An alternative to MCMC: Hamiltonian Monte Carlo

Guru Sai Haveesh Singirikonda : EP17BTECH11010
Nenavath Prashanth Babu : EP17BTECH11013

The Metropolis-Hastings algorithm albeit simple has its own flaws, especially when needed to create samples in higher dimensions. This work is a short summary of the Hamiltonian or Hybrid Monte Carlo, which is a Markov Chain Monte Carlo algorithm. It attempts to remove the difficulties faced by conventional random-walk algorithms by using concepts from Hamiltonian Mechanics an alternative formulation of the famous Newton's Laws in Classical Physics. We will first discuss some of the concepts of Hamiltonian mechanics and the analogies we use here. We then discuss the algorithm of HMC with a pseudo-code for the same. Then we finish up by discussing the application of this algorithm through three examples.

1 Introduction

Markov Chain Monte Carlo (MCMC) methods are used to create samples from a probability distribution. MCMC algorithms are widely used nowadays spanning across various fields of research. A lot of effort has been put to improve these algorithms. The first and the most simplest MCMC algorithm was developed in [Metropolis et al., 1953], which was later named as the Metropolis-Hastings algorithm. A good summary of this method can be found on [Hogg and Foreman-Mackey, 2018].

Although it is a very simple and powerful algorithm, it is difficult to implement for creating samples of higher dimensions. This happens due to the 'random-walk' mechanism of the Metropolis-Hastings algorithm where the number of possibilities increases with increase in dimensions. With so many possible ways a random sample can be generated, the algorithm would not be able to converge quickly to the original distribution. Another issue while using Metropolis-Algorithm is the choice of the proposal distribution from which samples are drawn randomly from. This choice of the proposal probability distribution function is not a trivial one to make and it would affect the distribution of the samples created.

Many other alternatives to the Metropolis-Hastings algorithms have since been proposed to tackle these issues. Some of these methods are Gibbs Sampling, Metropolis-adjusted Langevin algorithm, Pseudo-marginal Metropolis–Hastings, Slice sampling, Multiple-try Metropolis, Reversible-jump and the Hamiltonian (or Hybrid) Monte Carlo. In this report we will be focusing on the Hamiltonian Monte Carlo which employs ideas from the Hamiltonian formulation of Classical Mechanics to traverse the sample-space to obtain samples for a given Probability Distribution Function.

Hamiltonian Monte Carlo or Hybrid Monte Carlo was originally developed in the late 1980s to work with calculations in Lattice Quantum Chromodynamics [Duane et al., 1987], a field focused on understanding the structure of the protons and neutrons that comprise nuclei and atoms. Within a few years this method gained attention and started to appear in a few textbooks. This technique started to serve as an alternative to other MCMC algorithms.

2 Hamiltonian Mechanics

The concepts used here to tackle the above problems are derived from the Hamiltonian formulation of Classical Mechanics. Hamiltonian mechanics is the reformulation of classical mechanics originally described by Newton's laws of Motion.

In Hamiltonian mechanics, a physical system is described by a set of coordinates (q, p) whose time evolution is governed by Hamilton's equations,

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{1}$$

$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \tag{2}$$

where $\mathcal{H}$ is the Hamiltonian, $\mathbf{q}$ is the generalized coordinates of the system and $\mathbf{p}$ is the corresponding momenta. The simplest interpretation of the Hamiltonian formulation is where the Hamiltonian represents the total energy of the system. Here, the Hamiltonian would be given by

$$\mathcal{H}(\mathbf{q}, \mathbf{p}, t) = U(\mathbf{q}, t) + K(\mathbf{p}, t) \tag{3}$$

where $U = U(\mathbf{q}, t)$ is the potential energy of the system and $K = K(\mathbf{p}, t)$ is the Kinetic Energy of the system.

The basic idea of Hamiltonian Monte Carlo is that the generalized coordinates, $\mathbf{q}$, correspond to the variables in interest and new samples, or new values of $\mathbf{q}$, are generated by solving equations 1 and 2 over the phase space. (The parameter vector when used for parameter estimation.). The momentum vector, $\mathbf{p}$ has no physical significance, but it will help us use Hamilton's equations to traverse through the space.

3 Hamiltonian Monte Carlo

For Hamiltonian Monte Carlo (HMC) we use U and K as scleronomic quantities. The Hamiltonian can be written as

$$\mathcal{H}(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}) \tag{4}$$

Here we can take the kinetic energy as $K(\mathbf{p}) = \mathbf{p}^2/2m$, just as in Classical Mechanics. This form of Kinetic Energy also corresponds to minus the log probability density (plus a constant) of the zero-mean Gaussian distribution. Whereas the potential energy $U(\mathbf{q})$ will be now defined as the minus the log probability density of the distribution we wish to sample, plus any constant that is convenient.

Now the Hamilton equations can be written as

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial U}{\partial \mathbf{q}} \tag{5}$$

$$\frac{d\mathbf{q}}{dt} = \frac{\partial K}{\partial \mathbf{p}} \tag{6}$$

We now put the Hamilton's equations, equations 5 and 6, to use to traverse through the probability space. The following steps would be followed -

$$\mathbf{q}_{new} = \mathbf{q} + d\mathbf{q} = \mathbf{q} + dt\frac{\partial K}{\partial \mathbf{p}}$$

and

$$\mathbf{p}_{new} = \mathbf{p} + d\mathbf{p} = \mathbf{p} - dt\frac{\partial U}{\partial \mathbf{q}}$$

The implementation of this would be discussed in the next section. More detailed explanations of this topic can be found in [Betancourt, 2017] and [Neal, 2012].

In this report we do not discuss why Hamiltonian Dynamics can be used in improving MCMC and its underlying advantages. Some of the properties of Hamiltonian Dynamics ,which makes it so useful in an improvement of MCMC, are discussed in section 2.2 of [Neal, 2012]. Construction of an efficient Hamiltonian Monte Carlo and its Robustness are discussed in sections 4 and 5 of [Betancourt, 2017].

Note that this method will work for any Kinetic Energy, not just the one we use here. For the sake of simplicity we also take the masses to be equal to 1. We can then say that the Kinetic Energy corresponds to minus the log probability of a Gaussian distribution with zero mean and variance equal to one.

So, for each run of a HMC, the initial point for $\mathbf{p}$ is drawn from a normal distribution of zero mean and variance one. There is no systematic way to decide the initial value for $\mathbf{q}$ and we just assume a suitable value for the same.

3.1 Discretizing Hamilton's Equations

For implementation on a computer program, we would have to approximate Hamilton's equations by making the time intervals discrete. Taking a small step size 'ϵ' for the time interval, we will iteratively compute $\mathbf{q}$ and $\mathbf{p}$ after a few steps using Hamilton's equations. Then we will be able to check if the new point is an acceptable choice.

The simplest way to do this is by using Euler's method to solve a differential equation. For Hamilton's equations each step, for each component of $\mathbf{q}$ and $\mathbf{p}$, would look like

$$p_i(t+\epsilon) = p_i(t) + \epsilon \frac{dp_i}{dt} = p_i(t) - \epsilon \frac{\partial U}{\partial q_i}(\mathbf{q}(t)) \tag{7}$$

$$q_i(t+\epsilon) = q_i(t) + \epsilon \frac{dq_i}{dt} = q_i(t) + \epsilon \frac{\partial K}{\partial p_i}(\mathbf{p}(t)) \tag{8}$$

But this method is not good as the trajectory in momentum and position space diverges out to infinity as illustrated in figure 1 of [Neal, 2012]. An improved version to obtain better results by modifying the steps a little,

$$p_i(t+\epsilon) = p_i(t) + \epsilon \frac{dp_i}{dt} = p_i(t) - \epsilon \frac{\partial U}{\partial q_i}(\mathbf{q}(t)) \tag{9}$$

$$q_i(t+\epsilon) = q_i(t) + \epsilon \frac{dq_i}{dt} = q_i(t) + \epsilon \frac{\partial K}{\partial p_i}(\mathbf{p}(t+\epsilon)) \tag{10}$$

Making these changes yield better results. But we can traverse the the coordinate-momentum space using the Leapfrog method of solving differential equations. The leapfrog method goes as follows -

$$p_i(t+\epsilon/2) = p_i(t) + (\epsilon/2) \frac{dp_i}{dt} = p_i(t) - (\epsilon/2) \frac{\partial U}{\partial q_i}(\mathbf{q}(t)) \tag{11}$$

$$q_i(t+\epsilon) = q_i(t) + \epsilon \frac{dq_i}{dt} = q_i(t) + \epsilon \frac{\partial K}{\partial p_i}(\mathbf{p}(t+\epsilon/2)) \tag{12}$$

$$p_i(t+\epsilon) = p_i(t+\epsilon/2) + (\epsilon/2) \frac{dp_i}{dt} = p_i(t+\epsilon/2) - (\epsilon/2) \frac{\partial U}{\partial q_i}(\mathbf{q}(t+\epsilon)) \tag{13}$$

We start with a half step for the momentum vectors. Then we evaluate the complete step in $\mathbf{q}$ and then we finish up the the remaining half step for momentum. This method gives us much better results.

We do not illustrate the different trajectories followed using the different methods. They can be found on figure 1 of [Neal, 2012].

3.2 Canonical Distributions

The distribution we wish to sample can be related to a potential energy function via the concept of a canonical distribution from statistical mechanics. This canonical distribution can be written as

$$\pi(q,p) = \pi(p|q)\pi(q) \tag{14}$$

Here we have chosen a conditional probability distribution over the auxiliary momentum vector, which ensures that if we marginalize out the momentum we immediately recover our target distribution [Betancourt, 2017]. We can write this canonical distribution in terms of the Hamiltonian $\mathcal{H}(q,p)$ as

$$\pi(q,p) = e^{-\mathcal{H}(q,p)}$$

Because of the relation mentioned in Equation 14, we can write this relation as

$$\mathcal{H}(q,p) = -log\pi(q,p) = -log\pi(p|q) - log\pi(q) \tag{15}$$

Drawing analogies, the term corresponding to distribution over momentum ,i.e $-log\pi(p|q)$, is the Kinetic Energy, $K(p,q)$ although, for the sake of simplicity, we will only consider Kinetic energies which depend only on momentum, $K = K(p)$. The second term corresponds to the target distribution, the probability distribution from which we want to draw samples from, which we have already shown that it will be the potential energy of the system

$$U(q) = -log\pi(q) = -log[P(q)L(q|D)]$$

where $P(q)$ is the prior density and $L(q|D)$ is the likelihood function given data D.

3.3 The Algorithm

The first step is to initialise the coordinate and momentum vectors. There is no systematic way to decide the initial point of **q** so we will be choosing the initial point based on our knowledge from a prior density or from some given constraints which we know from elsewhere. The initial point of the momentum vector, **p**, is simply drawn from the conditional probability distribution over the momentum vector,

$$p \leftarrow \pi(p|q)$$

which is again related to the kinetic energy as $K(p) = e^{-\pi(p|q)}$

Then we need to define the functions of the potential and kinetic energies. As mentioned before, the Potential Energy is

$$U(q) = -log\pi(q) = -log[P(q)L(q|D)]$$

and the Kinetic Energy is

$$K(p) = \sum_i \frac{p_i^2}{2}$$

The gradient of both these functions is trivial to compute and we will denote them as 'grad_U' and 'grad_K'.

We will be using the leap-frog method for traversing through the coordinate-momentum space. Samples will be chosen over current samples when

$$\frac{\pi(q_{new}, p_{new})}{\pi(q_{current}, p_{current})} > r \tag{16}$$

where r is a random number with $0 < r < 1$ drawn from a uniform distribution. This expression can also be written in terms of the Potential and Kinetic energies, which we will do subsequently.

The parameters of the algorithm are the step-size ϵ and the number of steps to get the new sample L. N denotes the number of samples to be created. The higher the value of N, more is the convergence of the samples to the original distribution. Although a very large value of N would increase the time, so a optimal number of samples must be chosen.

The algorithm/psuedo code for this algorithm is :

for i: 1 to N **do**
  $p_{current}$ = random_normal(mean=0,scale=1,size=q.size)
  p = $p_{current}$
  q = $q_{current}$
  samples = samples.append($q_{current}$)
  p = p - ϵ * grad_U(q)/2 {Make a half step for momentum in the beginning}
  for i: 1 to L **do**
    q = q + ϵ*grad_K(p) {Make a full step for position}
    if i!=L **then**
      p = p - ϵ * grad_U(q) {Make a full step for momentum}
    end if
  end for
  p = p -ϵ * grad_U(q)/2 {The final half-step for momentum}
  $U_{current}$ = U($q_{current}$)
  $K_{current}$ = K($p_{current}$)
  U_{new} = U(q)
  K_{new} = K(p)
  r = random_uniform()
  if exp($-U_{new} - K_{new} + U_{current} + K_{current}$)>r **then**
    $q_{current}$ = q
    $p_{current}$ = p
  end if
end for

4 Examples

We will now discuss a few examples where this algorithm has been put to use. Different values of path length, steps of leap-frog and number of samples are used in each example. The optimal choice of these parameters is required to obtain good results, which would depend on the distribution function we are sampling.

4.1 Sampling from a Gaussian Distribution

In this example, we simply create samples for a Gaussian distribution. This example is similar to the exercise given in [Hogg and Foreman-Mackey, 2018](Problem 2) where they use the Metropolis-Hastings algorithm to create samples for a Gaussian distribution. The Gaussian distribution of mean = 2 and a variance = 2 is taken as the target distribution.

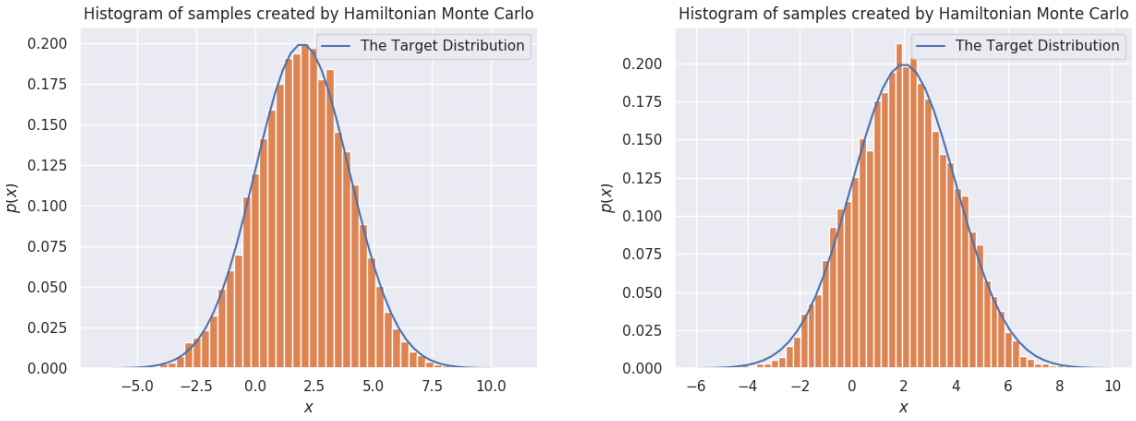
The histograms of the samples drawn from this distribution using HMC is plotted in Figure 1 for two different configurations. An initial value of $x = 0$ is used for both the cases. For the figure 1a we draw 10000 samples with a step-size $\epsilon = 0.3$. For the figure 1b we draw 20000 samples with a step-size $\epsilon = 0.6$. For both the figures, $L = 10$.

4.2 Fitting randomized data to a polynomial of order 5

In this example we use data with random errors drawn from a fifth order polynomial. As done in [emc,] for a linear model. The polynomial from which the data was drawn is -

$$y = -0.959 + 4.294x - 2.174x^2 + 1.687x^3 + 0.247x^4$$

which we will call as the true model. A Gaussian likelihood is used for this purpose where we can write the likelihood in terms of the χ^2 value. This HMC run returns 10000 samples with a step size



(a) N=10000, $\epsilon = 0.3$

(b) N=20000, $\epsilon = 0.6$

Figure 1: The histogram of the samples drawn from the Gaussian Distribution

$\epsilon = 10^{-3}$ and $L = 10$. After drawing samples for the coefficients of the best fit polynomial

$$y = a + bx + cx^2 + dx^3 + ex^4$$

, we get the values

$$[a, b, c, d, e] = [-0.817, 2.733, -2.457, 4.676, -1.206]$$

Figure 2 is a plot of the best fit model with the true model for comparison. Figure 3 is a corner plot of the samples. This corner plot was made using the python package `corner` [Foreman-Mackey et al., 2016]. Corner plots are generally used to show the results of a MCMC algorithm. Figure 4 shows the trajectories of the samples at each step.

4.3 Parameter estimation in the ΛCDM model

Although we have already shown fitting data using HMC, we would also like to include an application, an example from Cosmology. Proof of the accelerated expansion of the universe can obtained if we estimate the values of Ω_M and Ω_Λ, the cosmological parameters. There are various models of Cosmology and the ΛCDM model is considered as the standard model. We will not go into the details, but we will use data points of the Hubble parameter at different redshifts to obtain the values of Ω_M and Ω_Λ in the ΛCDM model. In this model the Hubble parameter is given as

$$H(z) = H_0\sqrt{\Omega_m(1+z)^3 + \Omega_\Lambda}$$

.

We start with a initial point of $[\Omega_M, \Omega_\Lambda, H_0)] = [0.5, 0.5, 70]$. We use a top hat prior which is non zero for $0 < \Omega_M < 1$, $0 < \Omega_\Lambda < 1$ and $60 < H_0 < 80$. Again a Gaussian likelihood is used, using the value of χ^2. We use this with the assumption that the errors from the observation are uncorrelated.

After running HMC for 20000 samples, with a step size of $\epsilon = 10^{-3}$, we get the best fit values for the parameters, along with the 1-σ confidence intervals, as following -

$$\Omega_M = 0.262^{+0.008}_{-0.008}$$

$$\Omega_\Lambda = 0.727^{+0.031}_{-0.033}$$

$$H_O = 70.595^{+0.308}_{-0.419}$$

Figure 5 is the corner plot for these samples. neat contours of the confidence intervals can be seen. The best fit values of the cosmological parameters match the generally accepted values for these quantities.

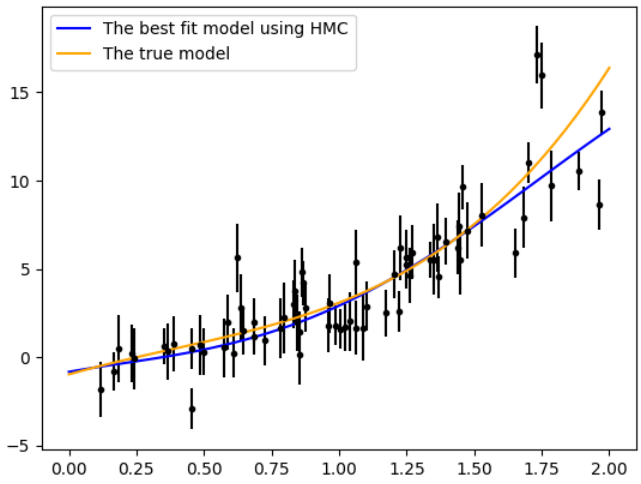


Figure 2: The best fit polynomial plotted alongside the true model, with the data points.

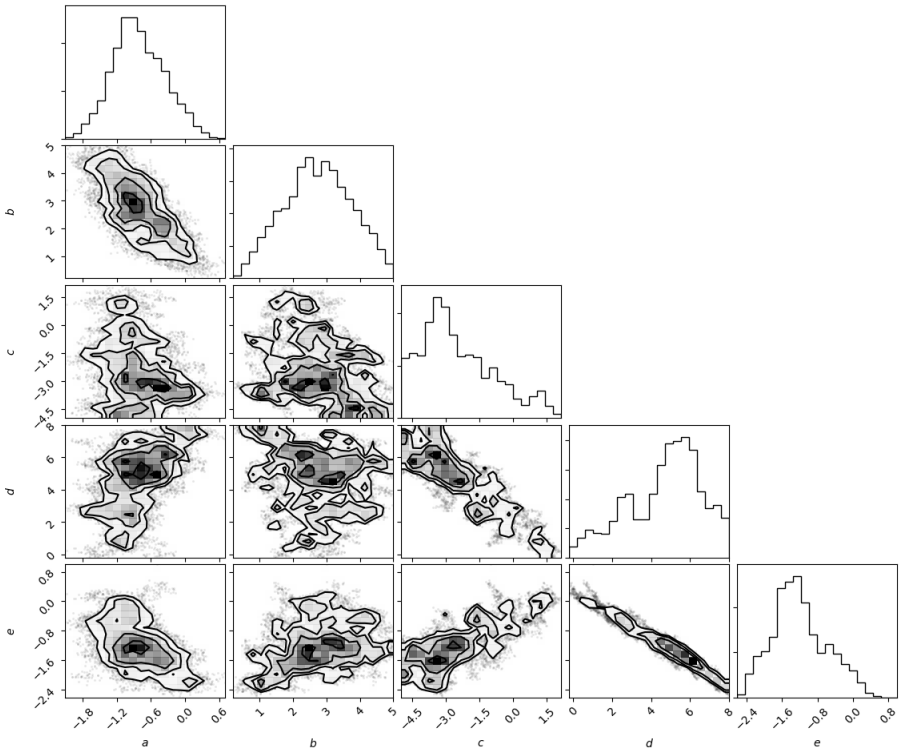


Figure 3: A corner plot of the samples for the coefficients of the polynomial

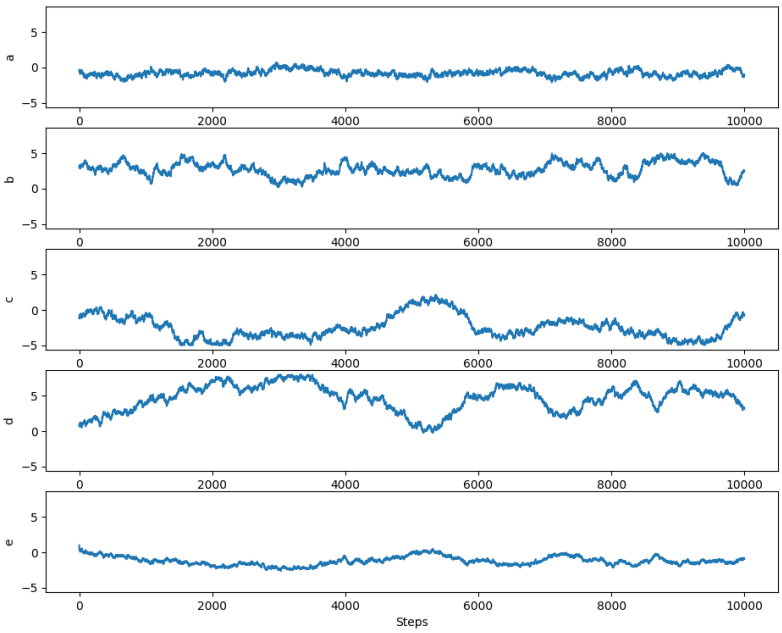


Figure 4: A plot of the trajectory taken for each coefficient of the polynomial

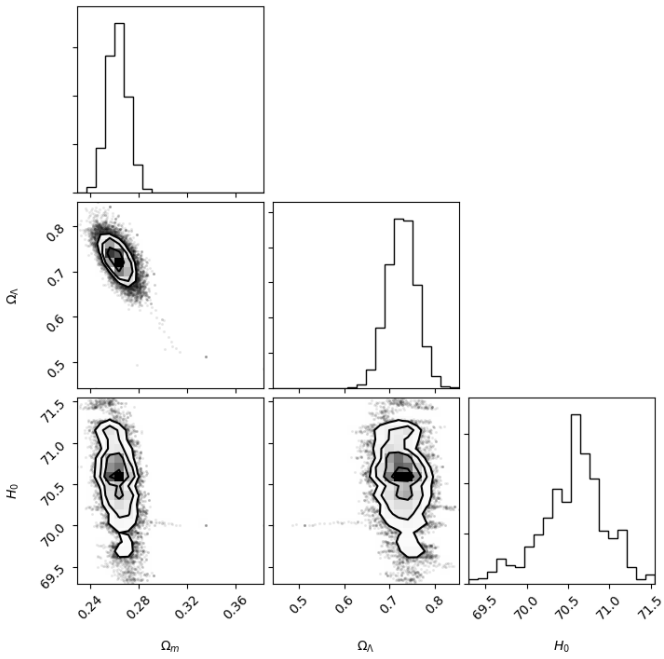


Figure 5: A corner plot of the samples drawn for the ΛCDM model

5 Conclusion

In this report, we have provided a short summary of the Hamiltonian or Hybrid Monte Carlo, an alternative method for MCMC sampling. This method draws concepts from Classical Physics to help traverse the probabilistic space, which is difficult with conventional methods of MCMC in higher dimensions. Hamilton's equations are used after introducing the momentum vector which, here, is just an abstract quantity without any physical meaning. Hamilton's equations are solved to create a new sample and then they are accepted based on the choosing criteria. We have also included a pseudo code for this algorithm here, in Section 3.3.

We have only discussed the simplest algorithm here, where we have taken the the kinetic energy $p^2/2m$. Here we have also taken the masses to be equal to one. The choice of mass can also affect the performance of HMC. The explanation of this is beyond the scope of this work, but can be found in [Neal, 2012]. [Neal, 2012] also explains why HMC works better than Metropolis-Hastings in much more detail. A similar explanation can be found in [Betancourt, 2017] where the author uses concepts from differential geometry to explain why HMC works better in higher dimensional probability spaces.

The next step would be to explore the countless possibilities by which we can improve this simplistic version of HMC, like a different choice of the Kinetic Energy function which could be dependent on both the coordinates an momentum, or choosing different masses for each dimension by which we can maybe segregate the important and nuisance parameters. Even better methods than the leap-frog method can be used to solve Hamilton's equations, like the RK-4 method, to get better results. Here we initialise the momentum vector from the same distribution, we could do something different. These improvements could lead to much more efficient Hamiltonian Monte Carlo which would probably ease the convergence of the samples to the target distribution.

The source codes for the three examples mentioned in Section 4 can be found on https://github.com/haveeshS/dsa. We have worked these examples on Python.

References

[emc,] emcee docs example: Fitting a model to data. https://emcee.readthedocs.io/en/v2.2.1/user/line/.

[Betancourt, 2017] Betancourt, M. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv e-prints*, page arXiv:1701.02434.

[Duane et al., 1987] Duane, S., Kennedy, A., Pendleton, B., and Roweth, D. (1987). Hybrid Monte Carlo. *Phys. Lett. B*, 195:216–222.

[Foreman-Mackey et al., 2016] Foreman-Mackey, D., Vousden, W., Price-Whelan, A., Pitkin, M., Zabalza, V., Ryan, G., Emily, Smith, M., Ashton, G., Cruz, K., Kerzendorf, W., Caswell, T. A., Hoyer, S., Barbary, K., Czekala, I., Rein, H., Gentry, E., Brewer, B. J., and Hogg, D. W. (2016). corner.py: corner.py v2.0.0.

[Hogg and Foreman-Mackey, 2018] Hogg, D. W. and Foreman-Mackey, D. (2018). Data analysis recipes: Using Markov Chain Monte Carlo. *Astrophys. J. Suppl.*, 236(1):11.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092.

[Neal, 2012] Neal, R. M. (2012). MCMC using Hamiltonian dynamics. *arXiv e-prints*, page arXiv:1206.1901.