WXML Final Report: Conditional Path Sampling

Faculty mentor: Panos Stinis
Graduate mentor: Jacob Price
Undergraduates: Jesse Rivera and Landon Shorack

Autumn 2017

1 Introduction

Suppose we are given a system and its associated *free energy surface* $U : \mathbb{R}^n \to \mathbb{R}$, which describes the free energy of the system as a function of a certain set of n features. These features could include anything that does a good job of describing the overall state of the system, e.g. bond angles, dipole moments, or positions of particles. Suppose we keep track of these features in a *state vector* $\mathbf{x}_t \in \mathbb{R}^n$ which describes the state of the system at time t. Without loss of generality we may assume $t \in [0, 1]$, and for computational purposes we discretize time into equal intervals of length $\Delta t \ll 1$. We then define a *path* (of resolution $m = \frac{1}{\Delta t} + 1$) to be an m-tuple of state vectors $\vec{\mathbf{x}} = (\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)$ so that each path represents a possible evolution of the system over time.

We are interested in systems with multiple metastable states. A *metastable state* is loosely defined as the preimage of a local minimum of U and the surrounding region. We then define a *transition path* to be any path whose first and last points are in different metastable states.

Some systems of interest are difficult to simulate because their metastable states are separated by high energy barriers. An example of this can be seen in protein folding systems. Standard molecular dynamics simulations require one to use time steps on the order of 10^{-15} seconds, however, a protein may take as long as 10^{-4} seconds to fold. Consequently, simulating even a single protein folding event may take years of dedicated computation time on high end machines. If our goal is to better understand transitions such as these (i.e. transitions between metastable states), standard simulations are clearly not a feasible option. We instead construct a conditional probability distribution of transition paths from which to sample. Doing so allows us to understand the statistics of such transitions in a more computationally efficient way.

The systems we are interested in are subject to some degree of random noise (which, for example, could be caused by collisions with particles in a solvent). Let $\mathbf{X}_t$ be a random variable corresponding to a state vector and $\vec{\mathbf{X}} = (\mathbf{X}_0, \mathbf{X}_{\Delta t}, \ldots, \mathbf{X}_1)$ a random variable representing a path. Under Langevin dynamics, the system changes according the stochastic differential equation

$$d\mathbf{X}_t = -\nabla U(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t \tag{1}$$

where $\sigma(\mathbf{X}_t)$ is the amplitude of noise (as a function of the current state) and $d\mathbf{W}_t$ is standard Brownian motion.

In order to include only transition paths in our distribution, we specify start and end distributions for our system. Let $g_{\mathbf{X}_0}(\mathbf{x}_0)$ be a distribution of state vectors centered in one metastable state and $h_{\mathbf{X}_1}(\mathbf{x}_1)$ a distribution of state vectors centered in a different metastable state. Then the probability density function for the distribution of transition paths between these two metastable states is given by

$$p_{\vec{\mathbf{X}}}(\vec{\mathbf{x}}) = g_{\mathbf{X}_0}(\mathbf{x}_0)\left[\prod_{i=1}^{m-2} \Pr\left(\mathbf{X}_{i\Delta t} = \mathbf{x}_{i\Delta t} \mid \mathbf{X}_{(i-1)\Delta t} = \mathbf{x}_{(i-1)\Delta t}\right)\right] h_{\mathbf{X}_1}(\mathbf{x}_1) \tag{2}$$

The conditional probabilities of the intermediate state vectors are determined by the stochastic differential equation 1, which we discretize according to the Euler-Maruyama scheme:

$$\mathbf{X}_{(i+1)\Delta t} = \mathbf{X}_{i\Delta t} - \nabla U(\mathbf{X}_{i\Delta t})\Delta t + \sigma(\mathbf{X}_{i\Delta t})\sqrt{\Delta t}\cdot \xi_{i\Delta t} \tag{3}$$

Here $\xi_{i\Delta t}$ is a sample from the standard normal distribution $\mathcal{N}(0,1)$ so that we may think of the state $\mathbf{X}_{(i+1)\Delta t}$ as being a sample from the normal distribution with mean $\mathbf{X}_{i\Delta t} - \nabla U(\mathbf{X}_{i\Delta t})\Delta t$ and standard deviation $\sigma(\mathbf{X}_{i\Delta t})\sqrt{\Delta t}$. It follows that for each $i \in \{1,\ldots,m-2\}$,

$$\Pr\left(\mathbf{X}_{i\Delta t} = \mathbf{x}_{i\Delta t} \mid \mathbf{X}_{(i-1)\Delta t} = \mathbf{x}_{(i-1)\Delta t}\right) = \exp\left\{-\sum_{j=1}^{n} \frac{(x^j_{i\Delta t} - (x^j_{(i-1)\Delta t} - \nabla U^j(x_{(i-1)\Delta t})\Delta t))^2}{2(\sigma^j(x_{(i-1)\Delta t}))^2\Delta t}\right\} \tag{4}$$

where the superscript j denotes the j^{th} component of the specified state vector or function. Plugging 4 into 2 and applying the properties of the logarithm we obtain the following density function:

$$p_{\vec{\mathbf{X}}}(\vec{\mathbf{x}}) = \exp\left\{\log\left(g_{\mathbf{X}_0}(\mathbf{x}_0)\right) - \left(\sum_{j=1}^{n}\sum_{i=1}^{m-2} \frac{(x^j_{i\Delta t} - (x^j_{(i-1)\Delta t} - \nabla U^j(x_{(i-1)\Delta t})\Delta t))^2}{2(\sigma^j(x_{(i-1)\Delta t}))^2\Delta t}\right) + \log\left(h_{\mathbf{X}_1}(\mathbf{x}_1)\right)\right\} \tag{5}$$

We use the Hamiltonian Monte Carlo (HMC) algorithm to sample from this distribution. This amounts to constructing a Hamiltoninan system with stationary distribution $p_{\vec{\mathbf{X}}}(\vec{\mathbf{x}})$. We treat the path $\vec{\mathbf{x}}$ as a position vector $\mathbf{q} \in \mathbb{R}^{mn}$ (we do this by putting each component of each state vector in $\vec{\mathbf{x}}$ into one long vector) and introduce a fictitious velocity vector $\mathbf{p} \in \mathbb{R}^{mn}$ drawn from the standard multivariate normal distribution. We define the Hamiltonian of the system to be

$$H(\mathbf{q},\mathbf{p}) \coloneqq -\log\left(p_{\vec{\mathbf{X}}}(\mathbf{q})\right) + \frac{\mathbf{p}^T\mathbf{p}}{2} \tag{6}$$

According to Hamiltonian dynamics, the system changes according to

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

for each $i \in \{1, \dots, mn\}$. We let this Hamiltonian system evolve for L timesteps of size δt using a symplectic integrator. The result is new position and velocity vectors $\mathbf{q}'$ and $\mathbf{p}'$. The new position vector corresponds to a new path, which we accept with probability

$$\Pr(\text{accept}) \coloneqq \min\{1, \exp[H(\mathbf{q}, \mathbf{p}) - H(\mathbf{q}', \mathbf{p}')]\} \tag{7}$$

Since symplectic integrators conserve the Hamiltonian, the algorithm should accept every path it proposes. In practice, however, floating point arithmetic will result in some paths being rejected. In this event we reiterate the current path by setting $\mathbf{q}' = \mathbf{q}$ and repeating the process. For a more detailed discussion of the HMC algorithm see [3].

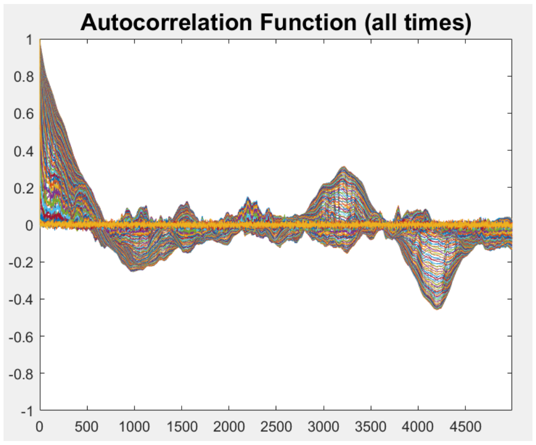
2 Parameter Tuning

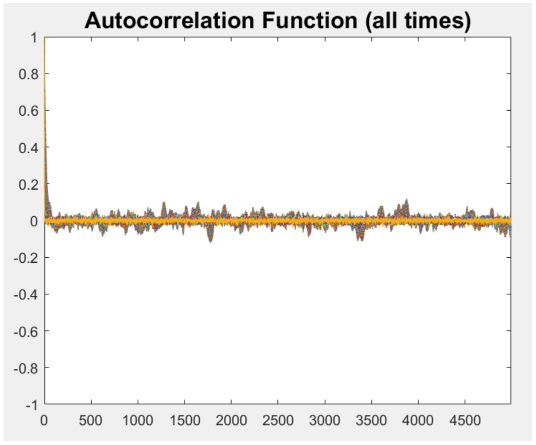
The HMC algorithm is flawed in that the samples it produces are highly correlated. One approach to mitigating this problem is to tune the parameters δt and L. This was the main focus of our work spring quarter.

We want the HMC to explore the state space as quickly as possible while still producing accurate results. We also want the paths produced by the HMC to be uncorrelated. Increasing δt results in faster exploration of the state space, but setting δt too high will result in proposed paths being rejected with a high probability, which in turn causes the simulation to run slowly or get stuck. Increasing L results in faster exploration of the state space and less correlated paths, but at the cost of computation time.

It should be noted that the effects of adjusting δt and L are disjoint, so they may be tuned separately. Furthermore, we can think of $\delta t \cdot L$ as how far the HMC is enabled to deviate from the current path; it can be thought of as a measure of how much risk we allow the HMC to take. We found through experimentation that $\delta t \cdot L = 5$ is close to optimal.

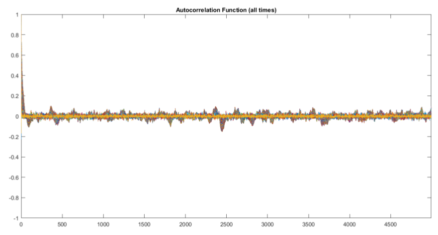
Our first goal was to understand the effect that changing δt and L had on our results. We found that increasing L more rapidly reduced the autocorrelation; however, it also greatly increased the computation time required for the HMC to run.

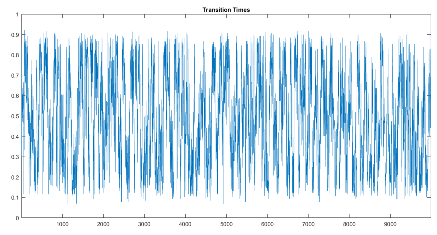




In addition, there is a point at which further increasing L has almost no effect; we have already let the HMC to explore sufficiently far away from the original path, so allowing it to take more steps yields us no further benefit. On the other hand, increasing δt also helped reduce autocorrelation, but the optimal value was far more specifically bounded. If δt was too small we would not be allowing the HMC to explore sufficiently and so the autocorrelation would be too high. But if it was too large, the likelihood that the HMC would return a bad path would increase and the path would be rejected with a higher probability.

It has been proven that the HMC performs optimally when it accepts 65.1% of the paths that it proposes [2]. As a result, we used the acceptance rate as our primary metric of determining the effectiveness of a given set of parameters, δt and L, and we made it our goal to get this ratio as close to 65.1% as possible. Below is the closest we managed to get, with an acceptance rate of 64.84% at a resolution of 2^{-7}.





3 Path Exchange Algorithms

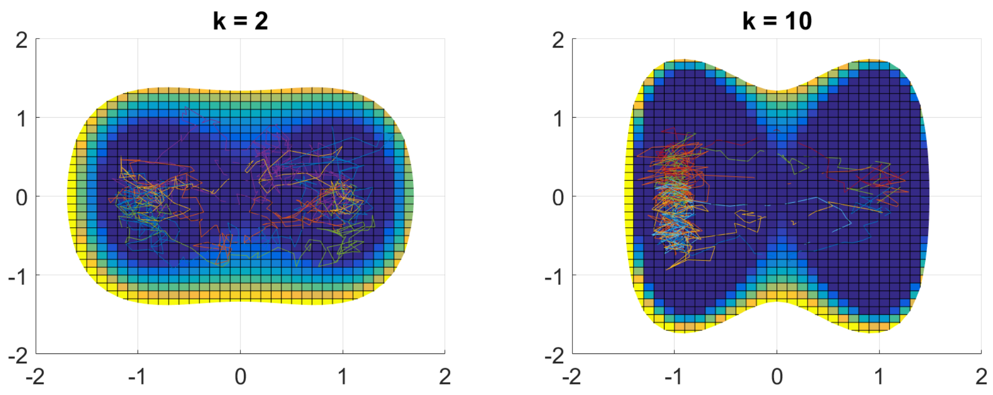
Another way to reduce the correlation of samples produced by the HMC is to implement a path exchange algorithm. This involves running the HMC algorithm simultaneously on multiple copies of the same system. Each copy, however, is slightly different in that it has an "easier" version of the problem. Two separate ways in which we have made the problem easier are by: (1) lowering the resolution of the paths, and (2) decreasing the depth of the wells in the given free energy surface. Then, after every certain number of time steps, we randomly select a pair of adjacent systems. We propose an exchange between the current paths of these systems, defining the success rate of this swap in terms of the ratio between the probabilities of the two paths. If the paths of each system have different resolutions, we interpolate missing points by drawing them from normal distributions and the acceptance ratio becomes slightly more complicated to account for the interpolated points.

These algorithms have been very effective in one-dimensional problems and we hope to see similar results when we implement them with our multidimensional code. See [4] for more details.

4 Increasing Dimensions

Our main task this quarter was increasing the dimensions that our simulations could consider. Primarily this required us to transition derivatives to gradients and second derivatives to hessians. In addition many of our vectors became matrices but the HMC algorithm only takes vectors. As a result we developed methods to un-spool matrices to vectors and vice versa. Finally we had to update our "toy-problem" to now include a two dimensional space to explore. Hence, we had to plot in 3-dimensions as the 3rd dimension was the 'z'-coordinate determined by the potential energy landscape, U. Putting all this together we manged to produce paths, by way of the HMC algorithm, that crossed from one well to the other.

Figure 1: Here k refers to the depth of the well.



In addition to updating our code, this particular project required us to do something of a code review. We were able to better ensure there were no subtle mistakes embedded in the work we had done so far.

5 Future Goals

Our plans for next quarter involve a retuning of the HMC parameters, δt and L. We want to ensure our work from the spring carries over to this work. This means we will need to bring over our methods from the one-dimensional case. These include a test for the correlation between paths and another that tests the time it takes for the HMC to explore the entire state space. Currently our plan for tracking the exploration of the state space is to consider the transition times as before. We will note the time step in which the end of the path becomes closer to the right well than it is to left well. When the HMC is running well we want to see this value vary widely. Further, we would like to implement the path sampling and time exchange methods described earlier. This will again require a through review of the code of these algorithms. We will need to transition derivatives to gradients again as well as manipulating matrices to be vectors, as before. We have also discussed adding new problems to explore. One may be the three body problem, a exploration of how a the motion of a system of three planetary bodies evolves with time. Allow this is not related to protein-folding per-say our methods may still provide insights. This may help us to test our model on a problem of more dimensions.

References

[1] A. Barducci et al. Metadynamics. *WIREs Computational Molecular Science*, 1:826–843, 2011.

[2] A. Beskos et al. Optimal Tuning of the Hybrid Monte-Carlo Algorithm. 2010.

[3] R. M. Neal et al. MCMC using Hamiltonian Dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

[4] P. Stinis. Conditional Path Sampling for Stochastic Differential Equations through Drift Relaxation. *Communications in Applied Mathematics and Computational Science*, 6(1):63–78, 2011.