WXML Final Report: Conditional Path Sampling

Panos Stinis, Jacob Price, Jesse Rivera and Landon Shorack

Spring 2017

This project involves running a simulation to find transition paths between metastable states of a stochastic system. Events such as protein folding occur so rarely that running a standard simulation would take hundreds of years to finish. We want to be able to simulate such events quickly and efficiently, as this data would allow us to better understand the statistics of such transitions. This is one of the long term goals of our project.

Our simulation is based on the following model: we represent transition paths as a conditional probability distribution and then sample from this distribution using the Hamiltonian Monte Carlo algorithm (also called the Hybrid Monte Carlo, and often abbreviated HMC).

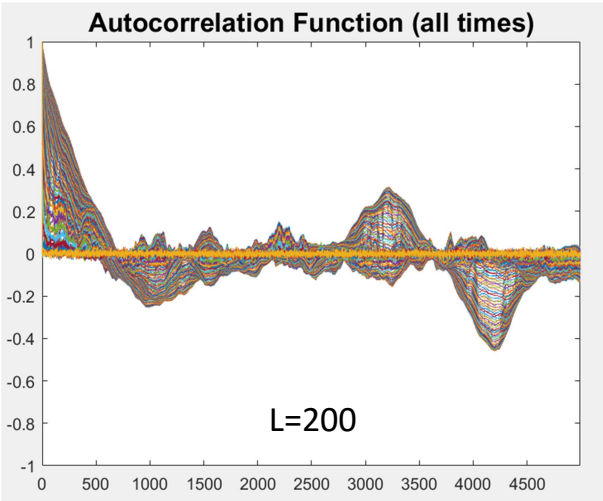
The HMC takes in the current path and outputs a new one. The input path (a vector of points along a given free energy landscape) is used to represent a velocity vector, and we introduce a fictitious, randomly generated momentum vector to go along with it. The system then evolves according to Hamilton's equations, and the result is a new position vector that corresponds to a new path. This proposed path is accepted with a probability that depends on the difference between the Hamiltonian of the current path and the Hamiltonian of the proposed path. Note that the Hamiltonian is a conserved quantity, but floating point arithmetic can result in slight variation. More precisely, the higher the error in the floating point arithmetic, the less likely the HMC will accept the proposed path. If a proposed path is rejected, the simulation reiterates the current path and tries again. Because of this, it is possible for the simulation to "get stuck" on a path if the acceptance probability is too low.

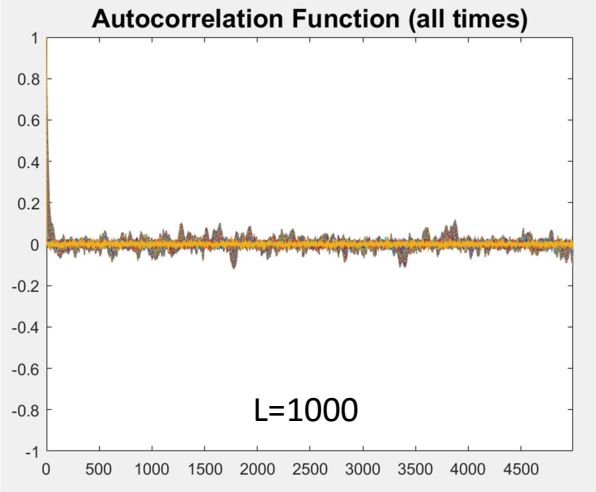
We are using the leapfrog method as our numerical integration scheme. Note that this is a symplectic integrator, so the Hamiltonian should be conserved. This requires the use of two arbitrary parameters, δt and L, which correspond to the step size of the leapfrog algorithm and the number of steps taken before a new path is proposed. It turns out that the choice of these two parameters, δt especially, greatly affects how well the HMC algorithm performs.

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.

Most of our work this quarter revolved around tuning these two parameters, δt and L. 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 $\delta t * L$ 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 $\delta t * L = 5$ is close to optimal.

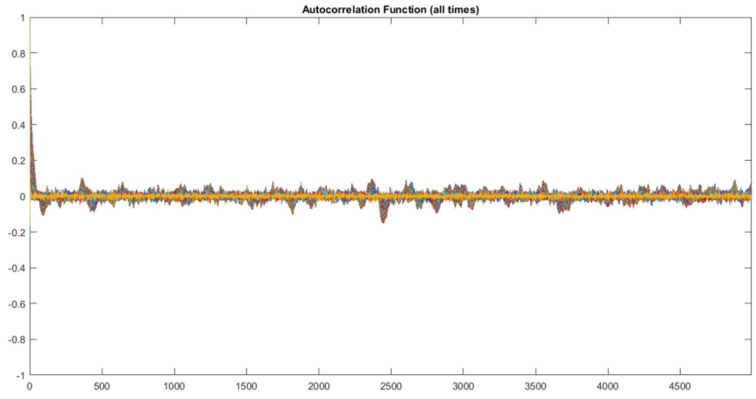
Our first goal was to understand the effect that changing δt and L had on our results. We found that increasing L more rapidly reduced the autocorrelation; however, it also greatly increased the computation time required for the HMC to run.

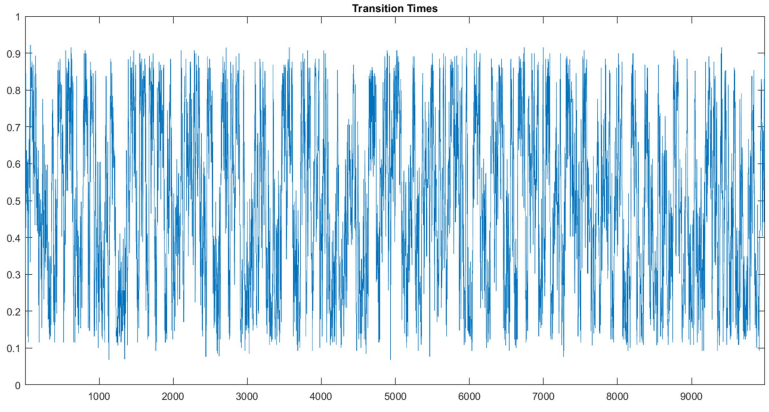




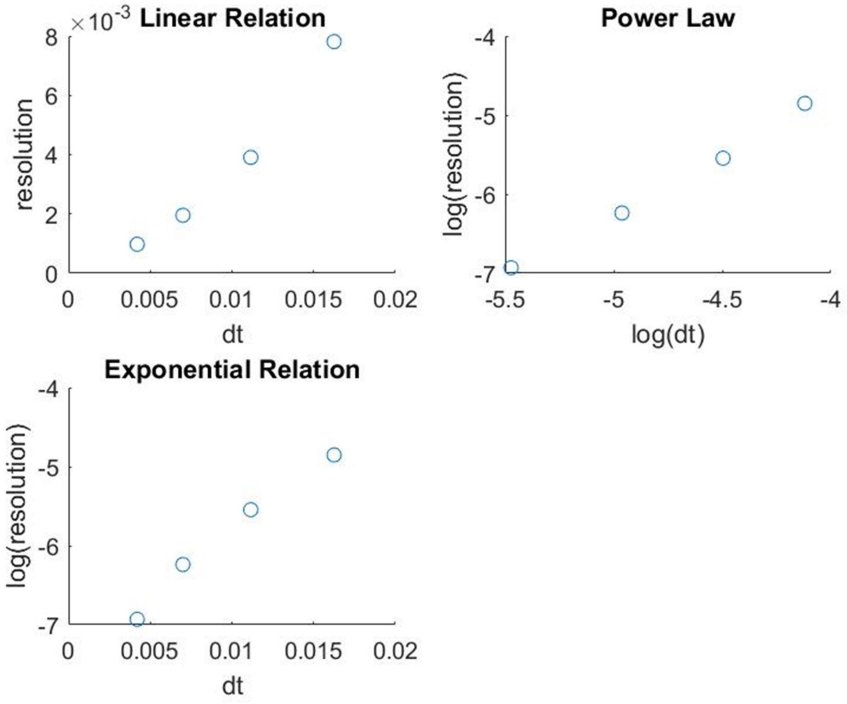
In addition, there is a point at which further increasing L has almost no effect; we have already let the HMC to explore sufficiently far away from the original path, so allowing it to take more steps yields us no further benefit. On the other hand, increasing δt also helped reduce autocorrelation, but 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 [1]. 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}.





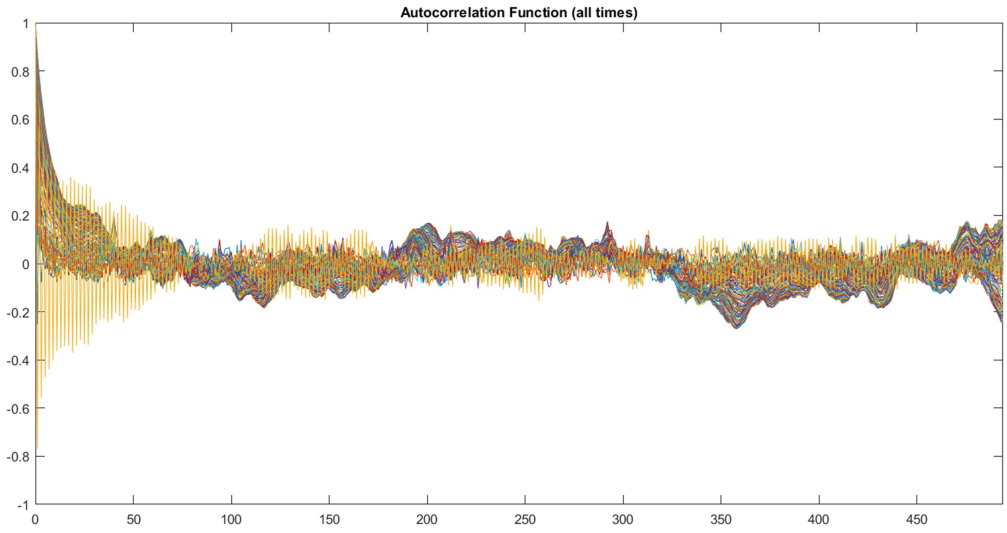
One of our main findings this quarter was that the effectiveness of the parameters δt and L is dependent on the resolution of the paths (the number of points in each path vector). So our next goal was to find a relationship between the resolution of the path vectors, Δt, and the δt that produced near optimal results at that resolution. We found that the following power law best describes the relationship between the two values: $\delta t = 0.2269(\Delta t)^{0.5434}$.



This power law gave fairly accurate predictions of appropriate δt values for other resolutions, but because the δt needs to be so finely tuned it had trouble meeting our ideal acceptance rate of 65.1% at finer resolutions; in the future we hope to create a more precise tuning law. We have also hypothesized that it might be the case that Δt ought to be raised to the power of .5 (instead of .5434). The quantity $\sqrt{\Delta t}$ is the standard deviation in our stochastic differential equation that we use to model the motion of a particle along a given free energy landscape. We believe that this quantity may be related to the tuning of δt.

Over the course of our experimentation we also came across some rather strange results that did not conform to the rules we had set up that defined what good paths were. These strange paths would move from the well at -1 to the well at +1, but the last point would not be within a standard deviation of 0.01 from +1 as we had specified. Instead the last point would be some very distant point from +1 for example +3. So, on the next path that the HMC would accept the last point would be at something like -2.9. The paths would continue alternating like this until they converged to the appropriate range around +1. So, the autocorrelation curve for this last point would oscillate wildly, but over time it would converge to zero. We think this occurs because when we seed the HMC with the original path vector it is just some random noise in the bottom of the well at -1. So, a path that does in fact reach the next well, except for its last point, would be accepted by the HMC. Then paths whose last point was closer to the appropriate value would still be accepted. So, it would take a long time for the HMC to correct this error. We also

found that randomizing the δt by a small amount would eliminate the problem, as this strange phenomenon only occurs at very specific values of δt and L. The image below shows the autocorrelation function of a simulation that exhibited the strange result.



We have a number of avenues to explore in the future. Currently our simulation is limited to one dimensional free energy landscapes. Our next main goal is to generalize it to higher dimensions; this is necessary if our simulation is to be applied to any real world problem (e.g. protein folding). We also wish to better implement path and time exchange algorithms. The premise of such algorithms is as follows: we run multiple simulations simultaneously at different resolutions or with different free energy landscapes, and then exchange paths between the simulations at certain points in time. Doing so encourages faster exploration of the state space and results in less correlated paths. These algorithms have already been implemented (and show promising results), but we would like to parallelize them and allow each simulation to use different parameters, for the sake of efficiency. In the future we also wish to explore the use of other sampling algorithms, as well as the possible use of a technique known as metadynamics in conjunction with our conditional path sampling method.

References:

[1] arXiv:1001.4460 [math.PR]

[2] Panos Stinis. Conditional path sampling for stochastic differential equations through drift relaxation. *Communications in Applied Mathematics and Computational Science*, 6(1):63-78, 2011.

[3] Radford M Neal et al. Mcmc using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113-162, 2011.