An Exploration of Hamiltonian Monte Carlo

Jake Coleman and Sayan Patra

1 Abstract

For our project we will use Neal's 2011 paper, "MCMC using Hamiltonian dynamics" [7]. The paper discusses how to use Hamiltonian dynamics as a sampling scheme to explore target spaces better than traditional Metropolis-Hastings algorithms. The Hamiltonian is the sum of potential energy (based on position) and kinetic energy (based on momentum) - Hamilton's equations relate the two partial derivatives of the Hamiltonian to each other, and define a mapping from the state at time t to the state at time $t + s$. In Hamiltonian Monte Carlo (HMC), we draw auxiliary momentum variables from a Gaussian distribution, and use Hamiltonian dynamics simulations to update the position variable (which follows the distribution of interest). At the end of a user-defined number of steps of simulation, the new variables are accepted or rejected in a Metropolis-Hastings step.

In this report we will explore basic HMC with the "Leapfrog" discretization method, and follow some examples (such as highly-correlated multivariate Gaussian distributions) comparing HMC to random-walk Metropolis Hastings that show improvement for HMC. We will establish the superiority of the HMC method over regular random walk sampling schemes in the case of higher dimensions. We will also implement an extension of HMC proposed by Neal (1994) that uses "windows" of states to allow for a high probability of acceptance for all trajectories. Finally, we demonstrated a method for simple parallelization that can increase speed by up to a factor of two

2 Introduction

Markov Chain Monte Carlo (MCMC) has been used to simulate distributions since the landmark paper by Metropolis et al. in 1953 [4]. In this report we follow Neal (2011) [7] in his exploration of Hamiltonian Monte Carlo (HMC), a variant of MCMC which utilizes properties physical properties of molecules known as "*Hamiltonian dynamics*." Hamiltonian dynamics rely on deterministic motion of molecules described by Newton's laws of motion, and were only applied as part of an MCMC algorithm in 1987 by Duane, Kennedy, Pendleton, and Roweth [1], with statistically applications beginning in 1996 with Neal's use on neural networks [6]. The marriage of deterministic molecular simulations and MCMC allows fast exploration of state spaces of distributions while avoiding random walks necessary in Metropolis-Hasting algorithms.

In the subsequent sections we describe the Hamiltonian dynamics (section 3) and how to construct a MCMC method using them (section 4). Like in other Markov Chain methods HMC has its issues of tuning, which are discussed briefly in section 4. Section 5 compares HMC to Metropolis random-walk, in both low and high dimensions. In section 6 a variation of HMC is presented, where the acceptance rate of HMC is shown to be increased by looking at the "windows" of states at the beginning and the end of the trajectories.

3 Background - Hamiltonian Dynamics

Hamiltonian dynamics describes the potential energy and kinetic energy of an object, linking them through Hamilton's Equations. The potential energy is a function of an object's position, while the kinetic energy is the function of an object's momentum. In HMC, the target distribution is related to the position variable, while the momentum variable is an auxiliary variable used in the deterministic updates via Hamilton's Equations.

3.1 Hamilton's Equations

Hamiltonian dynamics results from a certain set of differential equations. The d-dimensional vector p and q describe the full state space of $2d$ dimensions. The partial derivatives of the Hamiltonian $H(q, p)$ determine how the vectors change over time, as follows.

$$\begin{aligned}\frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i}\\ \frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i}\end{aligned} \tag{3.1}$$

for $i = 1, ..., d$. These equations map the state at any time t to a new state $t + s$, for continuous time. A discretization will be discussed in Section 3.3.

3.2 Potential and Kinetic energy

The total energy of the dynamics at point q, p of the state space is called the "Hamiltonian." denoted by $H(q, p)$, and is generally considered the sum of energy generated independently by position and the momentum vector i.e.

$$H(q, p) = U(q) + K(p) \tag{3.2}$$

Here, $U(q)$ and $K(p)$ are called the potential and kinetic energy respectively. The potential energy relates to the distribution from which we want to sample, as described in Section 4.1 The kinetic energy for ease of sampling is usually defined as

$$K(p) = p^T M^{-1} p/2 \tag{3.3}$$

where M is a symmetric, positive definite matrix. As is easily seen, $K(p)$ is proportional to the multivariate Gaussian distribution with mean 0 and covariance matrix M. When M is diagonal (as is often the case), then the sampled momentum variables p are independent.
Note that $K(p)$ is a quadratic form, and we know from linear algebra that

$$\frac{dx^T Ax}{dx} = x^T(A + A^T)$$

In our case, $A = M^{-1}$, which is symmetric. So $\frac{\partial K(p)}{\partial p} = p^T M^{-1}$

With these forms for H and K, Hamilton's equation 3.1 reduces to

$$\begin{aligned}\frac{dq_i}{dt} &= [p^T M^{-1}]_i\\ \frac{dp_i}{dt} &= -\frac{\partial U}{\partial q_i}\end{aligned} \tag{3.4}$$

3.3 Discretizing Hamiltons Equation

Hamilton's Equations describe how the system changes in continuous time - in practice, the dynamics is simulated with some small finite sample size ϵ and hence must be discretized. The discretizing methods is applicable for any form of $H(q, p)$, however we will assume the form in 3.2 as it simplifies the expressions. Also M is assumed to be diagonal with diagonal elements $m_1, ...m_d$, so that the kinetic energy has the following form.

$$K(p) = \sum_{i=1}^{d} \frac{p_i^2}{2m_i} \tag{3.5}$$

Which, of course, means that $\frac{\partial H(p,q)}{\partial p_i}(t) = p_i(t)/m_i$.
Euler's method is standard method to approximate the solution to a system of differential equation. However for HMC the Leapfrog method produces better results. Thus we will skip the Euler's method and go straight to Leapfrog method.

3.3.1 Leapfrog Method

The *leapfrog* method works as follows,

$$\begin{aligned}p_i(t + \epsilon/2) &= p_i(t) - (\epsilon/2)\frac{p_i}{dt}(t) &&= p_i(t) - (\epsilon/2)\frac{\partial U}{\partial q_i}(q(t))\\ q_i(t + \epsilon) &= q_i(t) + (\epsilon/2)\frac{p_i}{dt}(t + \epsilon/2) &&= q_i(t) + \epsilon\frac{p_i(t + \epsilon/2)}{m_i} \qquad 2\\ p_i(t + \epsilon) &= p_i(t + \epsilon/2) - (\epsilon/2)\frac{p_i}{dt}(t + \epsilon) &&= p_i(t + \epsilon/2) - (\epsilon/2)\frac{\partial U}{\partial q_i}(q(t + \epsilon))\end{aligned} \tag{3.6}$$

The algorithm is essentially a half-step in the momentum (auxiliary) variables, a full step in the position variables using the updated momentum, and then another half-step in the momentum using the updated position. This is for one leapfrog step - typically there are many leapfrog steps per iteration. For more steps, full momentum steps are taken in between position steps and the final half-momentum step.

Because 3.6 uses mere transformations, the leapfrog method preserves volume exactly. One achieves reversibility by simply negating p due to its symmetry. These two aspects are crucial properties necessary for MCMC, but are not discussed in depth here.

4 Background - Hamiltonian Monte Carlo

Recall that the main objective of MCMC is to simulation variables from target distributions. In our case, the variables of interest will be our momentum variables q, with target distributions uniquely determined by the potential energy function $U(q)$. Momentum variables p are introduced artificially as auxiliary random variables in order to complete the system dynamics. The following sections describe how to translate the target density into potentially energy for use in the Hamiltonian, and the provide the complete HMC algorithm.

4.1 Probability and the Hamiltonian

Statistical mechanics gives us a relationship between the target distribution and potential energy function called the *canonical distribution*. Given a temperature T and energy function $E(x)$, the canonical distribution has probability density (known as the "Boltzman distribution")

$$P(x) = \frac{1}{Z} exp(-E(x)/T) \tag{4.1}$$

where Z is the normalizing constant.
As the Hamiltonian $H(p, q)$ defined by Equation 3.2 is an energy function it induces a separable joint distribution over the position and the momentum as follows:

$$P(p, q) = \frac{1}{Z} exp(-H(p, q)/T) = \frac{1}{Z} exp(-U(q)/T) exp(-K(p)/T) \tag{4.2}$$

It is clear from Equation 4.2 that q and p are independent and each have canonical distribution with their own energy function. Thus the sample is generated by simulating an ergodic Markov chain that has the canonical distribution for (q, p) as its stationary distribution. We set the temperature equal to 1, unless we are using some tempering methods and define the potential energy to be

$$U(q) = -log\big[P(q)\big] \tag{4.3}$$

where $P(q)$ is the target distribution of interest.

4.2 The Algorithm

Each iteration of the HMC algorithm has two steps. Only the momentum is changed in the first step and both the position and the momentum is changed at the second step. This explores the target density $P(q)$ defined by $U(q)$ more efficiently than using a proposal probability distribution. Starting at an initial state (q_0, p_0), we simulate Hamiltonian dynamics for a short time using the leapfrog method. We then use the state of the position and momentum variables at the end of the simulation as our proposed states variables q^* and p^*. The proposed state is accepted using an update rule analogous to the Metropolis acceptance criterion. Specifically if the probability of the proposed state after Hamiltonian dynamics $P(q^*, p^*)$ is greater than probability of the state prior to the Hamiltonian dynamics $P(q_0, p_0)$ then the proposed state is accepted, otherwise, the proposed state is accepted randomly with probability equation to the ratio between proposed and current probabilities. If the state is rejected, the next state of the Markov chain is set as the state at $t - 1$. For a given set of initial conditions, Hamiltonian dynamics will follow contours of constant energy in phase space. Therefore we must randomly perturb the dynamics so as to explore all of $P(q)$. This is done by simply drawing a random momentum from the corresponding canonical distribution $P(p)$ before running the dynamics prior to each sampling iteration t. Combining these steps, sampling random momentum, followed by Hamiltonian dynamics and Metropolis acceptance criterion defines the HMC algorithm for drawing M samples from a target distribution:

1. set $t = 0$
2. generate an initial position state $q_0 \sim U(q)$

3. repeat until $t = M$
   (a) set $t = t + 1$
   (b) sample an initial momentum state $p_0 \sim K(p)$
   (c) set $x_0 = x^{(t-1)}$
   (d) run leapfrog starting at (p_0, q_0) for L steps and stepsize ϵ to obtain (q^*, p^*)
   (e) calculate acceptance probability $\alpha = min(1, exp(U(p_0) + K(q_0) - U(q^*) - K(p^*)))$
   (f) draw $u \sim U(0, 1)$
   (g) if $u \leq \alpha$, accept and set $x^{(t)} = x^*$, else set $x^{(t)} = x^{(t-1)}$

HMC satisfies the “detailed balance” criterion and hence leaves the canonical distribution invariant. The algorithm is also “ergodic” and therefore explores the entire sample space.

5 Illustration of HMC and its benefits

In this section we will use several examples to demonstrate the potential of the HMC algorithm described in previous section. We will compare these to random-walk Metropolis mostly. This section closely follows Neal (2011) [7], as we recreate his comparison plots.

5.1 Trajectories for a two-dimensional problem

In this example we are sampling from a two dimensional gaussian distribution with zero mean, standard deviation one and correlation 0.95. The momentum variables are generated from an independent standard two dimensional bivariate normal. Therefore the Hamitonian has the form

$$H(q, p) = q^T \Sigma^{-1} q/2 + p^T p/2, \quad \text{with} \quad \Sigma = \begin{bmatrix} 1 & 0.95 \\ 0.95 & 1 \end{bmatrix} \tag{5.1}$$

Figure(1) shows trajectories for the position coordinates and the momentum coordinates respectively along with the hamiltonian value for each of the $L = 25$ leapfrog steps with a stepsize of $\epsilon = 0.25$ that is used to generate the plots.

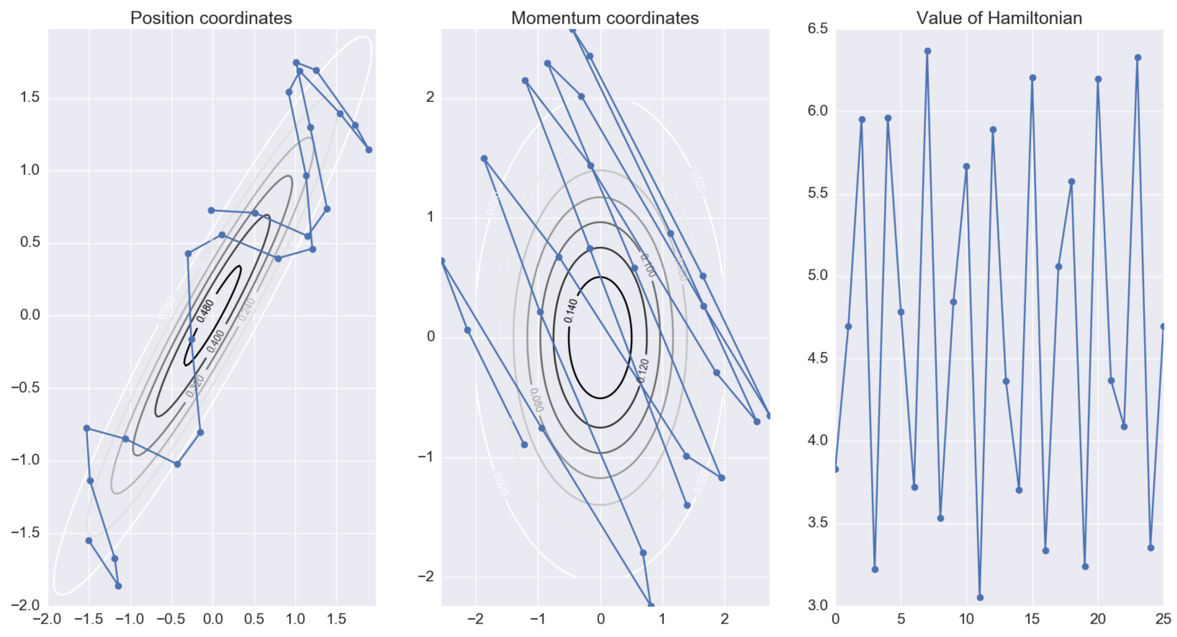

Figure 1: A trajectory for a 2D Gaussian distribution, simulated using 25 leapfrog steps with a stepsize of 0.25. The initial state for position variable q is $[-1.5, -1.55]^T$. Note how the position coordinate fully explores the space of the highly correlated distribution, while the momentum variables traverse the proposal space.

Note the differences between this and a random walk. The position variables move systematically throughout the space, reversing direction as dictated by the momentum coordinates so that the Hamiltonian oscillates appropriately. The small oscillations help the position variables fully explore the highly correlated space.

As in other methods, tuning HMC is important. Figure(2) demonstrates what can happen when the step size is catastrophically large.

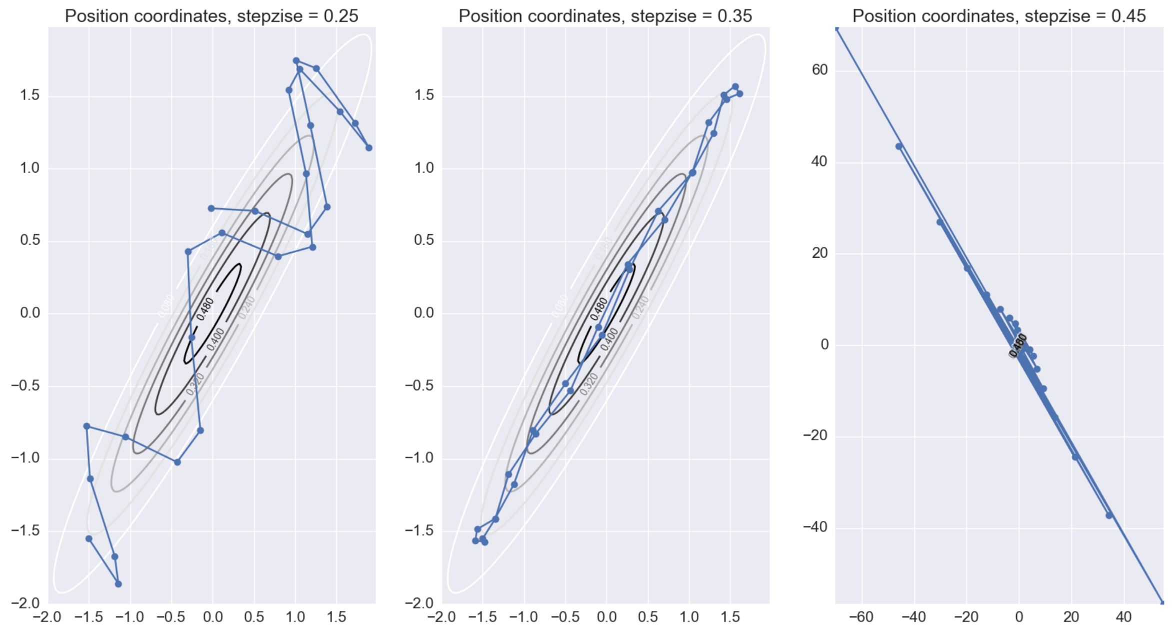

Figure 2: Illustration of Hamiltonian trajectories with increasing stepsize until it reaches critical state.

The step size is mostly controlled by the need to keep the smaller steps ize under control. If larger step size were used the oscillation would be larger in the hamiltonian values. At a critical step size ($\epsilon = 0.45$), the trajectory becomes unstable and the value of Hamiltonian grows without bound (figure 2). As long as the steps ize is less than that, the error in the hamiltonian stays bounded regardless of the number of leapfrog steps.

5.2 Sampling from a two-dimensional distribution

In this section we compare the trajectories between HMC and simple random-walk Metropolis method. The aim is to simulate from a bivariate Gaussian similar to the previous one but with stronger correlation of 0.98. The HMC used the same kinetic energy as before. The results of 20 HMC iteration with $L = 20$ leapfrog steps with stepsize $\epsilon = 0.18$ are shown in the right of figure(3). These values were chosen so that the trajectory length, ϵL, is sufficient to move to a distant point in the distribution without being so large that the trajectory will often back on itself. The rejection rate for this trajectories was 0.1.

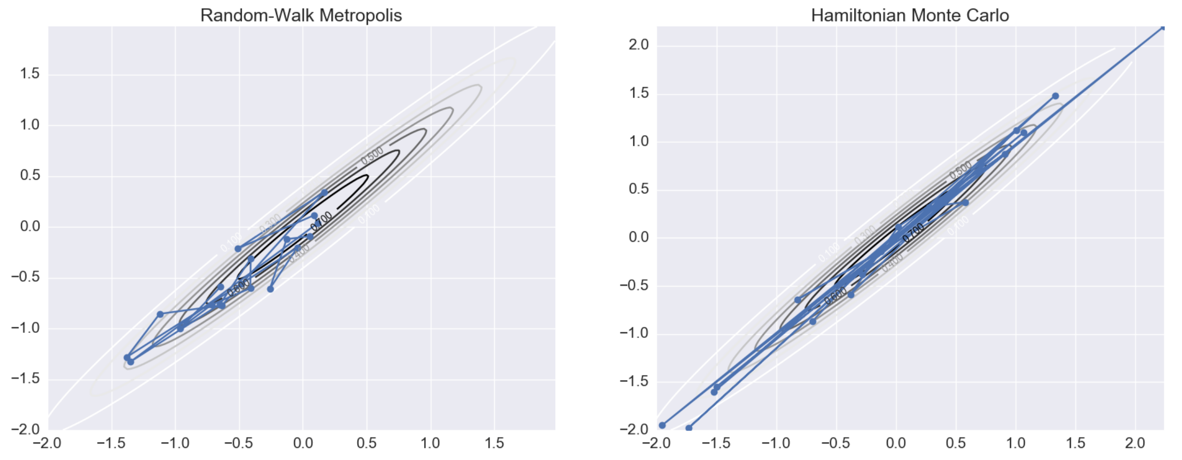

Figure 3: Twenty iterations from Metropolis (left) and HMC (right) sampling from bivariate Gaussians with .98 correlations. The Metropolis method used a thinning of 20 iterations to compare to HMC with 20 steps per iterations, but HMC still explores the space much better.

Left image of the figure(3) shows every 20th state from multiple iteration of random-walk Metropolis, with a bivaraite Gaussian proposal distribution (independent with standard deviations the same as step size for HMC, $\epsilon = 0.18$). The rejection rate for the random-walk proposals is 0.38. Even with comparable steps, it is clear that HMC explores the target space much better than Metropolis, which is hampered by high correlation between the two variables.

Figure(4) illustrates the result for 200 HMC iterations and random-walk Metropolis iterations (thinned by 20 iterations). It is evident the autocorrelation between samples is much less in HMC and hence the state space is traversed more uniformly.

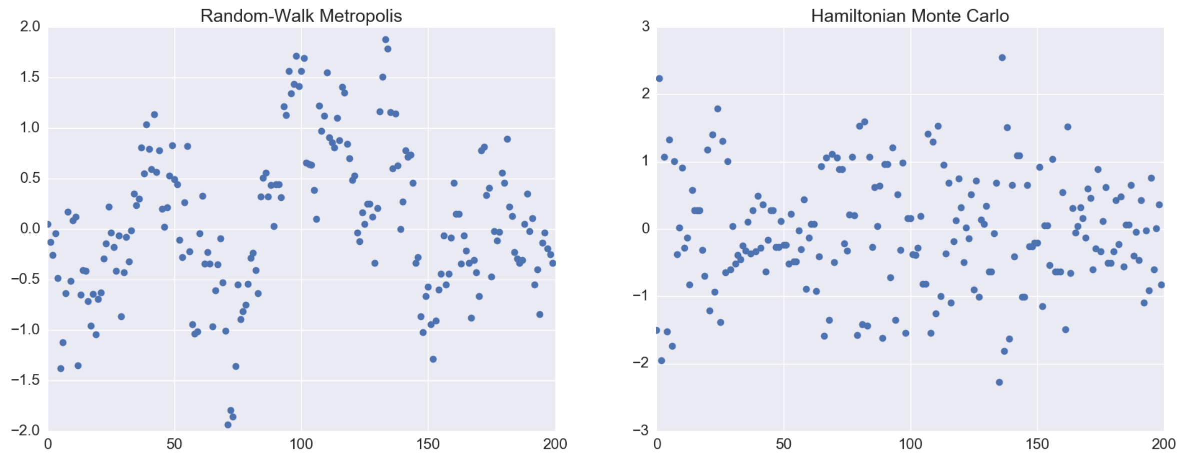

Figure 4: The first coordinate of the bivariate Gaussian sampled by the two methods. Metropolis (left) uses a thinning of 20 iterations to compare to HMC, which has 20 leapfrog steps per iteration. It is clear that there is high autocorrelation for Metropolis, even with this thinning, that is not apparent in HMC.

5.3 Efficiency in High dimension

In this section we show that HMC scales better than MH as a sampling algorithm. The target distribution to sample from is a 100 dimensional Gaussian distribution, where the variables are independent, with means of zero, and standard deviations of $0.01, 0.02, ..., 0.99, 1.00$. We will use the same Kinetic energy function for the HMC as in the previous section, however it is 100 dimensional instead of 2.

For step size, we chose a number uniformly in $0.013 \pm 20\%$ - we chose a random number at each iteration in order to avoid situations where a leapfrog produces a full or half cycle of variables with standard deviations that are multiples of the step size. 0.013 was chosen to keep the step size on the same order as the smallest standard deviation of the target variables - in practice, we would not know this ahead of time, but diagnostics can be used to find the right value (selecting appropriate step sizes and number of leapfrog steps is not covered in this report). 150 leapfrog steps were used in order to allow the sampler to fully explore the space for the variables with larger standard deviations.

Figure(5) shows the sample iterations for the last dimension (and thus largest standard deviation). To compensate for the 150 leapfrog steps, the Metropolis sampler thinned by 150 iterations.

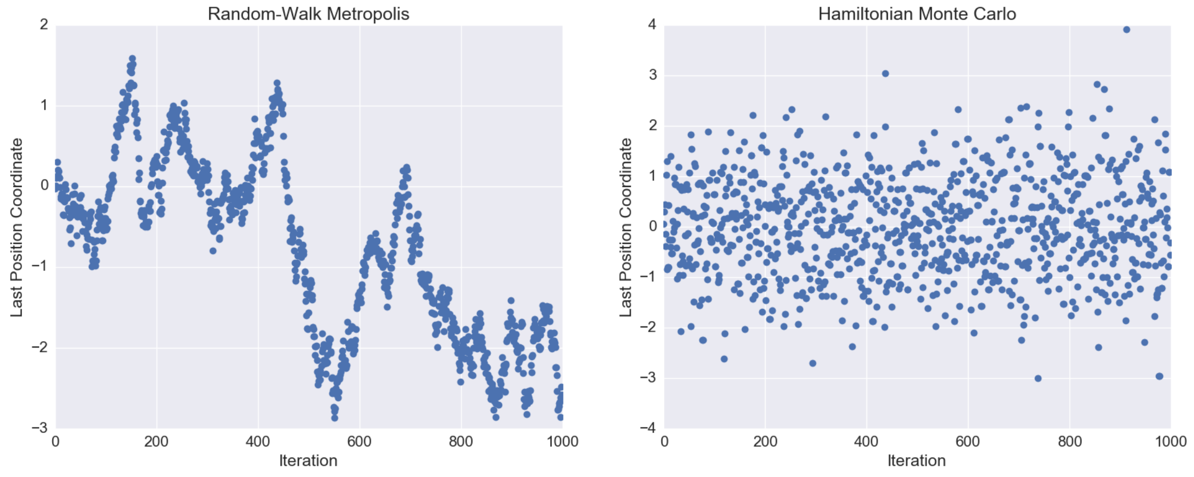

Figure 5: Iterations of sampled values for the last coordinate. The autocorrelation for random-walk Metropolis is even more pronounced.

Even when the target distributions have no correlations between variables, the high dimensionality causes the Metropolis sampler to be highly autocorrelated. Had the sampler not been thinned, the autocorrelation would have been even worse.

5.3.1 Tests

Here we raise the dimension to 100, sampling from independent multivariate Gaussians with increasing standard deviations proportional to dimension number. We expect all sampled variables to have mean zero, and standard deviation to lie along the line y=x. Figures(6) and (7) show the results for Metropolis and HMC. Note that this serves as a test - we know the theoretical target distribution, so we can see how the sampled summary statistics compare in order to assess how well HMC is working.

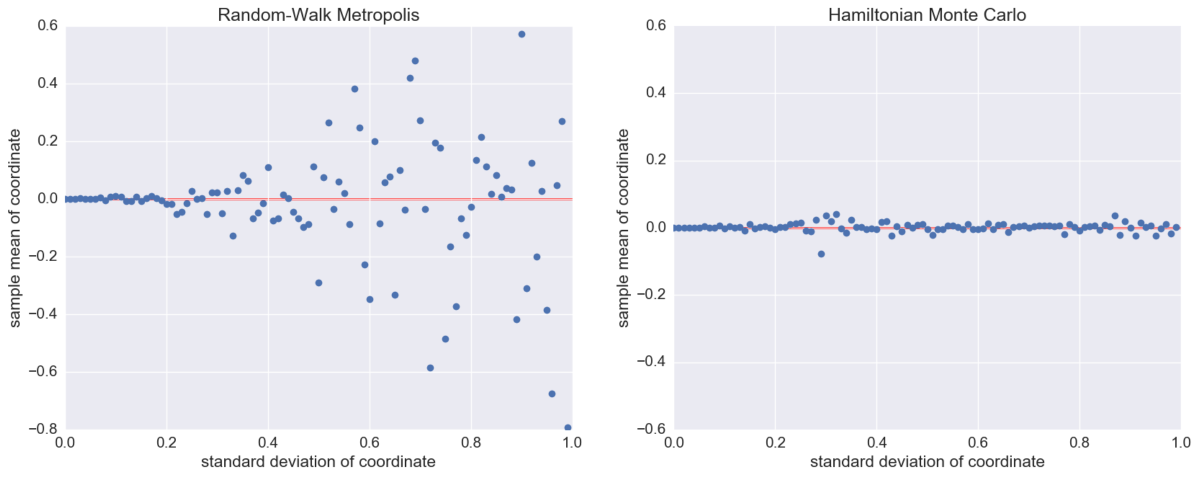

Figure 6: Sampled means for all 100 variables - we expect them all to be zero.

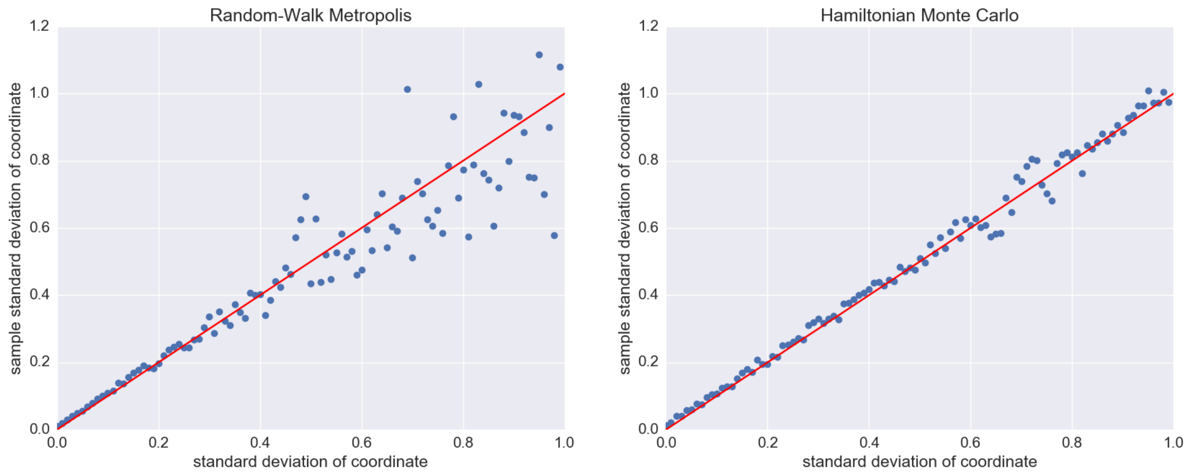

Figure 7: Sampled standard deviations for all 100 variables - we expect them to lie along the line y=x.

Clearly, Metropolis fails us in the higher dimensions in this case. As the dimension increases (along with the standard deviations), the means fluctuate wildly. The standard deviations of the sampled values stray from their true values, and become more erratic in the higher dimensions. HMC shows neither of these traits - all the means are close to zero, and the sampled standard deviations lie close to the line y=x even in higher dimensions. Even with very simple, independent distributions, Metropolis fails in the larger dimensions where HMC succeeds just as well as in the lower dimensions.

6 Application to Real Data

For our application to real data, we will demonstrate how the algorithm can be used to find the posterior values for parameters in simple linear regression. The model is as follows:

$$
\begin{aligned}
Y &\sim \mathcal{N}(X\beta, \phi^{-1}\mathcal{I}) \\
\beta &\sim \mathcal{N}(0, \tau^{-1}\mathcal{I}) \\
\phi &\sim \mathcal{G}(a, b)
\end{aligned}
$$

With hyperparameters $\tau, a, b = 0.001$ to keep the priors for β, and ϕ flat. We are most interested in the regression parameters β - from frequentist statistics, we know that the least squares estimate $\hat{\beta} = (X^TX)^{-1}X^TY$. If our MCMC works correctly, it should produce a posterior value centered around $\hat{\beta}$ - we can use this as a test. Before we perform HMC, however, we need the potential energy $U(\theta)$ and its gradient. In our case, $\theta = [\phi, \beta]$, with β p-dimensional. Recall that $U(\theta) = -\log(p(\theta))$ - in our case, $p(\theta) \propto \pi(\theta)L(Y \mid \theta)$, the posterior of $\theta \mid Y$. The derivation of $U(\theta)$ is below.

$$
\begin{aligned}
\text{Ł}(\theta \mid Y) &\propto L(Y \mid \phi, \beta)\pi(\beta)\pi(\phi) \\
&= \phi^{n/2}e^{-\frac{1}{2}\phi(Y-X\beta)^T(Y-X\beta)}e^{-\frac{1}{2}(\beta^T\tau\mathcal{I}\beta}\phi^{a-1}e^{-\phi b} \\
log(\theta \mid Y) &\propto (\frac{n}{2} + a - 1)\text{log}\phi - \frac{1}{2}\phi[\beta^TX^TX\beta - 2Y^TX\beta + Y^TY] - \frac{1}{2}\beta^T\tau\mathcal{I}\beta - \phi b \\
U(\theta) &= (-\frac{n}{2} - a + 1)\text{log}\phi + \phi b + \frac{1}{2}\{\beta^T[\phi X^TX + \tau\mathcal{I}]\beta - 2\phi Y^TX\beta + Y^TY\}
\end{aligned}
$$

From here we can get the partial derivatives

$$
\begin{aligned}
\frac{\partial U}{\partial \phi} &= \frac{-n - 2a + 2}{2\phi} + b + \frac{1}{2}\{\beta^T[X^TX]\beta - 2Y^TX\beta + Y^TY\} \\
\frac{\partial U}{\partial \beta} &= \beta^T[\phi X^TX + \tau\mathcal{I}] - \phi Y^TX
\end{aligned}
$$

We performed this analysis on a dataset of car speeds and stopping distances from 1920 [2]. We regressed stopping distance on speed, including an intercept - thus, β is 2-dimensional here. We intended on sampling both ϕ and β, but ϕ must be positive and required an extension of HMC that we were unable to provide for this report. Thus, we fixed phi at $(1/15)^2$ and focused on recovering the regression coefficients β. Figure 8 shows 1,000 draws via HMC, starting from the point [0,0].

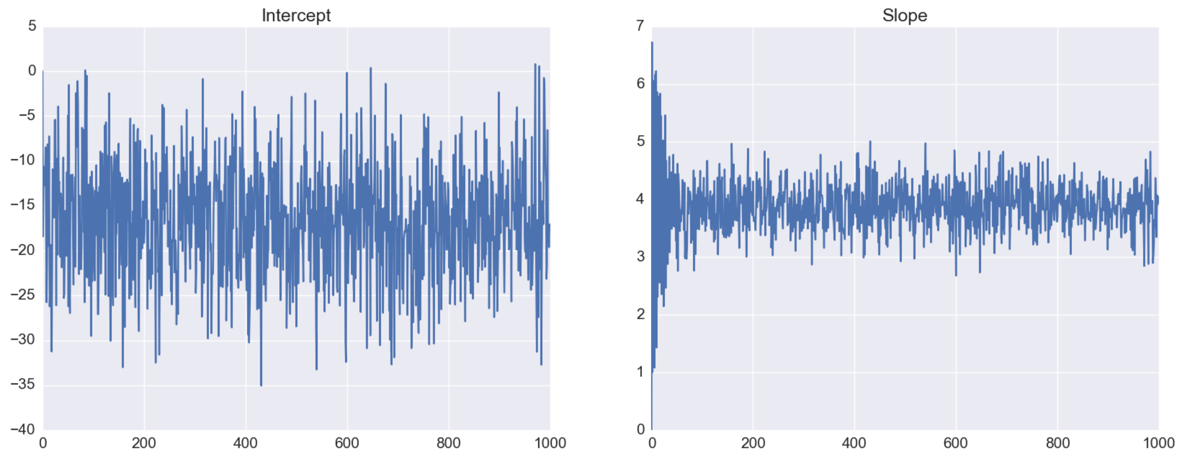

Figure 8: Posterior draws of linear regression coefficients β for the cars dataset.

As we can see from Figure 8, there is good convergence to around -17 for the intercept, and 4 for the slope. However, we also see that it takes some iterations for the slope posterior to converge. To adjust for this, we take a burn-in of 200 iterations for both parameters. To test to see how HMC is performing, we compare our estimated regression line with the build-in regression tools that perform frequentist analysis.

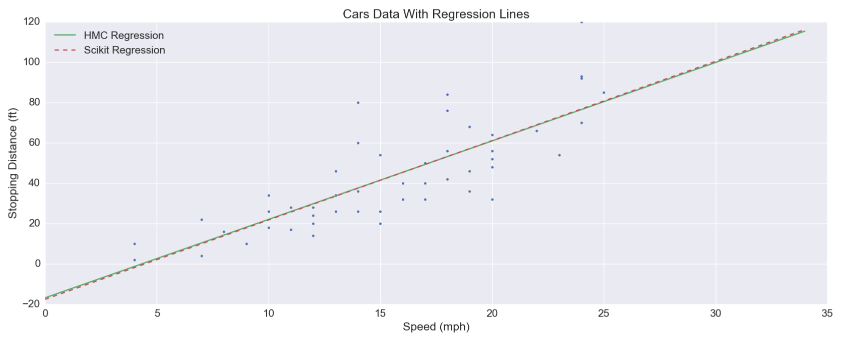

Figure 9: Cars dataset with regression lines. HMC performs well in finding the correct posterior values for β

Figure 9 shows that the frequentist regression line from the Scikit package and our posterior estimate are so close they are almost overlapping. (Part of this is due to the fact that we have very flat priors - improperly flat would have been the frequentist regression). This demonstrates the success of HMC, and we conclude that it does indeed perform well on real data.

7 Windowed States

For the final report, we plan on including a comparison to a modified version of HMC that helps obtain a higher probability of acceptance. This method, introduced by Neal in 1994 [?], uses "windows" of states to probabilistically map the pair (q, p) to a sequence of pairs, perform similar accept-reject steps, and then probabilistically map back. Following the 1994 paper, we will show improved efficiency of the algorithm.

The standard algorithm is as follows. For $1 \geq W \leq L + 1$ and K randomly chosen from $\{0, ..., W - 1\}$, beginning at the initial state, perform K leapfrog steps a random direction, and then $L - K$ from the initial state in the opposite direction. The reject window are the states from $-K$ to $-K + W - 1$, and the accept window are the states from $L - K - W + 1$ to $L - K$. Note that each window is W long and the whole leapfrog space is length L. The accept probability is the sum of the energy of the accept window divided by the sum of the energy of the reject window, chosen via a standard metropolis step. From the selected window, a specific state is then probability chosen with weight equal to its energy. Neal (1994)[?] also offers a method which does not require a second pass through the chosen window. We apply that algorithm in our report - it is given explicitly below.

Algorithm - Efficient Windowed States

1. Begin with state X_0 (in our case $X_i = (q_i, p_i)$), stepsize ϵ, leapfrog length L, window size W, offset K randomly chosen from $\{0, ...W-1\}$, and λ randomly chosen from $\{-1, 1\}$.

2. Perform K leapfrog steps from X_0 with stepsize $\epsilon_R = -\lambda\epsilon$. These are states $\{X_{-1}, ...X_{-K}\}$.

3. Perform $L-K$ leapfrog steps from X_0 with stepsite $\epsilon_A = +\lambda\epsilon$. These are $\{X_1, ..., X_{L-K}\}$.

4. Define the reject window $\mathcal{R}$ as $\{X_{-K}, ..., X_{-K+W-1}\}$ and the accept window $\mathcal{A}$ as $\{X_{L-K-W+1}, ..., X_{L-K}\}$.

   (a) Define $\mathcal{C}^1_{\mathcal{R}} = X_{-K}$, 'free energy' $\mathcal{F}^1_{\mathcal{R}} = -H(X_{-K})$

   (b) While traversing the accept window via leapfrog steps, set $\mathcal{F}^i_{\mathcal{R}} = -\log[\exp(-H(X_{-K+i})) + \exp(-\mathcal{F}^{i-1}_{\mathcal{R}})]$. Set $\mathcal{C}^i_{\mathcal{R}} = \mathcal{C}^{i-1}_{\mathcal{R}}$ with probability $\exp(-\mathcal{F}^{i-1}_{\mathcal{R}} + \mathcal{F}^i_{\mathcal{R}})$ and $\mathcal{C}^i_{\mathcal{R}} = X_{-K+i}$ with probability $\exp(-H(X_{-K+i}) + \mathcal{F}^i_{\mathcal{R}})$

   (c) Define $\mathcal{C}^1_{\mathcal{A}} = X_{L-K-W+1}$, 'free energy' $\mathcal{F}^1_{\mathcal{A}} = -H(X_{-L-K-W+1})$

   (d) While traversing the reject window via leapfrog steps, set $\mathcal{F}^i_{\mathcal{A}} = -\log[\exp(-H(X_{-L-K-W+1+i})) + \exp(-\mathcal{F}^{i-1}_{\mathcal{A}})]$. Set $\mathcal{C}^i_{\mathcal{A}} = \mathcal{C}^{i-1}_{\mathcal{A}}$ with probability $\exp(-\mathcal{F}^{i-1}_{\mathcal{A}} + \mathcal{F}^i_{\mathcal{A}})$ and $\mathcal{C}^i_{\mathcal{A}} = X_{-L-K-W+1+i}$ with probability $\exp(-H(X_{-L-K-W+1+i}) + \mathcal{F}^i_{\mathcal{A}})$

5. Choose $C^W_{\mathcal{A}}$ with probability $-\log\left(\frac{\sum_{X\in\mathcal{A}}\exp(-H(X))}{\sum_{X\in\mathcal{R}}\exp(-H(X))}\right)$ and $C^W_{\mathcal{R}}$ with 1 minus that probability.

To demonstrate that this works as effectively as standard HMC, we plot the first 200 iterations with windowed HMC next to those from standard HMC. The target distribution was a bivariate Gaussian with standard deviations of one and covariances of 0.98. This also serves as a test to make sure the windowed HMC is working - the samples should be clustered around zero (as they are).

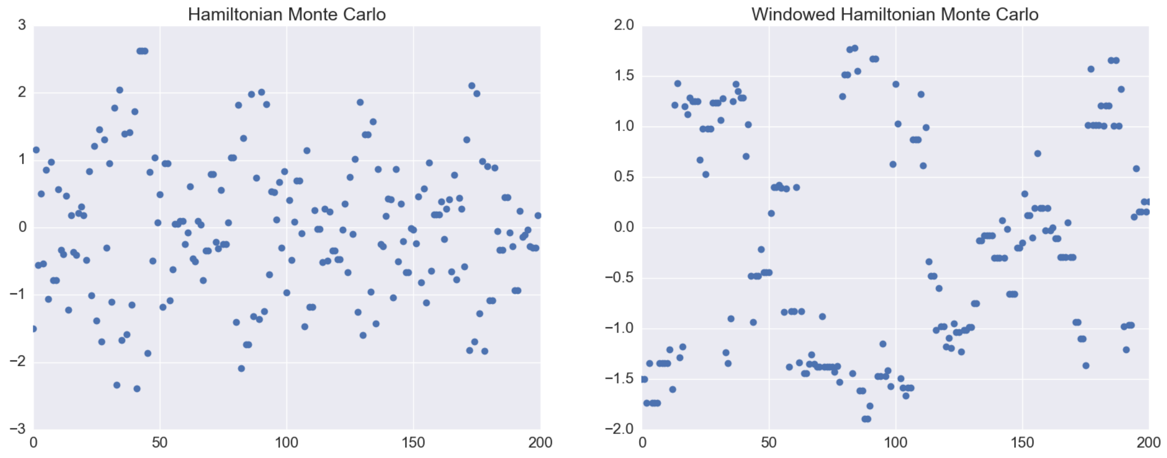

Figure 10: First 200 iterations of HMC (left) and windowed HMC (right) for bivariate Gaussian with standard deviations of 1 and covariance 0.98.

As Figure 10 shows, windowed HMC performs just as well as standard HMC with regards to exploring the space. Its value, however, is in the higher acceptance probability without sacrificing autocorrelation. To see this, we increase the dimensions to 10 and again compare to standard HMC.

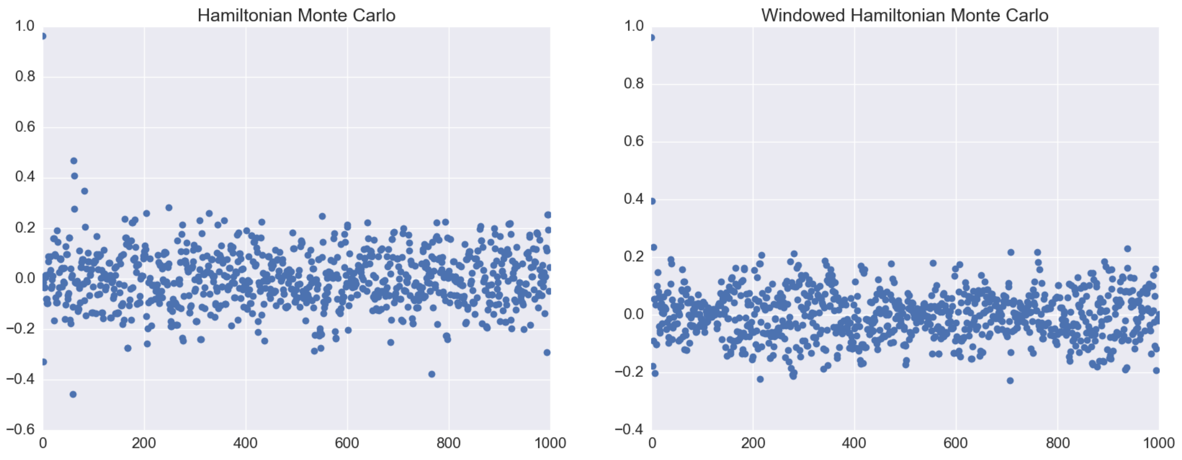

Figure 11: 1000 iterations of HMC (left) and windowed HMC (right) for the last dimension of a 10-dimensional Gaussian with standard deviations ranging from 0.1 to 1.

As we can see from Figure 11, again there is no loss in effective sample size with windowed HMC. The gain however, is in acceptance probability: we bump up from 87% acceptance to 90% acceptance. Though this may not seem like a significant increase, the difference becomes higher as dimension increases and the edge cases become more prevalent. For computational reasons, we do not perform higher dimensions.

8 Parallelization

We perform a trivial parallelization for MCMC. When sampling via MCMC, theory tells us that the chain will eventually converge to the target distribution. The standard method is to take a large burn-in, examine the trace plot, and look to see if convergence has occurred. If it has occurred, then each draw (if there is negligible autocorrelation, which is the case for HMC as we have seen) is an independent draw from the target distribution. We can thus parallelize easily by running the chain separately, taking a burn-in in the hopes of convergence, and then pool the draws together. If there is no autocorrelation and the chains have converged, then we should be able to run them for a factor less than the number of chains running to get the same number of draws (i.e. if we run 4 chains, we should only have to run 4 times fewer iterations).

To demonstrate improvement in speed via parallelization, we ran 10 chains in the bivariate Gaussian case as before, with windowed HMC. We got a speedup factor of anywhere ranging from 1.3 to 2, which is not insignificant. In Figure 12 we plot every 100 iterations of the first 5 chains - we can see that even when starting in random places, the chains converge around the dense portion of the contour plot.

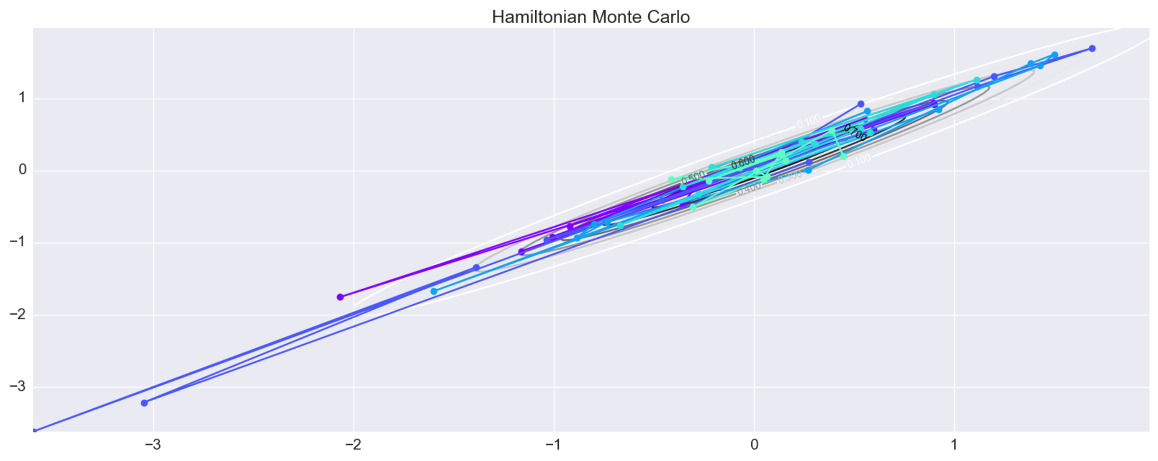

Figure 12: Every 100 iterations of windowed HMC on bivariate Gaussians with standard deviations of 1 and covariance of 0.98. 5 chains with random starting points are run

9 Conclusion

In this report we followed Neal's 2011 paper [7] in our exploration of Hamiltonian Monte Carlo, or HMC. We reproduced his plots demonstrating the effectiveness of HMC on correlated and high-dimensional data as compared to simpler Metropolis methods. Along the way, we provided tests to make sure HMC recovered the true values of target distributions. Applying HMC to real data, we found the regression line of stopping distance vs speed of cars and compared it to the frequentist version from built-in Python packages. In an extension, we implemented the modified algorithm suggested by Neal [**?**] that added a "window" of states to increase acceptance probability especially in edge cases. We showed that the windowed algorithm was as effective as regular HMC and showed improved acceptance probability even in low dimensions - improvements should only become higher in larger dimensions. Finally, we demonstrated a way in which the algorithm might be parallelized, getting a speedup of up to double. HMC is a valuable tool for bayesian analysts, and we look forward to considering it as an implementation tool when aiming to sample from target distributions.

References

[1] Duane, Simon; A.D. Kennedy, Brian J. Pendleton, and Duncan, Roweth (1987). "Hybrid Monte Carlo". Physics Letters B 195 (2): 216?222 1

[2] Ezekiel, M. (1930). *Methods of Correlation Analysis.* Wiley. [https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/cars.html] 8

[3] Leimkuhler, B., Reikh, S. (2004). "Simulating Hamiltonian dynamics". Cambridge University Press.

[4] Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. (1953). "Equations of State Calculations by Fast Computing Machines". Journal of Chemical Physics 21 (6): 1087?1092. 1

[5] Neal, R. M. (1994). "An improved acceptance procedure for the hybrid Monte Carlo algorithm". Journal of Computational Physics, vol. 111, pp. 194-203

[6] Neal R.M. (1996). "Bayesian Learning for Neural Networks." Lecture Notes in Statistics, No. 118. Springer-Verlag. 1

[7] Neal, R., M. (2011). "MCMC using Hamiltonian dynamics". arXiv:1206.1901v1 5(2),. 1, 4, 12