The Hamiltonian Monte Carlo Algorithm in Parameter Estimation and Uncertainty Quantification

S. Subbey* (IMR), M. Alfaki (University of Bergen) & D. Haugland (University of Bergen)

SUMMARY

The Hamiltonian Monte Carlo (HMC) algorithm is a Markov Chain Monte Carlo (MCMC) technique, which combines the advantages of Hamiltonian dynamics methods and Metropolis Monte Carlo approach, to sample from complex distributions. The HMC algorithm incorporates gradient information in the dynamic trajectories and thus suppresses the random walk nature in traditional Markov chain simulation methods. This ensures rapid mixing, faster convergence, and improved efficiency of the Markov chain. The leapfrog method is generally used in discrete simulation of the dynamic transitions. In this paper, we refer to this as the leapfrog-HMC.

The primary goal of this paper is to present the HMC algorithm as a tool for rapid sampling of high dimensional and complex distributions, and demonstrate its advantages over the classical Metropolis Monte Carlo technique.

We demonstrate that the use of an adaptive-step discretization scheme in simulating the dynamic transitions results in an algorithm which significantly outperforms the leapfrog-HMC algorithm. Relevance to reservoir parameter estimation and uncertainty quantification will be discussed.
The Hamiltonian Monte Carlo (HMC) algorithm is a Markov Chain Monte Carlo (MCMC) technique, which combines the advantages of Hamiltonian dynamics methods and Metropolis Monte Carlo approach, to sample from complex distributions. The HMC algorithm incorporates gradient information in the dynamic trajectories and thus suppresses the random walk nature in traditional Markov chain simulation methods. This ensures rapid mixing, faster convergence, and improved efficiency of the Markov chain. The leapfrog method is generally used in discrete simulation of the dynamic transitions. In this paper, we refer to this as the leapfrog–HMC.

The primary goal of this paper is to present the HMC algorithm as a tool for rapid sampling of high dimensional and complex distributions, and demonstrate its advantages over the classical Metropolis Monte Carlo technique. We demonstrate that the use of an adaptive–step discretization scheme in simulating the dynamic transitions results in an algorithm which significantly outperforms the leapfrog–HMC algorithm.

Relevance to reservoir parameter estimation and uncertainty quantification will be discussed.

Introduction

Bayesian statistics allows update of prior belief by calculating a posterior probability density function (pdf), $p(m|\mathcal{O})$, given by (1). The function $p(m)$ denotes the prior belief about model $m$, while $p(\mathcal{O}|m)$ is the likelihood, which quantifies the probability that the observed observed data, $\mathcal{O}$, is explainable by the model for which the likelihood holds. Usually $m \equiv m(\mathbf{C})$ for $\mathbf{C} \in \mathbb{R}^k$, where $k$ is the number of parameters.

$$p(m|\mathcal{O}) = \frac{p(\mathcal{O}|m)p(m)}{\int_{\mathbb{R}^k} p(\mathcal{O}|m)p(m)dm}. \quad (1)$$

The posterior pdf provides the basis for inference about $\mathbf{C}$ and other derived quantities. Such inference usually requires computing a generic Bayesian integral, $\langle J \rangle$, as in (2), where $J(m)$ represents, e.g., the distribution of model parameters. Deriving $\langle J \rangle$ in (2) is conditioned on evaluating the normalizing constant in (1), i.e., $\int_{\mathbb{R}^k} p(\mathcal{O}|m)p(m)dm$. This integral could be intractable even for very low values of $k$. A Monte Carlo approach offers an attractive methodology for solving (1) without explicit evaluation of the normalizing constant. A Monte Carlo rendering of (2) is given by (3), where $n$ is the number of indexed models in an ensemble, and $m_j \equiv m(\mathbf{C}_j)$, with $j = 1, \ldots, n$.

$$\langle J \rangle = \int_{\mathbb{R}^k} J(m)p(m|\mathcal{O})dm, \quad (2)$$

$$= \frac{1}{n} \sum_{j=1}^{n} J(m_j)p(m_j|\mathcal{O})/h(m_j) \approx \frac{1}{n} \sum_{j=1}^{n} J(m_j), \text{ for } h(m_j) \approx p(m_j|\mathcal{O}). \quad (3)$$

---

* Institute for Marine Research, Bergen, Norway. Email: samuels@imr.no
† Univ. of Bergen, Dept. of Informatics, Norway. Email: mohammeda@ii.uib.no
‡ Univ. of Bergen, Dept. of Informatics, Norway. Email: dag@ii.uib.no
Thus if one derives an ensemble of models, \( m_1, m_2, \ldots, m_n \), whose distribution \( h(m) \), approximates \( p(m|O) \), the task of evaluating the generic integrals reduces to calculating averages over \( J(m_i) \). MCMC methods are based on Markov chains, which generate samples whose distribution approximate a given target distributions, such as \( p(m|O) \). The Metropolis-Hastings algorithm is perhaps the most popular of all implementation of MCMC algorithms. For MCMC methods to be effective, the set of parameters \( C_j \) must be as uncorrelated and independent as possible, [7, 8].

Most MCMC techniques, however, are slow in sampling from target distributions, require huge number of samples to evaluate \( \langle J \rangle \), and the samples generated are usually highly correlated. The methods become even more inefficient, especially when applied to high-dimensional target distributions. For detailed literature on MCMC methods, see e.g., [7, 18].

The Hamiltonian Monte Carlo Algorithm

The Hamiltonian Monte Carlo (HMC) algorithm [5] is a Markov Chain Monte Carlo (MCMC) technique, which combines the advantages of Hamiltonian dynamics [2, 19] and Metropolis Monte Carlo approach [9, 15], to sample from complex distributions. The HMC algorithm incorporates gradient information in the dynamic trajectories and thus suppresses the random walk nature in traditional Markov chain simulation methods. This ensures rapid mixing, faster convergence, and improved efficiency of the Markov chain.

The first step in the dynamical approach involves augmenting the vector of parameters \( C \in \mathbb{R}^k \), with a conjugate momentum vector \( P \in \mathbb{R}^k \) [17], and the introduction of a Hamiltonian function \( \mathcal{H}(C, P) \), defined on the phase–space \( (C, P) \). The Hamiltonian function is given by (4), where \( V, K \) and \( \pi(C) \) are the potential and kinetic energies, and the target distribution.

\[
\begin{align*}
\mathcal{H}(C, P) & = V(C) + K(P), \quad (4) \\
V(C) & = -\log \pi(C), \quad K(P) = \frac{1}{2}|P|^2. \quad (5)
\end{align*}
\]

If \( V(C) \) induces a Boltzmann distribution over \( C \) with a pdf given by (6) then \( \mathcal{H}(C, P) \) induces a similar distribution on \( (C, P) \), with a pdf given by (7), [16].

\[
\begin{align*}
p(C) & = \frac{e^{-V(C)}}{\int_{\mathbb{R}^n} e^{-V(C)}dC}, \quad (6) \\
p(C, P) & = \frac{e^{-\mathcal{H}(C, P)}}{\int_{\mathbb{R}^n} \int_{\mathbb{R}^n} e^{-\mathcal{H}(C, P)}dCdP} = p(C)p(P), \quad (7) \\
p(P) & = (2\pi)^{-n/2}e^{-\frac{1}{2}|P|^2}. \quad (8)
\end{align*}
\]

Thus if one simulates an ergodic Markov chain that has the Boltzmann distribution for \( (C, P) \) as its stationary distribution, then values of \( C \) from successive Markov chain states with marginal distribution given by (6), can be used to estimate \( \langle J \rangle \) in (3) [16].

In simulating the required Markov chain, each step consists of drawing a new pair of \( (C, P) \) according to \( p(C, P) \), starting from the current \( C \), and a random variable \( P \), according to (8). The dynamic transitions are governed by Hamiltonian equations of motion (9), where \( \tau \) is fictitious time.

\[
\frac{dC}{d\tau} = +\frac{\partial \mathcal{H}}{\partial P} = P, \quad \frac{dP}{d\tau} = -\frac{\partial \mathcal{H}}{\partial C} = -\nabla V(C). \quad (9)
\]

The Hamiltonian dynamic transitions satisfy the conditions of time reversibility (invariance under \( \tau \rightarrow -\tau, \quad P \rightarrow -P \)), conservation of energy \( (\mathcal{H}(C, P) \) remains invariant with time), and Liouville’s theorem (conservation of phase–space volume). If the trajectories are long enough so that the equilibrium configuration is independent of the starting configuration, the dynamical
transitions sample regions of constant $\mathcal{H}$ without bias, i.e., if the initial configuration is such that $(C^0, P^0) \sim p$, then future configurations $(C^t, P^t) \sim p$.

The stochastic transitions consist in replacing $P$ with a value from (8). The consequence is that $\mathcal{H}(C, P)$ remains invariant while different regions of $\mathcal{H}$ are explored, and ensures ergodicity of the Markov chain [16].

**Practical Implementation – The Leapfrog HMC Algorithm**

In practice however, the Hamiltonian dynamics is simulated by the leapfrog algorithm [10], over $L$ leapfrog–steps, using a finite step size, $\epsilon$, according to (10)–(12). The choice of $\epsilon$ and $L$ are decisive in the solution obtained. Fig. 1 shows example plots for a 2D uncorrelated Gaussian.

![Figure 1: Effect of varying $\epsilon$ and $L$. Observe different scaling of $\delta\mathcal{H}$ axis. $\Delta j \equiv$ iteration count.](image)

The leapfrog scheme is popular, partly because it is simple to implement, and often exhibits higher accuracy and stability than other second-order schemes [13].

\[
P(\tau + \frac{\epsilon}{2}) = P(\tau) - \frac{\epsilon}{2} \nabla V(C(\tau)), \quad (10)
\]

\[
C(\tau + \epsilon) = C(\tau) + \epsilon P(\tau + \frac{\epsilon}{2}), \quad (11)
\]

\[
P(\tau + \epsilon) = P(\tau + \frac{\epsilon}{2}) - \frac{\epsilon}{2} \nabla V(C(\tau + \epsilon)). \quad (12)
\]

Though each leapfrog transition is volume–preserving and time–reversible, finite $\epsilon$ does not keep $\mathcal{H}$ constant. Hence systematic error is introduced into the sampling procedure.

The error introduced by non-zero step–size can be eliminated by considering the end-point configuration of each leapfrog transition as a candidate for the next state of the Markov chain, based on the Metropolis rule [7, 16], given by (13), where the current and candidate Hamiltonian are given by $\mathcal{H}$ and $\mathcal{H}'$, respectively.

\[
\min\{1, e^{[\mathcal{H}(C, P) - \mathcal{H}'(C, P)]}\}. \quad (13)
\]

For a given choice of $L$, simulating the dynamics with constant step–size, $\epsilon$, usually leads to extra computation cost, especially when the system changes along different regions of its trajectory. The extra computation cost can be reduced by using an adaptive step–size to simulate the dynamics.

**A Variable Step–Size Hamiltonian Monte Carlo Algorithm (SVHMC)**

The approach in adaptive time–stepping is an attempt by the integrator (roughly speaking) to solve a perturbed time–dependent Hamiltonian, $\mathcal{H}(C, P) + \delta \mathcal{H}(C, P, \tau)$, where $\delta \mathcal{H}$ is a perturbation of the Hamiltonian. Several schemes based on this approach have been reported in the literature, see e.g., [3]. An explicit variable step–size using a Runge–Kutta scheme (Störmer–Verlet discretization) is reported in [12]. The scheme adopted in this paper is based on [11], which generalizes and simplifies the integrator in [12].
The authors, in [12], introduced a variable modification of the Störmer–Verlet scheme by introducing a fictive variable $\rho$, which was related to a scaling function $G$. The resulting variable step–size Störmer–Verlet scheme is explicit if $G$ depends only on $C$ and semi-explicit if $G$ depends on $P$. Equations (14)–(17) summarize the resulting explicit variable step–size and time reversible Störmer–Verlet integration scheme.

\[
C_{n+\frac{1}{2}} = C_n + \frac{\epsilon P_{n+\frac{1}{2}}}{2\rho_n}, \quad P_{n+\frac{1}{2}} = P_n - \frac{\epsilon \nabla V(C_n)}{2\rho_n},
\]

\[
2G(C_{n+\frac{1}{2}}, P_{n+\frac{1}{2}}) = \rho_{n+1} + \rho_n,
\]

\[
P_{n+1} = P_{n+\frac{1}{2}} - \frac{\epsilon}{2\rho_{n+1}} \nabla V(C_{n+1}),
\]

\[
C_{n+1} = C_{n+\frac{1}{2}} + \frac{\epsilon}{2\rho_{n+1}} P_{n+\frac{1}{2}}.
\]

The scheme is fully explicit, symmetric and time–reversible if $G$ satisfies $G(C, P) = G(C, -P)$. An initial value for the fictive variable $\rho$, and the scaling function, $G$ must be determined in order to implement the scheme. The paper suggests $\rho_0 = G(C_0)$. However, a modified initialization of $\rho_0$ is suggested as an inappropriate choice could introduce undesirable oscillations in the numerically computed values of $\rho_n$. Choices of this function can be found in [11, 12]. In this paper, we adopt the definition of $G$ as in (18), and refer to this variant of the HMC algorithm with variable step–size as the SVHMC algorithm.

\[
G(C, P) = \sqrt{V'(C)|V'(C)|^T + P^T V''(C) V''(C) P},
\]

\[
V'(C) = |\nabla V(C)|^T, \quad V''(C) = |\nabla V'(C)|^T.
\]

**Evaluation Criteria**

This paper follows [6, 14, 20] in adopting a spectral analysis approach to evaluating the comparative performance of the HMC and SVHMC algorithms (Algorithms 1 and 2 below, respectively), in terms of ability to generate independent realizations, convergence rate, and efficiency of the chain.

The power spectral density (PSD) function for a MCMC chain of length $N$ (after burn-in) can be approximately fitted with the template in (20), where $P(0)$ is the PSD of white noise region for $k \to 0$, the parameter $\kappa^*$ indicates the transition point from uncorrelated realizations (white noise) to a different power law behavior (characterized by $\alpha$) and $j$ is a small nonzero positive integer, see Fig. 3. The parameter $\kappa^*$ is also indicative of the effective number of samples in a given chain. If $\sigma$ is the standard deviation of the target distribution, then for an ideal chain, $P(\kappa) = \sigma^2$ for all $\kappa$.

Another set of diagnostics used in this paper are the sample means, $\overline{C}$, of a finite chain and the dimensionless efficiency, $E$. The use of the variance of the sample mean, $\sigma^2_{\overline{C}}$, as a diagnostic tool, has been discussed in [4]. In [6], the authors referred to the dimensionless ratio, $r = \sigma^2_{\overline{C}} / \sigma^2_0$, as the convergence ratio, where $\sigma^2_0$ is the variance of the underlying distribution.
Further, [6] presented how $r$ could be estimated for a single chain, and a proposed a cutoff of $r < 0.01$ to guarantee convergence. For an infinite MCMC chain, $E$ quantifies how the sample mean variance compares to an ideal (completely) uncorrelated chain.

It can be shown that if the chain is normalized to have unit variance, an estimate for the convergence rate, $r$, and efficiency ratio, $E_r$, are given by (22) and (23), where $E_1, P_i(0)$ and $E_2, P_2(0)$ refer to the efficiency and $P(k \to 0)$ values for, e.g., Algorithm 1 and Algorithm 2, respectively.

$$P(k) \approx P(0) \frac{(\kappa^* / \kappa)^{\alpha}}{(\kappa^* / \kappa)^{\alpha} + 1},$$  

(20)  

$$k = j(2\pi/N),$$  

(21)  

$$r \approx P(0)/N,$$  

(22)  

$$E_r = E_1/E_2 \approx P_2(0)/P_1(0).$$  

(23)

Figure 3: PSD template with $(\alpha \approx 2)$

### Algorithm 1: Leapfrog HMC

1. Initialize $C_0$ and $P_0$
2. Set $\epsilon$
3. for $i = 1$ to nSamples do
   4. draw $P \sim N(0, I)$
   5. $(C(i), P(0)) = (C_{i-1}, P)$
   6. for $j = 1$ to $L$ do
      7. $P(j-\frac{1}{2}) = P(j-1) - \frac{\epsilon}{2} \nabla V(C(j-1))$
      8. $C(j) = C(j-1) + \epsilon P(j-\frac{1}{2})$
      9. $P(j) = P(j-\frac{1}{2}) - \frac{\epsilon}{2} \nabla V(C(j))$
   10. end
   11. $(C', P') = (C(L), P(L))$
12. draw $\alpha \sim U[0, 1]$  
13. if $\alpha < \exp(-\delta H)$ then  
14. $(C_i, P_i) = (C', P')$
   15. else  
16. $(C_i, P_i) = (C_{i-1}, P_{i-1})$
17. end
18. end
19. return $(C_i, P_i)_{i=0}^{n\text{samples}}$

### Algorithm 2: Störmer–Verlet HMC

1. Initialize $C_0$ and $P_0$
2. Set $\epsilon$
3. for $i = 1$ to nSamples do
   4. draw $P \sim N(0, I)$
   5. $(C(0), P(0)) = (C_{i-1}, P)$
6. Set $\rho_0$
7. for $j = 1$ to $L$ do
   8. $P(j-\frac{1}{2}) = P(j-1) - \frac{\rho_0}{2} \nabla V(C(j-1))$
   9. $C(j) = C(j-1) + \rho_0 P(j-\frac{1}{2})$
10. $\rho_j = -\rho_{j-1} + 2G(C(j-\frac{1}{2}), C(j\frac{1}{2}))$
11. $C(j) = C(j-\frac{1}{2}) + \rho_j P(j-\frac{1}{2})$
12. $P(j) = P(j-\frac{1}{2}) - \frac{\rho_j}{2} \nabla V(C(j))$
   13. end
14. $(C', P') = (C(0), P(0))$
15. draw $\alpha \sim U[0, 1]$  
16. if $\alpha < \exp(-\delta H)$ then  
17. $(C_i, P_i) = (C', P')$
   18. else  
19. $(C_i, P_i) = (C_{i-1}, P_{i-1})$
20. end
21. end
22. return $(C_i, P_i)_{i=0}^{n\text{samples}}$

### Numerical Experiments

This paper will present an application of the variable step Störmer–Verlet discretization scheme to Gaussian target distributions in 64 and 128 dimensions. See [1] for other numerical experiments and more details about strategies for improving efficiency of the HMC algorithm.

The general equation for the target distributions is given by (24), where $D$ is the dimension of $C$ and $\Sigma$ is the variance–covariance matrix. For simplicity, this paper will investigate Gaussian targets with uncorrelated covariates, i.e., $\Sigma = I \in R^{D \times D}$ (the identity matrix). For the Metropolis-Hastings type algorithm, the optimal proposal distribution for such a target is $\Sigma = \left( \frac{2\pi D}{\sigma_0^2} \right) \sigma_0 I$, where $\sigma_0$ is the standard deviation of the target distribution, see [6].

$$\pi(C) = \frac{1}{(2\pi)^{\frac{D}{2}} \text{det}(\Sigma)^{\frac{1}{2}}} \exp \left( -\frac{1}{2} C^T \Sigma^{-1} C \right).$$  

(24)
Results and Discussion of Numerical Experiments
In order to compare the Störmer–Verlet discretization and the leapfrog scheme, we first need to
determine an optimal value for \( \epsilon \), and the starting \( \rho_0 \) value for the SVHMC algorithm. We use
a pragmatic approach, which involves running short chains of the algorithms with increasing
(discrete) values for \( \epsilon \) and \( \rho_0 \), and adopting values that optimize the efficiency, \( E \).

The same number of leapfrog steps, \( L = 10 \), was used in each of the numerical experiments,
and a total of 2000 iterations was performed for each of HMC and SVHMC algorithms in 64 and
128–dimensions. The length of the chain generated by the MH algorithm was constrained by
the CPU time required to generate 2000 iterates of the HMC algorithm, see Table 1. The results
presented in this paper are based on analysis of chains corresponding to the first parameter.

Figure 4 shows the chain realizations, and the fitted PSD-templates for all the algorithms in 64
and 128 dimensions. Using the fit, we quantify the efficiency, convergence ratio and effective
number of samples in the chain.

![Figure 4: Fitted templates (top row) & PSD (bottom row) – MH, SVHMC & HMC in 64 and 128–dimensions.](image)

Table 1 presents a summary statistic of the numerical experiments. The table shows (the
obvious result) that given the same CPU time, the MH algorithm generates more iterates than
the HMC algorithm. Using the threshold value of \( r < 0.1 \), it could be said that both algorithms
(MH and HMC) have converged. Even though convergence of the algorithms is of essence,
the emphasis of the comparison is to determine which algorithm (MH or HMC) shows superior
performance, given the same CPU time, and assuming that gradients (\( \nabla V \)) can be calculated.
It is interesting to note that, in 64 dimensions, the HMC algorithm significantly
outperforms the classical MH algorithm in terms of efficiency (\( E_r = E_{\text{HMC}} / E_{\text{MH}} = 8.25 \)) and
effective sample size (\( \kappa_r = \kappa_{\text{HMC}} / \kappa_{\text{MH}} = 72.5 \)). In 128 dimensions, the comparative performance
indices confirm the gain in using the HMC algorithm, with \( E_r = 11.17 \) and \( \kappa_r = 77.5 \).

Whereas the SVHMC and HMC algorithms have comparable acceptance rates, the effi-
ciency ratio, \( E_r = E_{\text{SVHMC}} / E_{\text{HMC}} \approx 1.4 \), for the target distribution in 64 and 128 dimensions. In
other words, about 40% improvement in efficiency is obtained (for these particular targets) by
adopting an adaptive step approach in simulating the dynamic trajectories.

An idea about the gain in effective number of samples is obtained by looking at the ratio
\( \kappa_r = \kappa_{\text{SVHMC}} / \kappa_{\text{HMC}} \approx 2 \), i.e., about 100% increase in the effective number of samples.

These improved results from the SVHMC algorithm (acceptance rates and effective number
of samples) are however obtained with increase in CPU time (\( \Delta T_{\text{CPU}} = T_{\text{SVHMC CPU}} - T_{\text{HMC CPU}} \)).
However, a CPU time increase becomes almost insignificant as a drawback, when compared
to gain e.g., in the effective number of samples, which is decisive for correct resolution of the
target distribution. The 100% increase in the effective number of samples, for example, only
Table 1: Summary statistics for numerical experiment

<table>
<thead>
<tr>
<th></th>
<th>64 Dimensions</th>
<th></th>
<th>128 Dimensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P(0)</td>
<td>13.10</td>
<td>1.11</td>
<td>1.51</td>
<td>10.86</td>
</tr>
<tr>
<td>κ*</td>
<td>0.02</td>
<td>3.19</td>
<td>1.45</td>
<td>0.02</td>
</tr>
<tr>
<td>E</td>
<td>0.08</td>
<td>0.90</td>
<td>0.66</td>
<td>0.06</td>
</tr>
<tr>
<td>r</td>
<td>0.0008</td>
<td>0.0005</td>
<td>0.0007</td>
<td>0.0081</td>
</tr>
<tr>
<td>CPU time [sec]</td>
<td>557.38</td>
<td>669.40</td>
<td>557.38</td>
<td>1117.78</td>
</tr>
</tbody>
</table>

implies \( \sim 20\% \) (for 64 dimensions) and \( \sim 40\% \) (for 128 dimensions) increase in the CPU time.

The convergence ratio of the HMC and SVHMC algorithms remain approximately constant (irrespective of the target dimension) at \( 7 \cdot 10^{-4} \) and \( 5 \cdot 10^{-4} \), respectively, with both values being below the threshold value of 0.01 (as proposed in [6]). Our general observation is that beyond some threshold dimension, the convergence ratio of the HMC algorithm appears to remain approximately invariant.

Relevance to Parameter and Uncertainty Quantification in Reservoir Modeling

In many applications in reservoir modeling, the problem of estimating model parameters, given data, can often be formulated as a least squares problem, which is solvable in the Bayesian framework. Assume that \( G : \mathcal{R}^l \rightarrow \mathcal{R}^n \) is a given mathematical model, and that the vectors \( c \in \mathcal{R}^l \) and \( O \in \mathcal{R}^n \) represent sought model parameters and observed data, respectively. The relationship \( O \sim G(c) \), is usually non-linear. In the simplest case (neglecting model errors and assuming normal distribution on data errors), (25) gives the likelihood (consistent with \( l_2 \)-norm), where \( \Sigma \) is the variance–covariance matrix of the data errors, and \( A \) is an appropriate constant. Consistent with notations in this paper, the potential energy, \( E \), is given by (26).

\[
p(O|m) = A \exp \left( -\frac{1}{2} \left(G(c) - O\right)^T \Sigma^{-1} \left(G(c) - O\right) \right), \tag{25}
\]

\[
E(c) \sim \left(G(c) - O\right)^T \Sigma^{-1} \left(G(c) - O\right). \tag{26}
\]

Since \( c \) enters (26) non-linearly, the challenge lies in deriving \( \nabla V(c) \). Specifically for reservoir modeling (results to be published elsewhere), our approach has been to use an adjoint method (Automatic differentiation) in calculating the derivatives.

Conclusions

This paper has presented the Hamiltonian Monte Carlo (HMC) algorithm as an effective method for sampling high dimensional target distributions. Our results demonstrate that the HMC algorithm significantly outperforms the MH algorithm in sampling Gaussian target distributions with uncorrelated covariates. We have investigated a further improvement in the performance of the HMC algorithm by simulating the Hamiltonian dynamics using a variable step–size Störmer–Verlet discretization scheme. Our results show that the use of variable step–size can lead to about 40\% improvement in the efficiency, and up to 100\% increase in effective sample size of the chain produced by the HMC algorithm. These results have significance for reservoir modeling, which usually involves high dimensional target distributions and where the CPU time for generating a single realization in an MCMC chain can be high.

Practical application and gain in using the HMC/SVHMC algorithm lies in the ease with which \( \nabla V(c) \) can be evaluated. Our experience has shown that the use of automatic differentiation (possibly even in parallel/distributed–mode) provides a viable solution. Our specific results on application to reservoir parameter estimation and uncertainty quantification will be published elsewhere.
References


