodera. In prep.