Improving Efficiency in Parameter Estimation Using the Hamiltonian Monte Carlo Algorithm

by

Mohammed Alfaki

A thesis submitted in partial fulfillment for the
degree of Master of Science
in the

Faculty of Mathematics and Natural Sciences
Department of Informatics
Optimization Group

May 2008

UNIVERSITY OF BERGEN

Abstract

Faculty of Mathematics and Natural Sciences
Department of Informatics

Master of Science

by

Mohammed Alfaki

The Hamiltonian Monte Carlo algorithm, or alternately called hybrid Monte Carlo is Markov chain Monte Carlo technique, which combines a Gibbs sampling update with the Metropolis acceptance-rejection rule. The Hamiltonian Monte Carlo algorithm simulates the distribution using Hamiltonian dynamics which, involves gradient information to investigate the distribution space, and thus has better convergence properties than Metropolis–Hastings and Gibbs sampling algorithms.

The Hamiltonian Monte Carlo algorithm suffers from random walk in generating the dynamics momentum, and an additional error when the dynamics is simulated using constant step–size.

This thesis investigates three approaches to improve the performance of the Hamiltonian Monte Carlo algorithm. The first approach enhances the Hamiltonian Monte Carlo by suppressing random walk in the Gibbs sampling using ordered over–relaxation. The second approach investigates the simulation of the Hamiltonian dynamics using an adaptive step–size to reduce the error of the simulation. The third proposal is to combine the two versions into one algorithm.

Acknowledgements

I am very grateful to my supervisor Dr. Sam Subbey for the privilege of having excellent guidance and constant support during this work. My knowledege has benefited greatly from his expertise, enthusiasm and encouragement. I also wish to express my warmest gratitude to my co-supervisor Prof. Dag Haugland for the numerous invaluable suggestions during this work and for giving valuable suggestions to the manuscript. I also thank the administration staff at the department of Informatics. I am enormously grateful to the Norwegian state educational loan fund for the support during my master study.

The main body of this work was carried out at the Institute of Marine Research (IMR). I thank the staff, researchers and employees at this institute for the comfortable working environment and excellent facilities for this work.

Especially warm thanks go to my parents Altoma and Ali, my wife Sara, my family, and to all my friends for their constant support and patience during this work.

Contents

Abstract i

Acknowledgements ii

Contents iii

List of Algorithms v

List of Figures vi

List of Tables viii

1 Overview 1
- 1.1 Motivation . . . 1
- 1.2 Goal of this thesis . . . 2

2 Inverse theory 3
- 2.1 Introduction to inverse problems . . . 3
- 2.2 Deterministic approach to solving inverse problems . . . 4
- 2.3 Statistical inversion . . . 6
- 2.4 Definitions . . . 6
- 2.5 Bayesian model for inverse problems . . . 8

3 Markov Chain Monte Carlo algorithms 10
- 3.1 Basic definitions . . . 10
- 3.2 Simulation of the Markov chains . . . 11
  - 3.2.1 The Gibbs sampler algorithm . . . 11
  - 3.2.2 The Metropolis-Hastings algorithm . . . 13
- 3.3 Classical MCMC in high dimensions . . . 14

4 Hamiltonian Monte Carlo algorithm 16
- 4.1 Definitions . . . 16
- 4.2 The Hamiltonian equations . . . 19
- 4.3 Classical Hamiltonian Monte Carlo algorithm . . . 22

5 Problem definition 25
- 5.1 The HMC algorithms in practice . . . 25
  - 5.1.1 Step–size effect . . . 25
  - 5.1.2 Constant vs. adaptive step–size . . . 25
  - 5.1.3 Number of simulation steps . . . 26
  - 5.1.4 The random walk in choosing the momentum . . . 27

6 Improving the Hamiltonian Monte Carlo algorithm **28**
6.1 Basic definitions . . . 28
6.2 Over–relaxation methods . . . 29
6.2.1 Ordered over-relaxation . . . 29
6.3 Symplectic integrators . . . 31
6.4 Strategies for improving the HMC . . . 32
6.4.1 Restricting random walk in the momentum . . . 33
6.4.2 Adaptive step–size . . . 34
6.5 Combining ordered Over-relaxation and Störmer–Verlet . . . 36

7 Criteria for evaluating the improved HMC algorithm **38**
7.1 Degree of the correlation criteria . . . 38
7.1.1 Spectral analysis criteria . . . 40
7.1.2 Testing convergence of a chain . . . 44

8 Simulation and results **45**
8.1 Gaussian target and parameter estimation . . . 45
8.2 Distributions for the numerical experiments . . . 47
8.3 Results from numerical experiments . . . 47
8.3.1 Comparing OHMC and HMC algorithms . . . 47
8.3.2 Comparing SVHMC and HMC algorithms . . . 50
8.3.3 Comparing OSVHMC and SVHMC algorithms . . . 53
8.4 Conclusion . . . 56

9 Suggestions for further works **57**

A Abstract accepted at Thiele conference **59**

B Abstract accepted at ECMOR XI conference **61**

Bibliography **64**

List of Algorithms

3.1 The Gibbs Sampler 12
3.2 The Metropolis-Hastings 13
4.1 The Hamiltonian Monte Carlo–leapfrog 23
6.1 Ordered Over-relaxation 31
6.2 HMC algorithm with ordered over–relaxation applied to the Gibbs sampling stage to drawn the momentum variables. 33
6.3 HMC algorithm with Störmer–Verlet discretization used to simulate the Hamiltonian dynamics 35
6.4 HMC algorithm using ordered over–relaxation to pick the momentum variables, and Störmer–Verlet scheme to simulate the Hamiltonian dynamics. 37

List of Figures

2.1 The figure shows how small error in the data lead to a different model. Ideally we expect that the error in model should be proportional to the data error. This is the typical features of the inverse problem. Here $A = G^{-1} : \mathcal{D} \to \mathcal{M}$, and δ is the uncertainty in the data. 4

3.1 The histogram of the chain of 1000 samples generated according to Gibbs sampler Algorithm 3.1 is illustrated on the left, and the estimated marginal distribution for x_1 from equation (3.3) on the right of the plot, when the parameters $\alpha = 2$, and $\beta = 4$. 12

3.2 10000 sample form the Cauchy distribution left plot. The histogram and the Cauchy distribution curve drawn in the same plot (the right plot), which gives an acceptable match. 14

4.1 The contour plot (left), and 3-dimensional plot of the correlated Gaussian distribution defined in (4.39). 23

4.2 The marginal distributions and contour plots for the correlated 2–dimensional Gaussian distribution, sampled from both Hamiltonian Monte Carlo (HMC) on the left, and Metropolis–Hastings (MH) on the right, with chain length $n = 4000$ for each. It is clear that HMC captured the target distribution very well, while MH appears trapped, leading to a skewed distribution. 24

5.1 The figure shows the effect of varying the step–size $\epsilon = 0.025, 0.1$, and 1.9, in the histogram of the chain (top row), and the corresponding change in the total energy (bottom row). Observe the difference in scaling of δH axis. 26

5.2 The effect of using adaptive step–size instead of constant steps. The target distribution is a 2–dimensional Gaussian, $\Sigma = I$. 26

5.3 The figure shows the effect of using different values for the number of leapfrog steps $L = 5, 10$, and 30, in the histogram of the chain (top row), and the corresponding change in the total energy (bottom row). Observe the difference in scaling of δH axis. 27

5.4 The plot on the left shows the points generated by the HMC algorithm from the 2-dimensional correlated Gaussian distribution. The right plot presents the desirable behavior of the point without random walks. 27

6.1 The contour plot (left), and 3-dimensional plot of the correlated Gaussian distribution defined in (4.39). 34

6.2 The error introduced in the Hamiltonian $\mathcal{H}$ for both Leapfrog and Störmer–Verlet (SV) scheme (left plot). The plot on the right shows the comparison of the theoretical and observed acceptance rate using different starting value for the fictive variable ρ_0. Here $P_{acc} = \text{erfc}(\frac{1}{2}\sqrt{< \delta\mathcal{H} >})$. 36

7.1 The figure shows how to compare two given chains. We are interested in the chain with less correlation between the elements (in red), with smallest τ_{int}. 39
7.2 The discrete power spectral density of uncorrelated (white noise) chain for the Gaussian distribution with mean zero and unit variance (left figure). The discrete power spectral density function of an MCMC chain ($N = 1000$) for 2D Gaussian centered at the origin and with identity covariance (right figure). 42
7.3 The power spectral density template for the MCMC chains, the curve is approximately power of two ($\alpha \approx 2$). The figure shows where the curve turnover is, before correlated sample are drawn. 43

8.1 The discrete power spectral density (psd) and fitted template for the chains generated by the OHMC and HMC algorithm in 64-dimensions (the left column), and 128-dimensions (the right column). 48
8.2 The integrated autocorrelation time and estimated value of τ_{int} for both OHMC and HMC algorithms are shown in the top row of the figure. The corresponding autocorrelation function for different lags is presented in the bottom row. 49
8.3 500 trajectory points in the phase–space (q, p) for 64–dimensions (left column) and 128–dimensions (right column) from OHMC (top row) and HMC (bottom row). 49
8.4 The histograms for OHMC (top row) and HMC (bottom plots) in 64 and 128–dimensions. 50
8.5 Discrete power spectral density (psd) in logarithmic scale (bottom row) with the best fit template (top row) for the chains generated by the SVHMC and HMC algorithm in 64 and 128–dimensions. 51
8.6 The integrated autocorrelation time τ_{int} and the autocorrelation function $\rho(\tau)$ for both SVHMC and HMC in 64 and 128–dimensions. 51
8.7 500 trajectory points in the phase–space (q, p) for 64-dimensions (left column) and 128-dimensions (right column) from SVHMC (top row) and HMC (bottom row). 52
8.8 The histograms for SVHMC (top row) and HMC (bottom plots) for 64 and 128–dimensions. 53
8.9 The power spectral density (psd) in logarithmic scale (bottom row), and approximated template curves defined by (7.18) for the chains generated by the SVHMC and HMC algorithm (top row), for 64-dimensions (left column) and 128-dimensions (right column). 54
8.10 The integrated autocorrelation time and estimated value of τ_{int} for both OHMC and HMC algorithms are shown in the upper row of the figure. The corresponding autocorrelation functions are presented in the bottom row of this figure. 54
8.11 500 trajectory points in the phase–space (q, p) for 64-dimensions (left column) and 128-dimensions (right column) for OSVHMC (top row) and SVHMC (bottom row). 55
8.12 The histograms for OSVHMC (top) and SVHMC (bottom) for 64 and 128–dimensions. 55

List of Tables

8.1 Summary of the power spectral and the degree of correlation convergence tests for OHMC and HMC algorithms in 64 and 128–dimensions. 48
8.2 Summary of diagnostics convergence tests for SVHMC and HMC algorithm in Gaussian distribution of 64 and 128–dimensions 52
8.3 The convergence diagnostics and the efficiency criteria for OSVHMC and SVHMC. 53

to my

parents ALTOMA and ALI, and to my wife SARA

with my love

Chapter 1

Overview

Some of the results from this thesis have been accepted as contributions to be presented at the following conferences:

(1) Efficient Monte Carlo: From Variance Reduction to Combinatorial Optimization A Conference on the Occasion of R.Y. Rubinstein's 70th Birthday. Sandbjerg Estate, Snderborg, Denmark 14-18 July 2008.

(2) 11th European Conference on the Mathematics of Oil Recovery. 8 - 11 September 2008, Bergen, Norway

The Abstracts for the conference papers are provided in appendix A and B.

1.1 Motivation

Classical optimization based on stochastic sampling algorithms (e.g., Metropolis–Hastings and Gibbs sampling algorithms) are usually slow, and could be inefficient sampling the parameter space, especially in high dimensions. These algorithms could be made more efficient by introducing gradient information.

The Hamiltonian Monte Carlo (HMC) algorithm is a Markov chain Monte Carlo (MCMC) technique, which alternately combines a Gibbs sampling update with a Metropolis rule. The HMC algorithm uses the advantages of the Hamiltonian dynamics to investigates the parameter space. The trajectory is guided by gradient information, and thus has advantages over the classical Metropolis-Hasting, and Gibbs sampling by having higher acceptance rate, less correlated and faster converging chains.

The performance of the HMC algorithm can be further enhanced by suppressing the random walks in the Gibbs sampling. Ordered over–relaxation has been suggested in the literature as a means of suppressing random walk behavior, and which is applicable to any system involving Gibbs sampling stage. However, this has not been applied to the HMC algorithm.

The dynamics in the HMC are simulated with constant step–size, which lead to extra computation cost when the dynamics of the system change along different regions of its trajectory. This extra computation cost can be reduce by using adaptive step–step to simulate the dynamics.

1.2 Goal of this thesis

The goal of this thesis is to present numerical experiments, which show that the performance of the HMC algorithm is enhanced when ordered over–relaxation is applied to the Gibbs sampling stage of the algorithm, and adaptive step–size in the Störmer–Verlet discretization scheme is used to simulate the Hamiltonian dynamics.

We first investigate the use of the ordered over–relaxation in the Gibbs sampling step and compare the performance of the resulting algorithm to the classical Hamiltonian Monte Carlo.

In the second step, we apply the Störmer–Verlet discretization with adaptive step–size to simulate the dynamics and compare the performance of this algorithm to the Hamiltonian Monte Carlo using the leapfrog scheme with constant step–size.

Finally, compare the performance of a hybrid algorithm consisting of the ordered over–relaxation in the Gibbs sampling, and Störmer–Verlet scheme with adaptive step–size, to the classical Hamiltonian Monte Carlo algorithm.

The performance of these algorithms will be evaluated by spectral analysis and degree of the correlation of chains from numerical experiments using uncorrelated Gaussian distributions.

Chapter 2

Inverse theory

The main purpose of this chapter is to introduce inverse problems and parameter estimation. Section 2.2 gives a brief description of classical approaches to solve inverse problems, section 2.3 gives the probabilistic approach to solve inverse problems, while section 2.5 introduces and motivates the use of Bayesian inference to solve inverse problems. The material of this chapter is mainly taken from [3, 23, 28, 46, 47].

2.1 Introduction to inverse problems

Most problems in science and engineering are characterized by finding relationship between the system parameters that characterize a model[1] m, and a collections of measurement observations (data) d. Suppose that the mapping $G : \mathcal{M} \to \mathcal{D}$, where $\mathcal{M}$, and $\mathcal{D}$ are the model (parameter) space and data space respectively, define the relation between m, and d as stated in equation (2.1)

$$G(m) = d, \tag{2.1a}$$

$$Gm = d. \tag{2.1b}$$

The operator G may be nonlinear or linear. If G is a nonlinear operator then we can interpret m, and d as functions (2.1a). On the other hand, in the case of linear operator for example, matrix $G \in \mathbb{R}^{m\times n}$, usually we interpret m, and d as vectors, i.e., $m \in \mathbb{R}^n$, and $d \in \mathbb{R}^m$. In the sense of the linear algebra the mathematical model (2.1b) is called a linear system.

The forward problem is to determine d given the model m, using (2.1). On the other hand the inverse problem is to compute, or estimate the unknown m based on noisy observations data d in (2.1).

While the forward problem has a unique solution, this is not the case in the inverse problem due to lack of significant data or due to experimental uncertainties [28, 46].

According to Hadamard [52], the mathematical problem is well–posedness if has the properties that

[1] Mathematician usually refer to m as the parameter, and for $G(m) = d$ as the mathematical model.

(1) A solution exist. i.e. $\exists \quad m \in \mathcal{M}$, such that (2.1) hold.

(2) The solution m is unique.

(3) The solution is stable with respect to perturbations in d.

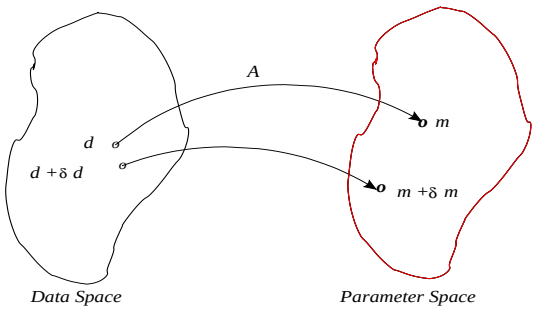


FIGURE 2.1: The figure shows how small error in the data lead to a different model. Ideally we expect that the error in model should be proportional to the data error. This is the typical features of the inverse problem. Here $A = G^{-1} : \mathcal{D} \to \mathcal{M}$, and δ is the uncertainty in the data.

A typical pathology of the inverse problems is that they are ill-posed, i.e. one of the Hadamard's postulates for well–posedness is not fulfilled. Because of these, in the inverse problem, one needs to make explicit any available a priori information on the model parameters. One also needs to be careful in the representation of the data uncertainties. Figure 2.1 shows a typical example of an ill–posed problem. Here a small error δ (uncertainty) in data d leads to large and disproportional error in estimation of the model m.

2.2 Deterministic approach to solving inverse problems

In the classical approach to solving inverse problem we begin with the mathematical model in (2.1), assume that there is a true model m_t and that we have exact measurement data d_t that satisfy (2.1). Then actual data d that we have is given in (2.2)

$$d + \delta d = G(m_t), \tag{2.2a}$$

$$d + \delta d = Gm_t, \tag{2.2b}$$

where δ is error of the measurements data. Assuming that the problem in (2.2) is well–posed (well–conditioned) and the error δ is independent and normally distributed $\delta \sim \mathcal{N}(0, \sigma_i)$, the maximum likelihood estimation (MLE) solution is equivalent to the least square (LS) solution [3], which is given by minimizing the square of the 2–norm of the residual,

$$m_{LS} = \arg\min \mathcal{F}(m) = \|G(m) - d\|_2^2, \tag{2.3a}$$

$$m_{LS} = \arg\min \mathcal{F}(m) = \|Gm - d\|_2^2. \tag{2.3b}$$

Note that the least square principle can be applied to both linear and nonlinear problems. In the linear problem if the matrix is full rank, the solution m_{LS} given by setting $\frac{\partial \mathcal{F}}{\partial m} = 0$ leads to solving the normal equations $G^T G m = G^T d$. Moreover if the Hessian matrix $G^T G$ of the objective function (2.3b) is positive definite, then m_{LS} is guaranteed to be a stationary point of $\mathcal{F}(m)$. On other hand, with nonlinear problems, the minimization task in (2.3a) is performed through numerical optimization techniques [37].

In many cases the assumption of the well–posedness (well–conditioned) is not fulfilled. In such situations, the least square solutions that approximately fit the data are large and diverse, and commonly contain many unreasonable models [3].

Regularization methods are remedy for ill–posedness by imposing stability on an ill-posed problem in a manner that yields accurate approximate solutions, often obtained through incorporating prior information [52]. There are a number of regularizing approaches to least square problems, which pick the best solution that approximately fits the data. One of these methods is the general Tikhonov regularization, where the goal is find minimizer m_α of the modified least square solution given by (2.4)

$$m_\alpha = \arg\min \mathcal{F}_\alpha(m) = \|G(m) - d\|_2^2 + \alpha \|A(d)\|_2^2, \tag{2.4a}$$

$$m_\alpha = \arg\min \mathcal{F}_\alpha(m) = \|Gm - d\|_2^2 + \alpha \|Ad\|_2^2, \tag{2.4b}$$

where $\alpha > 0$ is a regularization parameter, and A is the penalty functional. The zero–order Tikhonov regularization is given by taking the penalty functional to be the identity functional. An important issue in Tikhonov methods is how to choose the regularization parameter α. There are many methods for the choice of regularization parameter α, such as the discrepancy principle, and the generalized cross validation [3].

For linear problems the computation of the regularized solution is generally done with help of the singular value decomposition (SVD) [3]. In many applications the optimization problem (2.4a) and (2.3a) may contain additional constraints such as inequality constraints. Nonlinear least square problem may have a large number of local minimum solutions, and finding the global minimum can be extremely difficult. Moreover regularization introduces bias in the solution. This means that the derived solution is not necessarily the true solution [3].

The regularizing functional incorporates a priori information about the desired model m into the least square problem. The interpretation and choice of the penalty functional as additional a priori information, which is based on the knowledge about the desired model m [23], is emphasized especially in statistical approach of the inverse problems.

2.3 Statistical inversion

The most general theory is obtained when using a probabilistic point of view, where the a priori information on the model parameters is represented by a probability distribution over the model space. The general theory has a simple probabilistic formulation and applies to any kind of inverse problem, including linear as well as strongly nonlinear problems [46, 47].

The Bayesian approach is based on a philosophy different from the deterministic approach in section 2.2. The most fundamental difference between the deterministic and the Bayesian approaches is in the nature of the solution. In the deterministic approach we seek a single unknown model m. In the Bayesian approach the solution is a probability distribution for the model parameters [3]. For further discussion on statistical inversion the following definitions are necessary.

2.4 Definitions

The mathematical theory of probability begins with an experiment, which produces a set of possible outcome $\mathcal{S}$. We are interested in event $\mathcal{A} \subset \mathcal{S}$.

Definition 2.1. The probability function p, where $p : S \to [0, 1]$ has the following properties:

(1) $Pr(\mathcal{S}) = 0$.

(2) $Pr(\mathcal{A}) \geq 0$, where $\mathcal{A} \subset \mathcal{S}$.

(3) Let $\mathcal{A}_i \subset \mathcal{S}$; $i = 1, 2, \ldots,$ and $\mathcal{A}_i \bigcap \mathcal{A}_j = \phi$; $i \neq j$. then

$$Pr(\bigcup_{i=1}^{\infty} \mathcal{A}_i) = \sum_{i=1}^{\infty} Pr(\mathcal{A}_i), \tag{2.5}$$

$Pr(\mathcal{A})$ denotes the probability of the event $\mathcal{A}$. In practice, the outcome of an experiment is often a number rather than an event. Now let $\mathcal{S} = \mathbb{R}$.

Definition 2.2. A random variable X is a function that assigns a value for each outcome in the sample space $\mathcal{S}$. Then the relative probability of realization values for a random variable X, can be describe by a non–negative probability distribution (density) function (PDF), $p(x)$, defined by (2.6)

$$Pr(X \leq x_0) = \int_{-\infty}^{x_0} p(x)dx. \tag{2.6}$$

From definition (2.1) the probability density function satisfy (2.7)

$$\int_{-\infty}^{\infty} p(t)dt = 1. \tag{2.7}$$

The cumulative distribution function $F(x)$ (CDF) for the random variable X is given by the definite integral of the associated PDF (2.8)

$$F(x) = \int_{-\infty}^{x} p(t)dt. \tag{2.8}$$

Definition 2.3. Expected value and the variance

(1) The expected value of a random variable X, denoted by $E[X]$, or μ is

$$E[X] = \int_{-\infty}^{\infty} xp(x)dx. \tag{2.9}$$

In general if $q(X)$ is a function of random variable X, then

$$E[q(X)] = \int_{-\infty}^{\infty} q(x)p(x)dx. \tag{2.10}$$

(2) The variance of X, denoted by $Var(X)$ or σ^2, is given by

$$Var(X) = E[(X - E[X])^2], \tag{2.11}$$

$$= \int_{-\infty}^{\infty} (x - E[X])^2 p(x)dx. \tag{2.12}$$

The standard deviation of X defined by $\sigma = \sqrt{Var(X)}$.

Similarly to one dimensional case, we generalize the same results to higher dimensions.

Definition 2.4. The joint probability distribution function of two variables X and Y, is denoted by $p(X, Y)$ and satisfies (2.13)

$$Pr(X \leq x, Y \leq y) = \int_{-\infty}^{x} \int_{-\infty}^{y} p(x, y)dxdy. \tag{2.13}$$

The two random variables are independent if their joint probability distribution satisfies (2.14)

$$p(x, y) = p_X(x)p_Y(y), \tag{2.14}$$

where $p_X(x)$ and $p_Y(y)$, is the marginal density for the random variable X and Y, respectively, and are defined by

$$p_X(x) = \int_{-\infty}^{\infty} p(x, y)dy,$$

$$p_Y(y) = \int_{-\infty}^{\infty} p(x, y)dx.$$

Definition 2.5. The conditional probability of X, given $Y = y$ is defined by (2.15)

$$p(x \,|\, y) = \frac{p(x, y)}{p_Y(y)}, \tag{2.15}$$

By replacing the role of x and y, we have $p(x, y) = p(y \,|\, x)p_X(x)$, and plugging this again in (2.15) gives Bayes' theorem

$$p(x \,|\, y) = \frac{p(y \,|\, x)p_X(x)}{p_Y(y)}. \tag{2.16}$$

2.5 Bayesian model for inverse problems

In the Bayesian approach to inverse problems, we model the variables m, d and e in (2.2) as random variables. The interpretation of this modelling is that the information concerning their values is incomplete, due to lack of significant data or due to experimental uncertainties. Their values are thus expressed by their distribution as random variables.

Bayes theorem provides the ultimate means of statistical inference on the basis of observations, permits the use of arbitrary probability distributions other than Gaussian, and the use of arbitrary measures of uncertainty other than the variance. It also extends the analysis to higher levels of interpretation, e.g., the rejection of any particular model, and the selection of appropriate models [16].

Assume that $p(m)$ is prior probability distribution function (pdf), expressing our prior belief about m. The conditional probability distribution $p(d \,|\, m)$ is the likelihood density, expressing the probability distribution of the measurement observation d if we assume that our model is known. For simplicity, we assume that the model m is parameterized in terms of a vector, $m \in \mathbb{R}^k$, of k variables, $m_1, \ldots, m_k$. The theorem allows update of the prior belief by calculating a posterior pdf, $p(m \,|\, d)$, given by equation (2.17),

$$p(m \,|\, d) \quad = \quad \frac{p(d \,|\, m)p(m)}{\int_{\mathbb{R}^k} p(d \,|\, m)p(m)dm}. \tag{2.17}$$

The posterior distribution provides the basis for inference about, e.g., the marginal probability of a parameter m_i, given all other parameters. This is defined as the integral of the posterior probability over the remaining dimensions of the parameter space, i.e.,

$$p(m_i \,|\, m_j; m_j \in \{m \backslash m_i\}) \quad = \quad \int \cdots \int p(m \,|\, d) \prod_{\substack{k=1 \\ k \neq i}}^{m} dm_k. \tag{2.18}$$

A similar expression, $p(m_i, m_j \,|\, m_k; m_k \in \{m \backslash m_i, m_j\})$, could be derived for two-parameter interactions.

Evaluating (2.18) requires computing generic Bayesian integral, $\langle J \rangle$, in (2.19).

$$\langle J \rangle = \int_{\mathbb{R}^k} J(m) p(m \,|\, d) dm, \tag{2.19}$$

where $J(m)$ represents, e.g., the distribution of model parameters. The function $J(m)$ could also represent other derived quantities. Deriving $\langle J \rangle$ in (2.19) is conditioned on evaluating the normalizing constant in (2.17), i.e., $\int_{\mathbb{R}^k} p(d \,|\, m) p(m) dm$. This integral could be intractable even for very low values of k. A Monte Carlo approach offers an attractive methodology for evaluation (2.17) without explicit evaluation of the normalizing constant.

A Monte Carlo rendering of (2.19) is given by (2.20), where n is the number of indexed models in an ensemble.

$$\langle J \rangle = (1/n) \sum_{j=1}^{n} J(m_j) p(m_j \,|\, d) / h(m_j), \tag{2.20}$$

$$\approx (1/n) \sum_{j=1}^{n} J(m_j) \;\text{ for }\; h(m_j) \approx p(m_j \,|\, d), \tag{2.21}$$

$$m_j \equiv m(m_j). \tag{2.22}$$

Thus if one derives an ensemble of models, $m_1, m_2, \ldots, m_n$, whose distribution $h(m)$, approximates $p(m|d)$, the task of evaluating the generic integrals reduces to calculating averages over $J(m_j)$.

Markov chain Monte Carlo (MCMC) methods are based on Markov chains, which generate samples from target distributions, such as $p(m \,|\, d)$. The Metropolis-Hastings algorithm is perhaps the most popular of all implementation of MCMC algorithms. Chapter 3 will be devoted to the MCMC algorithms.

Chapter 3

Markov Chain Monte Carlo algorithms

This chapter will review some classical Markov chain Monte Carlo (MCMC) methods. The chapter starts with an introduction to basic concepts and definitions. See for example [11, 26, 32, 39] for more details concerning MCMC algorithms and related concepts.

3.1 Basic definitions

Definition 3.1. A stochastic process is a consecutive set of random variables $\{X_t \,|\, t \in T\}$, where T is any indexed set.

The common choice for the indexed set can be discrete $T = \{0, 1, 2, \dots\}$, where the stochastic process might describe e.g. outcome of successive tosses of coin in a learning experiment, or continuous such as $T = [0, \infty)$ where for example the stochastic process can represent the number of cars in the period of length t, i.e. in $[0, t]$ along a highway.

Definition 3.2. A Markov chain is stochastic process such that the distribution of X_{t+1} given the all the previous states $X_0, \dots, X_{t-1}, X_t$ depend only on the previous one X_t, i.e. the state is conditionally independent. This can be state more formally expressed by (3.1).

$$\begin{aligned} Pr(X_{t+1} \,|\, X_0, X_1, \dots, X_t) &= Pr(X_{t+1} \,|\, X_t), \\ &= T(X_{t+1}, X_t), \end{aligned} \tag{3.1}$$

and referred to as the *Markov property*. the distribution of X_{t+1}; $t \geq 0$, $T(X_{t+1} = j \,|\, X_t = i)$, is called the *transition kernel*. Here we assume that the Markov chain is *time-homogeneous*, i.e the transition kernel does not depend on the time t. Given the initial probability $Pr(X_0) = \pi_0(x)$ and using the transition kernel we can determine the behavior of the chain at any time by the following recursion using so called the first step analysis [48]

$$\pi_{t+1}(x) = \sum_{x'} \pi_t(x') T(x', x).$$

We are interested in a Markov chain that starting from any state X_0 with initial probability $\pi_0(x)$ will eventually forget its initial state and converge to a *stationary* distribution $\pi(x)$. As the sequence grows larger, such Markov chains must satisfy the following properties

(1) **Stationarity**: As $t \to \infty$ the chain converges to its stationary (invariant or equilibrium) distribution.

(2) **Irreducibility**: which means that there is a positive probability that the Markov chain can reach any non-empty set of states from all starting point.

(3) **Aperiodicity**: ensures that the chain will not oscillate between different sets of states.

Definition 3.3. If the Markov chain satisfy the stationarity, irreducibility, and aperiodicity we call it *ergodic*.

Ergodicity is sufficient condition for existence of the stationary distribution $\pi(x)$ independent of the initial probability at the starting state.

Definition 3.4. The sufficient, but not necessary condition for a distribution $\pi(x)$ to be stationary distribution is that $\pi(x)$ should satisfy the *detailed balance* equation (*reversibility* condition)

$$\pi(x)T(x, x') = \pi(x')T(x', x). \tag{3.2}$$

Definition 3.5. A Markov Chain Monte Carlo (MCMC) method for simulation of a distribution π is an any method producing an ergodic Markov Chain $\{X_t\}_{t=1}^{n}$ who's stationary distribution is π.

3.2 Simulation of the Markov chains

The main idea behind MCMC algorithms is to generate a Markov chain $\{X_i\}_{i=1}^{t}$ with probability transition kernel $T(x, y)$ that has stationary distribution $\pi(x)$ in the long run, independent of where the chain starts. However, this is not the case for all Markov chains. The sufficient condition for converging to stationary distribution is that the chain is ergodic. The Gibbs sampling and Metropolis–Hastings algorithms are two classical approaches for implementing MCMC algorithms.

3.2.1 The Gibbs sampler algorithm

The Gibbs sampler generates samples, $X_1, X_2, \ldots, X_n$ from $\pi(x_1, \ldots, x_d)$, and then uses these samples to estimate the desired statistical properties. Generating large number of sample improves the estimation accuracy. Algorithm 3.1 is the pseudo-code for the Gibbs sampler.

The Gibbs sampler always accept the candidate point, also the Gibbs sampling algorithm is restricted to distribution where the full conditional distributions is know and this makes it less applicable in practice [27].

Algorithm 3.1: The Gibbs Sampler

Set $t = 0$
Generate a starting point $X_t = (x_1^{(t)}, \ldots, x_d^{(t)})$
for $t = 1$ *to* n **do**
    Generate $x_1^{(t+1)} \sim \pi_1(x_1 \,|\, x_2^{(t)}, \ldots, x_d^{(t)})$
    Generate $x_2^{(t+1)} \sim \pi_2(x_2 \,|\, x_1^{(t+1)}, x_3^{(t)}, \ldots, x_d^{(t)})$
    $\vdots$
    Generate $x_d^{(t+1)} \sim \pi_d(x_d \,|\, x_1^{(t+1)}, \ldots, x_d^{(t+1)})$
    $X_{t+1} = (x_1^{(t+1)}, \ldots, x_d^{(t+1)})$
end

To illustrate the Gibbs sampler, suppose we want to estimate the marginal distribution of $x1$, i.e. compute $\pi(x_1)$ for the following joint distribution, which is picked from [39]

$$\pi(x_1, x_2) = \binom{n}{x_1} x_2^{x_1+\alpha-1}(1-x_2)^{n-x_1-\beta-1},$$

where $x_1 = 0, 1, \ldots, n$, and $x_2 \in [0, 1]$, by considering n, α, and β as a fixed parameters. The conditional distributions $\pi(x_1 \,|\, x2)$, and $\pi(x_2 \,|\, x_1)$ follow the binomial distribution with parameters n and x_2, and beta distribution with parameters $x_1 + \alpha$ and $n - x_1 + \beta$ respectively. $\pi(x_1)$ can be estimated by (3.3).

$$\pi(x_1) \approx \frac{1}{n-m} \sum_{t=m+1}^{n} \pi(x_1 \,|\, X_t). \tag{3.3}$$

The resulting marginal distribution is shown in Figure 3.1.

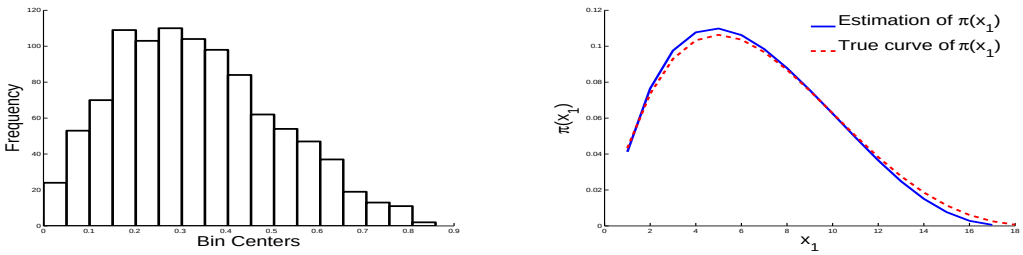


FIGURE 3.1: The histogram of the chain of 1000 samples generated according to Gibbs sampler Algorithm 3.1 is illustrated on the left, and the estimated marginal distribution for x_1 from equation (3.3) on the right of the plot, when the parameters $\alpha = 2$, and $\beta = 4$.

3.2.2 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is one of the best known MCMC methods, which was developed by [31] and generalized by [18]. Metropolis-Hasting algorithm was frequently used in physics literature until the papers published by [29, 49], and later introduced to statistic literature and many other fields.

The simplest Metropolis-Hastings algorithm to generate samples from the target distribution $\pi(x)$ work as follow. At step t generate a candidate state x'_t from the proposal distribution $q(. \mid x'_t)$ that have the same property as the target distribution. One convenient possibility is the normal distribution centered at the current state x'_t in the simulation (the mean), and fixed covariance matrix. One thing to consider when selecting $q(. \mid x'_t)$ is that the proposal distribution should be easy to sample from. The proposal distribution must satisfy the irreducibility and aperiodicity conditions this is achieved if the proposal distribution has a positive density on the same support as the target distribution. The candidate state is accepted or rejected for the next state in the chain with probabilities given by

$$x_{t+1} = \begin{cases} x'_t, & \text{with probability} \quad \alpha(x_t, x'_t), \\ x_t, & \text{with probability} \quad 1 - \alpha(x_t, x'_t), \end{cases}$$

where

$$\alpha(x_t, x'_t) = \min\left(\frac{q(x_t \mid x'_t)\pi(x'_t)}{q(x'_t \mid x_t)\pi(x_t)}, 1\right).$$

Algorithm 3.2: The Metropolis-Hastings

```
Generate starting point x_0
for t = 1 to n do
    Generate a candidate x'_t ~ q(. | x_{t-1})
    Generate u ~ U[0,1]
    Compute r(x_t, x'_t) = q(x_t | x'_t)π(x'_t) / q(x'_t | x_t)π(x_t)
    if u < min(r(x_t, x'_t), 1) then
        x_t = x'_t
    else
        x_t = x_{t-1}
    end
end
```

If the candidate state is not accepted, then the Metropolis-Hastings chain will not move i.e. $x_{t+1} = x_t$. If our target distribution appears with normalizing constant that we do not know, then in computing the ratio, $r(x_t, x'_t) = \frac{q(x_t \mid x'_t)\pi(x'_t)}{q(x'_t \mid x_t)\pi(x_t)}$ the normalizing constant will cancel out,

which gives one of the characteristics of Metropolis-Hastings algorithm. Algorithm 3.2 is the pseudo-code for the Metropolis-Hastings algorithm[1]

Figure 3.2 shows an example of simulating the Cauchy distribution (3.4) using the Metropolis-Hastings algorithm (3.2).

$$\pi(x) = \frac{1}{\pi(x^2+1)}, \qquad x \in \mathbb{R}. \tag{3.4}$$

In this example we use the Gaussian distribution as a proposal distribution with optimal variance $\sigma_T = 8.4$

$$q(x) = \frac{1}{\sqrt{2\pi}\sigma_T} \exp\left(-\frac{(x-\mu)^2}{2\sigma_T^2} \right), \qquad x \in \mathbb{R}. \tag{3.5}$$

The Metropolis-Hastings algorithm satisfied the detailed balance equation (3.2) [53].

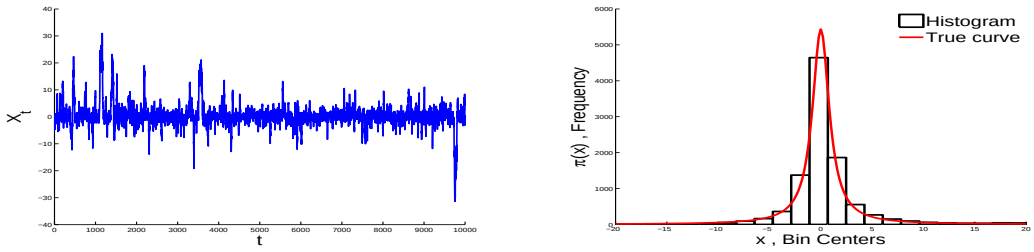


FIGURE 3.2: 10000 sample form the Cauchy distribution left plot. The histogram and the Cauchy distribution curve drawn in the same plot (the right plot), which gives an acceptable match.

3.3 Classical MCMC in high dimensions

The classical MCMC algorithms are very simple and easy to implement. However, in high dimension, they are very slow in generating samples from the target distribution, and require a huge number of samples in order to give a good accuracy for estimating the desired statistical properties [11, 39].

Classical MCMC algorithms follow random walk behavior to investigate the target distribution.They have the property that in k iterations the simulation will usually moves distance proportional to $\sqrt{k}$ [32].

The efficiency of any MCMC algorithm is defined as reciprocal of the number of samples needed to effectively provide a statistically independent samples from the desired target distribution.

[1] this type of the algorithm is called the Global Metropolis-Hastings, in which all components of the vector x_t are updated. Perferably the proposal distribution should be spherically and symmetric [32]. The other type is called the local Metropolis-Hastings, where the update happens in one component of x_t in turn, and we can use any suitable proposal distribution.

The efficiency of the Metropolis-Hastings with optimal choice of the proposal distribution drop as $\frac{0.33}{D}$ [9, 10, 15], where D is the dimension of the target distribution. In high dimensions the Metropolis-Hastings is unsuitable.

Chapter 4

Hamiltonian Monte Carlo algorithm

This chapter introduces Hamiltonian Monte Carlo algorithm. The algorithm was originally developed to model physical system and have had wide applications in molecular dynamics. the chapter starts with definitions of the main components of the algorithm which are derived from the classical mechanics. It then discusses the implementation of the algorithm as a method for parameter estimation. The material in this chapter is mainly from [13, 40, 51].

4.1 Definitions

Definition 4.1. Generalized Coordinates
A set of n independent coordinates $q_1(\tau), q_2(\tau), \ldots, q_n(\tau)$ which exactly determine a system in a fixed time τ are called generalized coordinates, their derivatives $\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n$ with respect to τ are referred to generalized velocities, even through they do not necessarily have dimensions of length [51], e.g. q_j may be an angle.

Now let $\mathbf{r}_i = \mathbf{r}_i(q_1(\tau), q_2(\tau), \ldots, q_n(\tau), \tau)$ be the position vector at time τ for particle i of a system containing N particles. Then

$$\begin{aligned}
\dot{\mathbf{r}}_i &= \frac{d\mathbf{r}_i}{d\tau}, \\
&= \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j}\frac{dq_j}{d\tau} + \frac{\partial \mathbf{r}_i}{\partial \tau}, \qquad (4.1) \\
&= \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j}\dot{q}_j + \frac{\partial \mathbf{r}_i}{\partial \tau}. \qquad (4.2)
\end{aligned}$$

Equation (4.2) defines the position vector of the system velocity. From the expression of the velocity $\dot{\mathbf{r}}_i$ of particle i, one can derive two relations. First, take the partial derivative with respect to time derivatives of the generalized coordinates $\dot{q}_k$, and by noting that the last term in

(4.2) will vanish immediately, and then we arrive at (4.4)

$$\begin{aligned}\frac{\partial \dot{\mathbf{r}}_i}{\partial \dot{q}_k}, &= \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j}\frac{\partial \dot{q}_j}{\partial \dot{q}_k} + \frac{\partial}{\partial \dot{q}_k}\frac{\partial \mathbf{r}_i}{\partial \tau}, \\ &= \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j}\delta_{jk}, \qquad (4.3) \\ &= \frac{\partial \mathbf{r}_i}{\partial q_k}, \qquad (4.4)\end{aligned}$$

where δ_{jk} is the Dirac delta function. Secondly, take the partial derivative with respect to the generalized coordinates q_k to arrive at (4.5), which shows that the role of $\frac{d}{dt}$ and $\frac{\partial}{\partial x_k}$ are interchangeable.

$$\begin{aligned}\frac{d}{d\tau}\left(\frac{\partial \mathbf{r}_i}{\partial q_k}\right), &= \sum_{j=1}^{n} \frac{\partial}{\partial q_j}\frac{\partial \mathbf{r}_i}{\partial q_k}\dot{q}_j + \frac{\partial}{\partial \tau}\frac{\partial \mathbf{r}_i}{\partial q_k}, \\ &= \sum_{j=1}^{n} \frac{\partial}{\partial q_k}\frac{\partial \mathbf{r}_i}{\partial q_j}\dot{q}_j + \frac{\partial}{\partial q_k}\frac{\partial \mathbf{r}_i}{\partial \tau}, \\ &= \frac{\partial}{\partial q_k}\left(\sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j}\dot{q}_j + \frac{\partial \mathbf{r}_i}{\partial \tau}\right), \\ &= \frac{\partial \dot{\mathbf{r}}_i}{\partial q_k}. \qquad (4.5)\end{aligned}$$

Now suppose that the system of the particles is moving, and we want to find the trajectory of the system particles. Let $\mathbf{F}_i$, and m_i be the applied force and the mass of particle i. According to Netown laws of motion we arrive at (4.6)

$$\mathbf{F}_i = m_i \frac{d}{d\tau}\left(\frac{\partial \mathbf{r}_i}{\partial \tau}\right), \qquad (4.6)$$

where the quantity $\frac{d}{d\tau}(\frac{\partial \mathbf{r}_i}{\partial \tau})$ is the acceleration of particle i.

Definition 4.2. The generalized force Q_j of the generalized coordinate j is defined by (4.7)

$$Q_j = \sum_{i=1}^{N} \mathbf{F}_i \frac{\partial \mathbf{r}_i}{\partial q_j}. \qquad (4.7)$$

Then using (4.6) and the definition of the generalized force in (4.2) we get (4.8)

$$Q_j = \sum_{i=1}^{N} m_i \frac{d}{d\tau}\left(\frac{\partial \mathbf{r}_i}{\partial \tau}\right).\frac{\partial \mathbf{r}_i}{\partial q_j}. \qquad (4.8)$$

Using the product rule of the derivatives, $\dot{u}v = (uv)^{\cdot} - u\dot{v}$ on the left hand side of (4.8) and plugging equations (4.4) and (4.5) in the resulting equation (4.9)

$$Q_j = \sum_{i=1}^{N} m_i \left[\frac{d}{d\tau}\left(\frac{\partial \mathbf{r}_i}{\partial \tau}.\frac{\partial \mathbf{r}_i}{\partial q_j} \right) - \frac{\partial \mathbf{r}_i}{\partial \tau}.\frac{d}{d\tau}\left(\frac{\partial \mathbf{r}_i}{\partial q_j} \right) \right], \tag{4.9}$$

$$= \sum_{i=1}^{N} m_i \left[\frac{d}{d\tau}\left(\dot{\mathbf{r}}_i.\frac{\partial \dot{\mathbf{r}}_i}{\partial \dot{q}_j} \right) - \dot{\mathbf{r}}_i.\frac{\partial \dot{\mathbf{r}}_i}{\partial q_j} \right],$$

$$= \sum_{i=1}^{N} m_i \left[\frac{1}{2}\frac{d}{d\tau}\frac{\partial}{\partial \dot{q}_j}\left(\dot{\mathbf{r}}_i.\dot{\mathbf{r}}_i \right) - \frac{1}{2}\frac{\partial}{\partial q_j}\left(\dot{\mathbf{r}}_i.\dot{\mathbf{r}}_i \right) \right],$$

$$= \frac{d}{d\tau}\frac{\partial}{\partial \dot{q}_j}\left(\sum_{i=1}^{N} \frac{1}{2} m_i \left(\dot{\mathbf{r}}_i.\dot{\mathbf{r}}_i \right) \right) - \frac{\partial}{\partial q_j}\left(\sum_{i=1}^{N} \frac{1}{2} m_i \left(\dot{\mathbf{r}}_i.\dot{\mathbf{r}}_i \right) \right). \tag{4.10}$$

Definition 4.3. The kinetic energy, T, for a system of N particles is defined by (4.11)

$$T = \sum_{i=1}^{N} \frac{1}{2} m_i \left(\dot{\mathbf{r}}_i.\dot{\mathbf{r}}_i \right). \tag{4.11}$$

The **Lagrange equations** (4.12) is given by substituting (4.11) in (4.10), and consists of n second order differential equations in n variables q_j. Recall that no assumptions have been made concerning the nature of the generalized forces Q_j's, thus equations (4.12) can be applied to both conservative and non–conservative systems.

$$Q_j = \frac{d}{d\tau}\frac{\partial}{\partial \dot{q}_j} T - \frac{\partial}{\partial q_j} T; \qquad j = 1, \ldots, n. \tag{4.12}$$

Now suppose that the system is conservative, then $\mathbf{F}_i$ can be expressed as the gradient of a scalar potential energy $V(\mathbf{r}_i, \tau)$ [51]

$$\mathbf{F}_i = -\nabla V.$$

The potential energy is function of $\mathbf{r}_i$, and therefore function of q_j. Taking the derivative of V with respect to q_j, we may write (4.13)

$$-\frac{\partial V}{\partial q_j} = -\sum_{i=1}^{N} \frac{\partial V}{\partial \mathbf{r}_i}\frac{\partial \mathbf{r}_i}{\partial q_j}, \tag{4.13}$$

$$= -\sum_{i=1}^{N} \nabla V.\frac{\partial \mathbf{r}_i}{\partial q_j}, \tag{4.14}$$

$$= Q_j. \tag{4.15}$$

Thus the Lagrange equations for a conservative system are given in (4.16). By substitute (4.15) into (4.12) and noting that the potential energy function $V(\mathbf{r}_i, \tau)$ is not a function of $\dot{\mathbf{r}}_i$, i.e. V

is independent of the generalized velocity $\dot{q}_j$, therefore $\frac{d}{d\tau}\frac{\partial V}{\partial \dot{\mathbf{q}}_j} = 0$,

$$\frac{d}{d\tau}\frac{\partial}{\partial \dot{q}_j}(T-V) - \frac{\partial}{\partial q_j}(T-V) = 0; \qquad j = 1, \ldots, n. \tag{4.16}$$

where $\mathcal{L}(q_j, \dot{q}_j, \tau) = T - V$ is called the **Lagrangian** or the Lagrange function.

$$\frac{\partial \mathcal{L}}{\partial q_j} = \frac{d}{d\tau}\frac{\partial \mathcal{L}}{\partial \dot{q}_j}; \qquad j = 1, \ldots, n. \tag{4.17}$$

4.2 The Hamiltonian equations

Hamiltonian mechanics provide re–formulation of the classical mechanics [13, 40]. A knowledge of the Hamilton's formalism is fundamental to understand statistical mechanics, geometrical optics and quantum mechanics [40].

For an introduction, let f be a function of two variables, $f = f(x, y)$ and set $u = \frac{\partial f}{\partial x}$, $v = \frac{\partial f}{\partial y}$, and define $g(u, y) = f - ux$. The differentials $df(x, y)$, and $dg(u, y)$ is given in (4.18) and (4.19)

$$df(x, y) = udx + vdy, \tag{4.18}$$

$$dg(u, y) = -xdu + vdy. \tag{4.19}$$

Thus, $\frac{\partial g}{\partial u} = -x$ and $\frac{\partial g}{\partial y} = v$. $g(u, y)$ is called the Legendre transformation of f with respect to x [13]. The inverse Legendre transformation given by (4.20).

$$f = g + udx, \tag{4.20}$$

Definition 4.4. The generalized momenta p_j are defined as

$$p_j = \frac{\partial \mathcal{L}}{\partial \dot{q}_j}. \tag{4.21}$$

Definition 4.5. The Hamiltonian $\mathcal{H}$ is defined by the negative Legendre transform of the Lagrangian $\mathcal{L}(q_j, \dot{q}_j, \tau)$ with respect to all generalized velocities

$$\mathcal{H}(q_j, p_j, \tau) = -\mathcal{L} + \sum_{j=1}^{n} p_j \dot{q}_j. \tag{4.22}$$

Thus the Hamiltonian $\mathcal{H}$ is a function of generalized coordinates q_j, generalized momenta p_j, and the time τ.

The total derivative of $\mathcal{H}$ can be stated into two different ways, as in (4.23), or (4.24).

$$d\mathcal{H} = \sum_{j=1}^{n} \frac{\partial \mathcal{H}}{\partial q_j} dq_j + \sum_{j=1}^{n} \frac{\partial \mathcal{H}}{\partial p_j} dp_j + \frac{\partial \mathcal{H}}{\partial \tau} d\tau, \tag{4.23}$$

$$d\mathcal{H} = -\sum_{j=1}^{n} \frac{\partial \mathcal{L}}{\partial q_j} dq_j - \sum_{j=1}^{n} \frac{\partial \mathcal{H}}{\partial \dot{q}_j} d\dot{q}_j - \frac{\partial \mathcal{L}}{\partial \tau} d\tau + \sum_{j=1}^{n} \dot{q}_j dp_j + \sum_{j=1}^{n} p_j d\dot{q}_j, \tag{4.24}$$

$$= -\sum_{j=1}^{n} \frac{\partial \mathcal{L}}{\partial q_j} dq_j + \sum_{j=1}^{n} \dot{q}_j dp_j - \frac{\partial \mathcal{L}}{\partial \tau} d\tau. \tag{4.25}$$

Now comparing (4.23) with (4.25) and using Lagrange equations (4.17) we arrive at the equations of motion of Hamiltonian mechanics, known as the canonical equations of Hamilton, or **Hamiltonian equations** (4.26).

$$\frac{dp_j}{d\tau} = -\frac{\partial \mathcal{H}}{\partial q_j}, \tag{4.26a}$$

$$\frac{dq_j}{d\tau} = \frac{\partial \mathcal{H}}{\partial p_j}, \tag{4.26b}$$

$$\frac{d\mathcal{H}}{d\tau} = -\frac{\partial \mathcal{L}}{\partial \tau}. \tag{4.26c}$$

The system of the equation in (4.26) define the dynamics on the system phase–space, in which the q_i and p_i are regarded as function of time τ [26, 32].

Definition 4.6. The **phase–space** of a system consists of all possible values of the generalized coordinate variables q_j's and the generalized momenta variables p_j's.

The total energy function for a point (p, q) in the phase–space, i.e. the sum of the kinetic energy $T(p)$ and the potential energy $V(q)$, is the Hamiltonian $\mathcal{H}$ for a conservative system [13].

$$\mathcal{H}(p, q) = V(q) + T(p). \tag{4.27}$$

Note that in (4.27) we use the notation for the vectors $p = (p_1, p_2, \ldots, p_n)^T$, and $q = (q_1, q_2, \ldots, q_n)^T$, which gives the Hamiltonian of the system, i.e. all the particles in the system. Now in the rest of the section we will discuss the Hamiltonian properties.

Firstly, the Hamiltonian $\mathcal{H}$ is conserved as q_j, and p_j evolve through the time τ according to the dynamics defined by (4.26). It can be shown that

$$\begin{aligned}
\frac{d\mathcal{H}}{d\tau} &= \sum_{j=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial q_j} \frac{dq_j}{d\tau} + \frac{\partial \mathcal{H}}{\partial p_j} \frac{dp_j}{d\tau} \right), \\
&= \sum_{j=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial q_j} \frac{\partial \mathcal{H}}{\partial p_j} - \frac{\partial \mathcal{H}}{\partial p_j} \frac{\partial \mathcal{H}}{\partial q_j} \right), \\
&= 0.
\end{aligned}$$

Secondly, the Hamiltonian dynamics (4.26) remain invariant under the transformation defined by (4.28). Thus classical mechanics is invariant to the direction of time, i.e. time reversibility.

$$p' = p, \tag{4.28a}$$
$$q' = -q, \tag{4.28b}$$
$$\tau' = -\tau. \tag{4.28c}$$

Finally the dynamics preserves the volume of region of phase–space. i.e. if one take an arbitrary time τ_0, and arbitrary region of phase–space $\mathcal{R}_0$. For simplicity assume this time to be zero, $\tau_0 = 0$. Let $\mathcal{R}_\tau$ be the region of phase–space at time τ occupied by the points that were at time zero. The volume $\mathcal{V}(\tau)$ which is given as follow

$$\mathcal{V}(\tau) = \int_{\mathcal{R}_\tau} dq'dp', \tag{4.29}$$
$$= \int_{\mathcal{R}_0} det(I + \tau J)dqdp + \mathcal{O}(\tau), \tag{4.30}$$

where to get from (4.29) to (4.30) we used the transformations (4.31).

$$q' = q + \tau\dot{q} + \mathcal{O}(\tau), \tag{4.31a}$$
$$p' = p + \tau\dot{p} + \mathcal{O}(\tau), \tag{4.31b}$$

$I + \tau J$ is Jacobian matrix, J is part of the Jacobian with derivatives of $\dot{q}$ and $\dot{p}$ and $\mathcal{O}(\tau)$ holds as $\tau \to 0$. Since

$$\begin{aligned} det(I + \tau J) &= 1 + \tau trace(J) + \mathcal{O}(\tau), \\ trace(J) &= \frac{\partial}{\partial q}(\dot{q}) + \frac{\partial}{\partial p}(\dot{p}), \\ &= \frac{\partial^2 \mathcal{H}}{\partial q \partial p} - \frac{\partial^2 \mathcal{H}}{\partial p \partial q}, \\ &= 0, \end{aligned}$$

$$\begin{aligned} \mathcal{V}(\tau) &= \int_{\mathcal{R}_0} det(I)dqdp + \mathcal{O}(\tau), \\ &= \mathcal{V}(0) + \mathcal{O}(\tau). \end{aligned}$$

since τ_0 was chosen arbitrary, the derivative is zero for all times, i.e., the volume is constant and does not depend on the time. This is known as Liouville's theorem [13, 40]. In other words , if each of the points within th volume are considered as different systems within the same Hamiltonian, but different initial conditions, this points will spread all over, even outside the original volume, $\mathcal{V}$. However, the equations of motion guarantee that the volume of the phase–space covered, remain invariant with time.

4.3 Classical Hamiltonian Monte Carlo algorithm

The Hamiltonian Monte Carlo (HMC) [8], or Hybrid Monte Carlo algorithm is a Markov Chain Monte Carlo (MCMC) technique, which combines the advantages of Hamiltonian dynamics [2, 40] and Metropolis Monte Carlo approach [15, 31], to sample from complex distributions.

The first step in the dynamical approach involves augmenting the vector of parameters $q \in \mathbb{R}^n$, with a conjugate momentum vector $p \in \mathbb{R}^n$ [35], and the introduction of a Hamiltonian function $\mathcal{H}(q,p)$, defined on the phase–space (q,p). The Hamiltonian function is given by (4.27), where V and T are potential and kinetic energies, which are defined in (4.32) and (4.33) respectively, and $\pi(q)$ is the target distribution, here q playing the role of x in the previous chaoters. The HMC algorithm requires that we can efficiently calculate the derivatives of $-\log \pi(q)$, i.e., $\pi(q)$ must be continuous.

$$V(q) = -\log \pi(q), \tag{4.32}$$

$$T(p) = \frac{1}{2} p^T p. \tag{4.33}$$

The joint (canonical) distribution $P(q,p)$ over the phase–space defined by the Hamiltonian (4.27) is given in (4.34)

$$P(q,p) = \frac{1}{Z} \exp(-\mathcal{H}(q,p)), \tag{4.34}$$

$$= \left[\frac{(2\pi)^{\frac{n}{2}}}{Z} \exp(-V(q))\right] \left[\frac{1}{(2\pi)^{\frac{n}{2}}} \exp(-\frac{1}{2} p^T p)\right], \tag{4.35}$$

$$= \pi(q)\mathcal{N}(p; 0, I). \tag{4.36}$$

The first proposal in HMC algorithm is to randomize the momentum variables, and leaving the parameters q unchanged. This proposal can be viewed as Gibbs sampling update by drawing new momentum from the Gaussian density $\mathcal{N}(p; 0, I)$ [14, 26].

The second proposal change q and p by simulating the Hamiltonian dynamics in (4.26). Here the momentum variable guide the moving of the parameters q, and the gradient gives the change in p each time. In practice however, the Hamiltonian dynamics is simulated by the *leapfrog algorithm* with a finite step size, ϵ, according to

$$p(\tau + \frac{\epsilon}{2}) = p(\tau) - \frac{\epsilon}{2} \nabla V(q(\tau)), \tag{4.37a}$$

$$q(\tau + \epsilon) = q(\tau) + \epsilon p(\tau + \frac{\epsilon}{2}), \tag{4.37b}$$

$$p(\tau + \epsilon) = p(\tau + \frac{\epsilon}{2}) - \frac{\epsilon}{2} \nabla V(q(\tau + \epsilon)). \tag{4.37c}$$

Though each leapfrog transition is volume–preserving and time–reversible, finite ϵ does not keep $\mathcal{H}$ constant. Hence systematic error is introduced into the sampling procedure [32].

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 [11, 32], given by (4.38), where the current and candidate states are given by (q, p) and (q', p'), respectively. Algorithm 4.1 is a pseudo code for the implementation of the HMC algorithm using the leapfrog in the dynamics simulation.

$$\min\{1, \exp[-(\mathcal{H}(q', p') - \mathcal{H}(q, p))]\}. \tag{4.38}$$

The leapfrog discretization almost preserves $\mathcal{H}$ to the order of $\mathcal{O}(\epsilon^2)$, preserves volumes since only shear transformation are involved, and it is time reversible. [32].

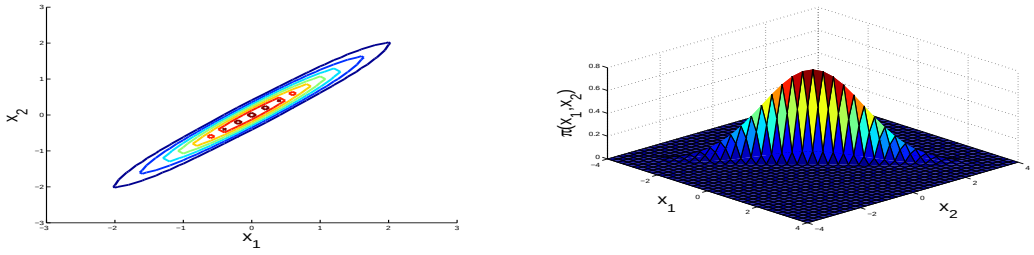


FIGURE 4.1: The contour plot (left), and 3-dimensional plot of the correlated Gaussian distribution defined in (4.39).

Algorithm 4.1: The Hamiltonian Monte Carlo–leapfrog

Initialize q_0 and p_0
Set ϵ
for $i = 1$ *to* $nsamples$ **do**
    draw $p \sim \mathcal{N}(0, I)$
    $(q^{(0)}, p^{(0)}) = (q_{i-1}, p)$
    for $j = 1$ *to* L **do**
        $p^{(j-\frac{1}{2})} = p^{(j-1)} - \frac{\epsilon}{2}\nabla V(q^{(j-1)})$
        $q^{(j)} = q^{(j-1)} + \epsilon p^{(j-\frac{1}{2})}$
        $p^{(j)} = p^{(j-\frac{1}{2})} - \frac{\epsilon}{2}\nabla V(q^{(j)})$
    end
    $(q', p') = (q^{(L)}, p^{(L)})$
    draw $\alpha \sim \mathcal{U}[0, 1]$
    $\delta\mathcal{H} = \mathcal{H}(q', p') - \mathcal{H}(q^{(0)}, p^{(0)})$
    if $\alpha < \min\{1, \exp(-\delta\mathcal{H})\}$ **then**
        $(q_i, p_i) = (q', p')$
    else
        $(q_i, p_i) = (q_{i-1}, p_{i-1})$
    end
end
return $\{q_i, p_i\}_{i=0}^{nsamples}$

Since the parameters q move in the direction of the momentum p during each dynamical proposal, the state of the system tends to move linearly with time. In contrast the system in random walk algorithms, such as the Metropolis– Hasting, usually move distance proportional to $\sqrt{k}$ in k iterations [32].

Figure 4.1 shows the contour plot and the corresponding 3-dimensions plot of the distribution in (4.39). In Figure 4.2 we compare HMC versus Metropolis–Hastings (MH) algorithm to give a realization of the correlated Gaussian distribution in two dimensions (4.39) see Figure 4.1, where the covariance matrix is given in (4.39b). In MH algorithm we use the optimal proposal distribution as discussed in [9] to sample from it a candidate point. The optimal proposal distribution for Gaussian target that we used defined by $\Sigma = \left(\frac{2.4}{\sqrt{D}}\right)^2 \sigma_0 I$, where σ_0 is the standard deviation of the target distribution, D the dimension, and $I \in \mathbb{R}^{D\times D}$ is identity matrix.

$$\pi(x) = \frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2} x^T \Sigma^{-1} x\right), \tag{4.39a}$$

$$\Sigma = \begin{pmatrix} 1 & 0.98 \\ 0.98 & 1 \end{pmatrix}. \tag{4.39b}$$

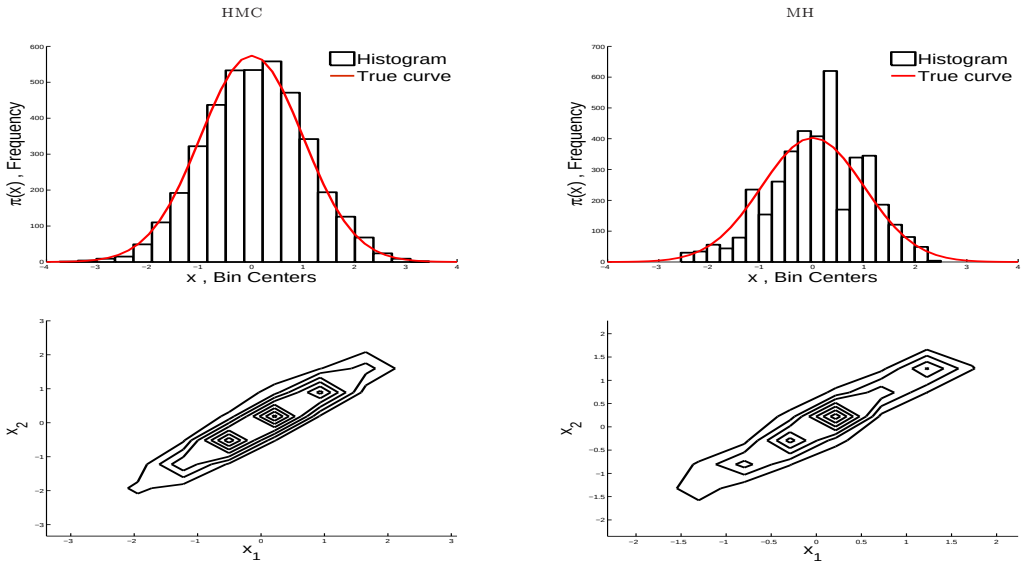


FIGURE 4.2: The marginal distributions and contour plots for the correlated 2–dimensional Gaussian distribution, sampled from both Hamiltonian Monte Carlo (HMC) on the left, and Metropolis–Hastings (MH) on the right, with chain length $n = 4000$ for each. It is clear that HMC captured the target distribution very well, while MH appears trapped, leading to a skewed distribution.

Chapter 5

Problem definition

The thesis investigates solutions to the problems in subsections 5.1.2 and 5.1.4.

5.1 The HMC algorithms in practice

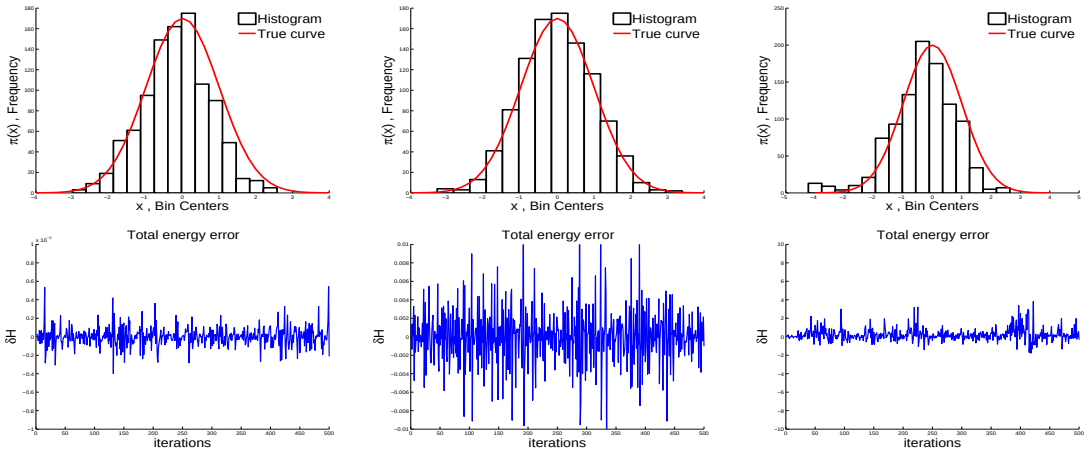
Since the Hamiltonian dynamics is time reversible and preserves the volume and total energy, the movement along the trajectories of constant energy $\mathcal{H}$ will leave the joint distribution $P(q, p)$ invariant [14] if the dynamics is simulated exactly. However, in practice, the dynamic is simulated with finite number of steps using the leapfrog scheme and this leads to errors in the simulation. Thus when using the HMC algorithm in practice, we have to address two main issues to get good performance; controlling the step–size ϵ and choosing the simulation length L.

5.1.1 Step–size effect

Figure 5.1 shows the effect of choosing different step–sizes in the leapfrog scheme for the HMC algorithm for a 2-dimensional Gaussian distribution with $\Sigma = I$. One of the simple ideas to find a good approximation for ϵ is to run the HMC algorithm for different values of ϵ and monitor the acceptance rate for each of them, then choose ϵ with highest acceptance rate. This approach could be impractical and time consuming, especially for higher dimensions, and for chains with long burn–in period.

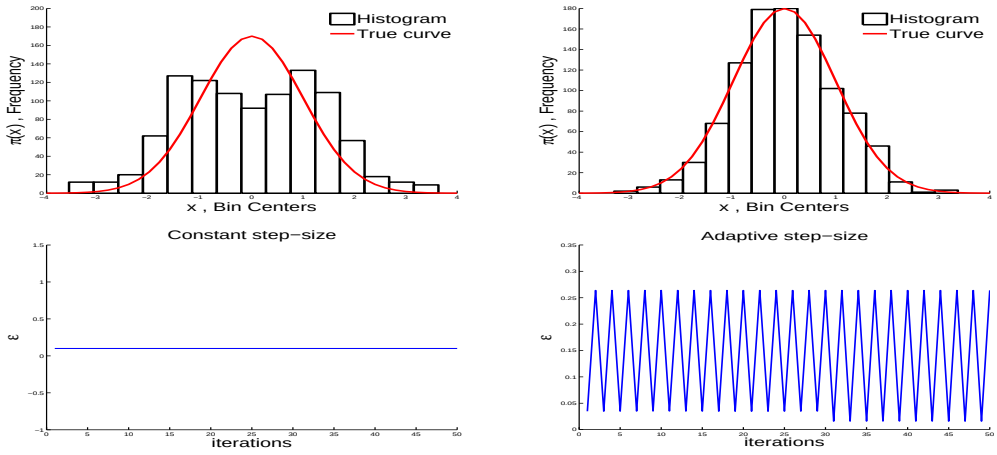
5.1.2 Constant vs. adaptive step–size

While small ϵ implies good exploration of the distribution space, it could be computationally expensive. In general, constant step–size schemes can lead to extra computational costs, especially in the application where the dynamic evolves with different speeds in different regions of the trajectory. Alternatively, one can use adaptive step–size to move with variable ϵ, depending on whether it is in a region of low or high probability. Figure 5.2 shows the effect of using adaptive step–size to reduce both computation cost and simulation errors involved in using constant step–size.



(a) $\epsilon = 0.025,\ L = 20$ (b) $\epsilon = 0.1,\ L = 20$ (c) $\epsilon = 1.9,\ L = 30$

FIGURE 5.1: The figure shows the effect of varying the step–size $\epsilon = 0.025, 0.1$, and 1.9, in the histogram of the chain (top row), and the corresponding change in the total energy (bottom row). Observe the difference in scaling of δH axis.



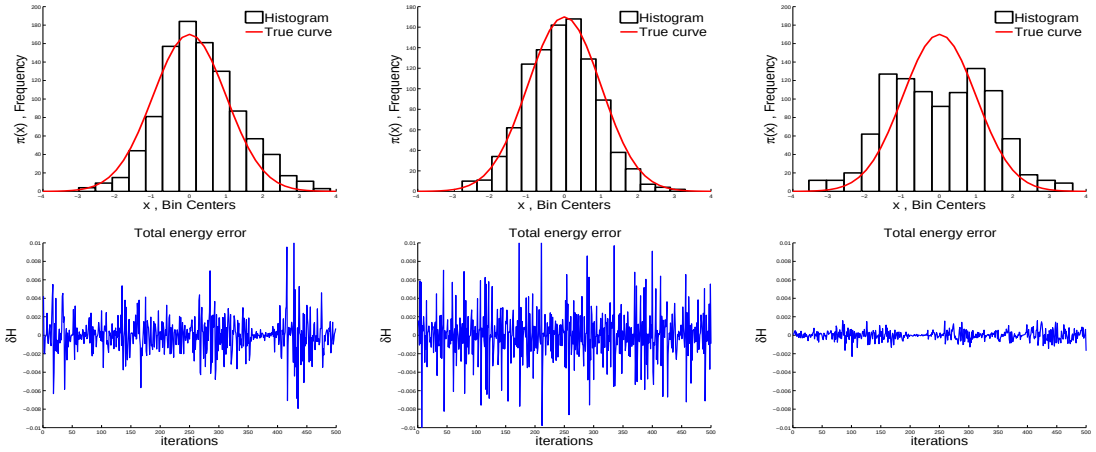
(a) Constant step–size, $\epsilon = 0.1,\ L = 30$ (b) Adaptive step–size, $L = 30$

FIGURE 5.2: The effect of using adaptive step–size instead of constant steps. The target distribution is a 2–dimensional Gaussian, $\Sigma = I$.

5.1.3 Number of simulation steps

Once the good step–size have been established, we need to pick appropriate length L of the simulation. The length of the simulation should be long enough to take the walker far from the starting point. However, this leads to computational costs. In contrast, few simulations step leads correlation between the points. Thus we need a tared off between long and short length of the simulation. This can usually done by some experimentation, e.g. monitoring the autocorrelation function and the efficiency of the chain, for more details about these diagnostics see Chapter 7.

Figure 5.3 presents example results of varying the number of simulation step in the HMC algorithm when sampling from a 2-dimensional Gaussian distribution with $\Sigma = I$. The figure shows that the wrong choice of L, leads to the chain converging to the wrong distribution, see Figure 5.3 (c)

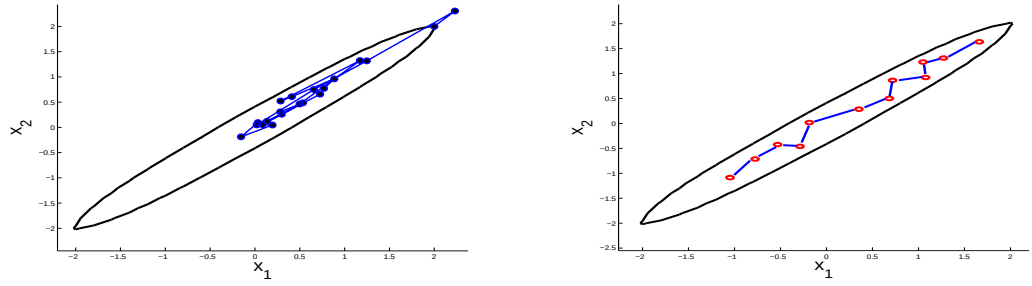


(a) $\epsilon = 0.1,\ L = 5$ (b) $\epsilon = 0.1,\ L = 10$ (c) $\epsilon = 0.1,\ L = 30$

FIGURE 5.3: The figure shows the effect of using different values for the number of leapfrog steps $L = 5, 10$, and 30, in the histogram of the chain (top row), and the corresponding change in the total energy (bottom row). Observe the difference in scaling of δH axis.

5.1.4 The random walk in choosing the momentum

Drawing the momentum variable in the HMC algorithm follows the Gibbs sampling, which is a random walk algorithm. Thus the momentum variable moves back and forth in the distribution space. Hence the algorithm takes a long time to traverse the space defined by the distribution. Figure 5.4 (a) shows a typical behavior of the random walk effect when the momentum variable is chosen by the Gibbs algorithm. Figure 5.4 (b) shows the desirable trajectory of the sample.



(a) HMC trajectory (b) desirable trajectory

FIGURE 5.4: The plot on the left shows the points generated by the HMC algorithm from the 2-dimensional correlated Gaussian distribution. The right plot presents the desirable behavior of the point without random walks.

Chapter 6

Improving the Hamiltonian Monte Carlo algorithm

This chapter introduces the main theories which underpin the thesis. Section (6.1) gives the basic definitions, which are needed to understand subsequent sections. Sections 6.2 through 6.5 discuss strategies for improving HMC algorithm. We introduce the following basic definitions.

6.1 Basic definitions

Definition 6.1. The Order statistics of random variables $X_0, \ldots, X_k$ are the sample values placed in ascending order such as in (6.1). They are denoted by $X^{[0]}, \ldots, X^{[k]}$.

$$X^{[0]} \leq X^{[1]} \leq \cdots \leq X^{[r]} \leq \cdots \leq X^{[k]}. \tag{6.1}$$

Theorem 6.2. *Let $X_1, \ldots, X_n$ be an independent and identically distributed (i.i.d.) random variable from the uniform distribution on $[0, 1]$, and $X^{[0]}, \ldots, X^{[n]}$ be the ordered statistics. Then the following statements are satisfied.*

(a) Let $N(x)$ be the cardinality (the number of element) of the set $\{X_i \mid X_i \leq x; i = 1, \ldots, n\}$, then the cumulative distribution function of $X^{[k]}$ is given by (6.2).

$$F_{[k]}(x) = Pr(N(x) \geq k) = \sum_{r=k}^{n} \binom{n}{r} x^r (1-x)^{n-r}. \tag{6.2}$$

(b) The probability density function $f_{[k]}(x)$ of $X^{[k]}$ is a Beta distribution with parameters k and $n - k + 1$.

$$\begin{aligned} f_{[k]}(x) &= Beta(k, n-k+1), \\ &= \frac{n!}{(k-1)!(n-k)!} x^{k-1}(1-x)^{n-k}. \end{aligned} \tag{6.3}$$

Proof of part (b):
Call the event $\{X_i \leq x\}$ "success", and the event $\{X_i > x\}$ "failure". Hence $N(x)$ count the

number of success, then $N(x)$ is binomially distributed $N(x) \sim \text{binomial}(n, x)$, and success event $\{X_i \leq x\}$ is equivalent to the event $\{N(x) \geq k\}$; that is, at least k of the sample values are less than or equal to x. The equation (6.2), which is then established.

proof of part (b): By differentiating (6.2) with respect to x we have

$$\begin{aligned} f_{[k]}(x) &= \sum_{r=k}^{n} \binom{n}{r} \left[r x^{r-1}(1-x)^{n-r} - x^r (1-x)^{n-r-1} \right], \\ &= \sum_{r=k}^{n} (S_{j-1} - S_j), \end{aligned}$$

where $S_j = \binom{n}{r}(n-r)x^r(1-x)^{n-r-1}$, Since $S_n = 0$ the sum telescopes down to S_{k-1}, which is gives

$$\begin{aligned} f_{[k]}(x) &= \binom{n}{k-1}(n-k+1)x^{k-1}(1-x)^{n-k}, \\ &= \frac{n!}{(k-1)!(n-k)!} x^{k-1}(1-x)^{n-k}. \qquad \square \end{aligned}$$

The above theory and it is prove is taken from [6, 7].

6.2 Over–relaxation methods

Over–relaxation is a method that was proposed to reduce the random walk behavior in the Gibbs sampling technique for a limited type of problem, namely, when all of the conditional distribution $T(p_i \,|\, \{p_j\}_{j \neq i})$ are Gaussian. For example, the distribution $T(p_1, p_2) = \exp(-4p_1^2 p_2^2 - 2p_1^2 - 2p_2^2 - 8)$ is not Gaussian, but all of its conditional densities are Gaussian [26, 34].

Alder's Over–relaxation [1] is applicable when the log probability density is multiquadratic, which is equivalent to when all of the conditional densities are Gaussian. Adler's over–relaxation replaces the Gibbs sampling update of component i by new value p_i', which will depend on it is conditional mean μ_i, and variance σ_i as in (6.4).

$$p_i' = \mu_i + \alpha(p_i - \mu_i) + \sigma_i(1-\alpha^2)^{\frac{1}{2}} u, \tag{6.4}$$

where $u \sim \mathcal{N}(0,1)$, and the parameter $-1 \leq \alpha \leq 1$ controls the degree of over–relaxation.

6.2.1 Ordered over-relaxation

Ordered over–relaxation [34, 36] is a technique to suppress the random walk of the Gibbs sampling for some distributions with strong positive correlations. The normal iteration of Gibbs sampling involves resampling $p_i^{(t+1)}$ from it is conditional distribution $T(p_i \,|\, \{p_j^{(t)}\}_{j \neq i})$. Ordered over–relaxation was introduced as a generalization of the Adler's overrelaxation method, applicable to

any system where Gibbs sampling is applied [34, 36]. A naive implementation of ordered over–relaxation is to pick one value uniformly at random from k samples drawn from the conditional distribution, or by including the current state and choosing a value uniformly at random from $k+1$ values. The ordered overrelaxation transition operator requires that the $k+1$ candidates for p_j be ordered in some way. Numerical sorting is possible when the $k+1$ candidates are real scalars [30]. The candidates are then relabeled according to (6.5).

$$p^{[0]} \leq p^{[1]} \leq \cdots \leq p^{[r]} = p_j \leq \cdots \leq p^{[k]}. \tag{6.5}$$

Based on (6.5), the operator chooses $p_j = p^{[k-i]}$ as the next step in the Markov chain. The rule is deterministic and reversible. Hence it leaves a uniform distribution over the points stationary [30]. Further, it is estimated that for practical purposes, ordered over–relaxation may speed up a simulation by a factor of $\sim 10 - 20\%$ [26].

To show the validity of Ordered over–relaxation method as describe in [34], it is enough to prove that each update for component j satisfies the reversibility condition (details balance, see equations (3.2)). Assume that p_j is replaced by p'_j using the over-relaxation method, i.e. we made a transition from p_j to p'_j. Since we have other $k-1$ values in our consideration along with p'_j, Then the transition probability $P(p_j, p'_j)$ of the Ordered over–relaxation is given in (6.6)

$$P(p_j, p'_j) = k!\, T(p'_j \,|\, \{p_i\}_{i\neq j})\, \mathcal{I}(s = k - r) \prod_{r\neq t\neq s} T(p_j^{[t]} \,|\, \{p_i\}_{i\neq j}), \tag{6.6}$$

where r is order of the old value p_j in (6.5), and s is the order of the chosen value p'_j. The indicator function $\mathcal{I}(s = k-r)$ takes the value 1, or 0 depending in whether the transition occur, or not in the set of $k-1$ value respectively. Now

$$\begin{aligned}
&T(p_j \,|\, \{p_i\}_{i\neq j})\, P(p_j, p'_j),\\
&\quad = T(p_j \,|\, \{p_i\}_{i\neq j})\, .\, k!\, T(p'_j \,|\, \{p_i\}_{i\neq j})\, \mathcal{I}(s = k - r) \prod_{r\neq t\neq s} T(p_j^{[t]} \,|\, \{p_i\}_{i\neq j}),\\
&\quad = T(p'_j \,|\, \{p_i\}_{i\neq j})\, .\, k!\, T(p_j \,|\, \{p_i\}_{i\neq j})\, \mathcal{I}(k - r = s) \prod_{r\neq t\neq s} T(p_j^{[t]} \,|\, \{p_i\}_{i\neq j}),\\
&\quad = T(p'_j \,|\, \{p_i\}_{i\neq j})\, P(p'_j, p_j). \qquad \square
\end{aligned}$$

In practice we can implement ordered over–relaxation in time that is proportional to k using some properties of the uniform distribution [34]. Suppose we want to apply the ordered over–relaxation on the distribution $T(p)$, where $x \in \mathbb{R}^n$.

Let $F(x_i)$ be the cumulative distribution function (cdf) for the conditional distribution $T(p_i \,|\, \{p_j\}_{j\neq i})$, and $F^{-1}(p_i)$ is the inverse of $F(p_i)$. Then applying over–relaxation for p_j is equivalent to:

(1) Transform p_j to u_j using the cdf, $u_j = F(p_j)$.

(2) Perform the ordered over–relaxation on u_j.

(3) Transform back to $p'_j = F^{-1}(u'_j)$.

Algorithm 6.1: Ordered Over-relaxation

```
Compute u = F(p_j);
Generate integer r ~ Binomial(k, u);
if r > k − r then
    Generate v ~ Beta(K − r + 1, 2r − k);
    u' = uv;
else if r < k − r then
    Generate v ~ Beta(r + 1, k − 2r);
    u' = 1 − (1 − u)v;
else if r = k − r then
    u' = u
end
p_j = F^{-1}(u)
```

Step (2) requires generating k additional samples from the uniform distribution. Then using part (a) of Theorem 6.2 we can easily find the number of samples r that are less or equal to u_j in the order statistics (6.7) of the k samples, this number will be binomially distributed with parameters k, and u_j, i.e. $r \sim \text{Binomial}(k, u_j)$.

$$u^{[0]} \leq u^{[1]} \leq \cdots \leq u^{[r]} = u_j \leq \cdots \leq u^{[k]}. \tag{6.7}$$

By assuming that $r > \frac{k}{2}$, Since we will pick $u' = u^{[k-r]}$, which is the $k - r + 1$th statistic of sample of size r from Uniform$[0, u_j]$. According to part (b) of Theorem 6.2 the distribution of $u' \sim \text{Beta}(k - r + 1, 2r - k)$. Algorithm 6.1 is the pseudo-code of ordered overrelation method according to this result.

6.3 Symplectic integrators

Here we focus our discussion about improving the simulation of the Hamiltonian dynamics in Chapter 4, where the simulation involved integration using the leapfrog scheme. The leapfrog scheme belongs to general class of geometric integrators used in the numerical solution of the Hamilton's equations in (4.26) [25].

To introduce the concept of symplecticity, first define $z = (q, p)$, and let the trajectory governed by the Hamiltonian, $\mathcal{H}(q, p)$, be $z(\tau)$. Then the map defined by this trajectory over some time interval τ, $z \longrightarrow Z = L_\tau(z)$, is said to be a canonical or symplectic transformation, if it satisfies

(6.8), [13].

$$M\mathbf{J}M^T = \mathbf{J}, \tag{6.8}$$

where

$$M_{ij} = \frac{\partial Z_i}{\partial z_j} \quad \text{and,}$$
$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}.$$

If S is the time–reversal operator (e.g. for Cartesian coordinates $S(q,p) = (q,-p)$), the dynamic system, L_l, satisfying (6.9) is referred to as being reversible.

$$L_l S L_l = T. \tag{6.9}$$

The properties of symplecticity and reversibility have implications for numerical integrators of the Hamiltonian equations [19, 42]. Specifically, if the map is symplectic, then the numerical integrator should as well be symplectic, while reversibility of the dynamical system constrains the map defined by the numerical integration algorithm to be reversible as well. Further, any Hamiltonian generates a symplectic transformation [25].

The leapfrog in the HMC algorithm works well because in addition to being time–reversible, it has been shown to be the simplest in the class of schemes, which generate a symplectic transformation, [21, 41, 55]. The leapfrog has a one–step error $\sim \mathcal{O}(\epsilon^3)$.

Irrespective of the underlying structure of the dynamics, which an integrating scheme seeks to preserve, accurate and stable fixed stepsize numerical integration schemes often require excessively small timesteps [20]. In many applications where the dynamical system can evolve either rapidly or slowly along different regions of its trajectory, the use of a constant step-size could lead to extra computational costs.

6.4 Strategies for improving the HMC

This thesis investigates two versions of HMC described in Chapter 4. First, by suppressing the random walks behavior introduced in the sampling the momentum variables, i.e. the Gibbs sampling step, using the ordered over–relaxation method. Second, using adaptive step–size in simulating the Hamiltonian dynamics, to allow different step–sizes according to present information, using the Störmer–Verlet discretization scheme.

6.4.1 Restricting random walk in the momentum

The performance of the HMC algorithm can be enhanced by suppressing random walk in the Gibbs sampling. Ordered overrelaxation has been suggested in the literature as a means of suppressing random walk behavior, and which is applicable to any system involving Gibbs sampling. However, has not been applied to the HMC algorithm.

Algorithm 6.2: HMC algorithm with ordered over–relaxation applied to the Gibbs sampling stage to drawn the momentum variables.

Initialize q_0 and p_0
for $i = 1$ *to* $nsamples$ **do**
  Compute $u = \text{cdf}(\text{`Normal`}, p; 0, 1)$
  Generate integer $r \sim \text{Binomial}(k, u)$
  if $r > k - r$ **then**
    Generate $v \sim \text{Beta}(K - r + 1, 2r - k)$
    $u' = uv$
  else if $r < k - r$ **then**
    Generate $v \sim \text{Beta}(r + 1, k - 2r)$
    $u' = 1 - (1 - u)v$
  else if $r = k - r$ **then**
    $u' = u$
  end
  $p = \text{icdf}(\text{`Normal`}, u; 0, 1)$
  $(q^{(0)}, p^{(0)}) = (q_{i-1}, p)$
  for $j = 1$ *to* L **do**
    $p^{(j-\frac{1}{2})} = p^{(j-1)} - \frac{\epsilon}{2}\nabla V(q^{(j-1)})$
    $q^{(j)} = q^{(j-1)} + \epsilon p^{(j-\frac{1}{2})}$
    $p^{(j)} = p^{(j-\frac{1}{2})} - \frac{\epsilon}{2}\nabla V(q^{(j)})$
  end
  $(q', p') = (q^{(l)}, p^{(l)})$
  draw $\alpha \sim \mathcal{U}[0, 1]$
  $\delta\mathcal{H} = \mathcal{H}(q', p') - \mathcal{H}(q^{(0)}, p^{(0)})$
  if $\alpha < \min\{1, \exp(-\delta\mathcal{H})\}$ **then**
    $(q_i, p_i) = (q', p')$
  else
    $(q_i, p_i) = (q_{i-1}, p_{i-1})$
  end
end
return $\{q_i, p_i\}_{i=0}^{nsamples}$

In this thesis we investigate and presents results on the performance of the HMC algorithm, with ordered overrelaxation applied to the Gibbs sampling stage of the algorithm. We refer to this version as the OHMC algorithm. The performance of the OHMC algorithm is evaluated in terms of the acceptance and convergence rates, as well as the degree of correlation of the chains. Algorithm 6.2 is the pseudo code for OHMC algorithm.

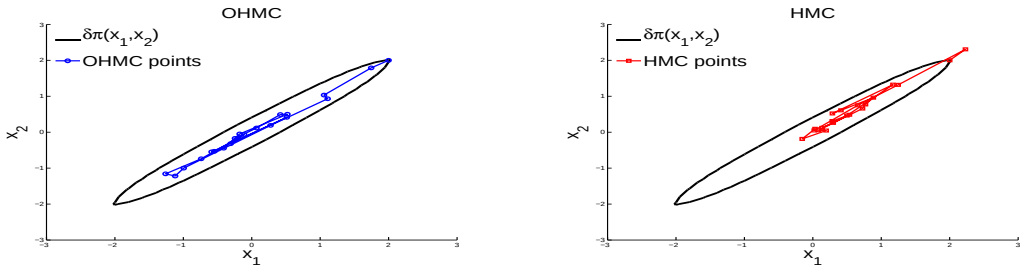


FIGURE 6.1: The contour plot (left), and 3-dimensional plot of the correlated Gaussian distribution defined in (4.39).

Figure 6.1 shows the OHMC algorithm applied to a highly correlated bivariate Gaussian (see the example in (4.39)). Some exact samples from the distribution are shown in blue. On the other hand, the Metropolis–Hastings algorithm for comparison is shown in red proceeds by slow diffusion. OHMC is able to make persistent progress along the distribution. In this example we set the over–relaxation parameter $k = 10$.

6.4.2 Adaptive step–size

The possible construction of variable stepsize, time–reversible integration schemes was first demonstrated in [21, 45]. Any integration scheme, which is symplectic for constant time-step cannot necessarily retain its symplectic characteristic with variable time-stepping, see [5, 12].

Further, rather than solve the original Hamiltonian problem, $\mathcal{H}(q,p)$, in adaptive time stepping, the integrators (roughly speaking) attempt to solve a perturbed time-dependent Hamiltonian, $\mathcal{H}(q,p) + \delta\tilde{\mathcal{H}}(q,p,\tau)$ [4]. Unless the time step is properly changed, secular terms are introduced. Consequently, $\delta\tilde{\mathcal{H}}$ grows with τ and the error in energy and positions grows similarly to standard non symplectic integrators [4].

An explicit stepsize scheme for the Störmer–Verlet scheme has been reported in [20], and improved by [19]. Lately, other schemes, e.g., [17], have been reported in the literature. Here we implements the scheme reported in [19].

In [20], the authors introduced a variable modification of the Störmer–Verlet scheme by introducing a fictive variable ρ, which was related to a scaling function U. The resulting variable stepsize Störmer–Verlet scheme is explicit if U depends only on q and semi-explicit if U depends on p. The approach adopted in [19] generalizes and simplifies the scheme in [20].

Equation (6.10)–(6.14) summarizes the explicit variable step–size and time reversible Störmer–Verlet integration scheme, see [20] for details.

$$q_{n+\frac{1}{2}} = q_n + \frac{\epsilon}{2\rho_n} p_{n+\frac{1}{2}}, \tag{6.10}$$

$$p_{n+\frac{1}{2}} = p_n - \frac{\epsilon}{2\rho_n} \nabla E(p_n), \tag{6.11}$$

$$\rho_{n+1} + \rho_n = 2U(q_{n+\frac{1}{2}}, p_{n+\frac{1}{2}}), \tag{6.12}$$

$$p_{n+1} = p_{n+\frac{1}{2}} - \frac{\epsilon}{2\rho_{n+1}} \nabla E(q_{n+1}), \tag{6.13}$$

$$q_{n+1} = q_{n+\frac{1}{2}} + \frac{\epsilon}{2\rho_{n+1}} p_{n+\frac{1}{2}}. \tag{6.14}$$

The scheme is fully explicit, symmetric and time–reversible if $U(q,p) = U(q,-p)$. Algorithm 6.3 is a pseudo code for the implementation of applying the Störmer–Verlet discretization to HMC algorithm.

Algorithm 6.3: HMC algorithm with Störmer–Verlet discretization used to simulate the Hamiltonian dynamics

Initialize q_0 and p_0
Set ϵ
for $i = 1$ *to* $nsamples$ **do**
    draw $p \sim \mathcal{N}(0, I)$
    $(q^{(0)}, p^{(0)}) = (q_{i-1}, p)$
    Set ρ_0
    for $j = 1$ *to* L **do**
        $p^{(j-\frac{1}{2})} = p^{(j-1)} - \frac{\epsilon}{2\rho_{j-1}} \nabla V(q^{(j-1)})$
        $q^{(j-\frac{1}{2})} = q^{(j-1)} + \frac{\epsilon}{2\rho_{j-1}} p^{(j-\frac{1}{2})}$
        $\rho_j = -\rho_{j-1} + 2G(q^{(j-\frac{1}{2})}, q^{(j-\frac{1}{2})})$
        $q^{(j)} = q^{(j-\frac{1}{2})} + \frac{\epsilon}{2\rho_j} p^{(j-\frac{1}{2})}$
        $p^{(j)} = p^{(j-\frac{1}{2})} - \frac{\epsilon}{2\rho_j} \nabla V(q^{(j)})$
    end
    $(q', p') = (q^{(l)}, p^{(l)})$
    draw $\alpha \sim \mathcal{U}[0,1]$
    $\delta\mathcal{H} = \mathcal{H}(q', p') - \mathcal{H}(q^{(0)}, p^{(0)})$
    if $\alpha < \min\{1, \exp(-\delta\mathcal{H})\}$ **then**
        $(q_i, p_i) = (q', p')$
    else
        $(q_i, p_i) = (q_{i-1}, p_{i-1})$
    end
end
return $\{q_i, p_i\}_{i=0}^{nsamples}$

An initial value for the fictive variable ρ, and the scaling function, U must be determined in order to implement the scheme. The paper suggests $\rho_0 = U(q_0, p_0)$. However, a modified initialization of ρ_0 is suggested as an inappropriate choice could introduce undesirable oscillations in the numerically computed values of ρ_n. The original time variable τ_n is recover from the update (6.15).

$$\tau_{n+1} = \tau_n + \frac{\epsilon}{2(\rho_n + \rho_{n+1})}. \tag{6.15}$$

Choices of this function $U(q, p)$ can be found in [19, 20]. Here we adopt the definition of U in (6.16).

$$U(q, p) = \sqrt{E'(q)[E'(q)]^T + p^T E''(q) E''(q) p}, \tag{6.16}$$

$$E'(q) = \nabla E(q)^T, \tag{6.17}$$

$$E''(q) = \nabla E'(q)^T. \tag{6.18}$$

The theoretical acceptance rate for HMC algorithm is approximated by $\text{erfc}(\frac{1}{2}\sqrt{< \delta\mathcal{H} >})$ [22], where $\delta\mathcal{H}$ is vector containing the error of total energy in each step in the simulation, and erfc(.) is the error function, see the right plot of Figure 6.2. The left plot of Figure 6.2 shows the simulation error of the adaptive Störmer–Verlet method and the leapfrog scheme.

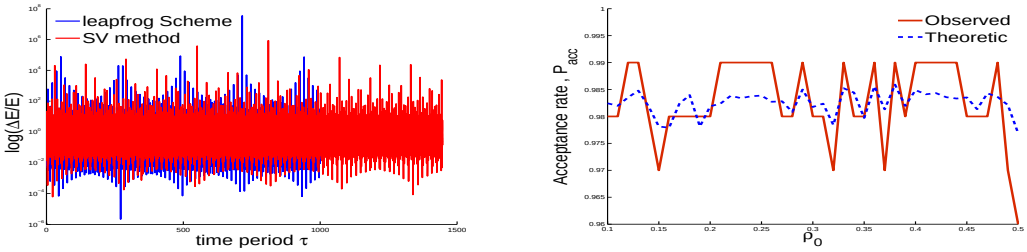


FIGURE 6.2: The error introduced in the Hamiltonian $\mathcal{H}$ for both Leapfrog and Störmer–Verlet (SV) scheme (left plot). The plot on the right shows the comparison of the theoretical and observed acceptance rate using different starting value for the fictive variable ρ_0. Here $P_{acc} = \text{erfc}(\frac{1}{2}\sqrt{< \delta\mathcal{H} >})$.

6.5 Combining ordered Over-relaxation and Störmer–Verlet

The third idea is combine the ordered over–relaxation in Gibbs sampling step and simulate the dynamics using the adaptive Störmer–Verlet discretization. Algorithm 6.4 is a pseudo code for the implementation of applying the Störmer–Verlet discretization and ordered over–relaxation to HMC algorithm.

Algorithm 6.4: HMC algorithm using ordered over–relaxation to pick the momentum variables, and Störmer–Verlet scheme to simulate the Hamiltonian dynamics.

Initialize q_0 and p_0
Set ϵ
for $i = 1$ *to* $nsamples$ **do**
    Compute $u = \text{cdf}(\text{`Normal'}, p; 0, 1)$
    Generate integer $r \sim \text{Binomial}(k, u)$
    if $r > k - r$ **then**
        Generate $v \sim \text{Beta}(K - r + 1, 2r - k)$
        $u' = uv$
    else if $r < k - r$ **then**
        Generate $v \sim \text{Beta}(r + 1, k - 2r)$
        $u' = 1 - (1 - u)v$
    else if $r = k - r$ **then**
        $u' = u$
    end
    $p = \text{icdf}(\text{`Normal'}, u; 0, 1)$
    $(q^{(0)}, p^{(0)}) = (q_{i-1}, p)$
    Set ρ_0
    for $j = 1$ *to* L **do**
        $p^{(j-\frac{1}{2})} = p^{(j-1)} - \frac{\epsilon}{2\rho_{j-1}} \nabla V(q^{(j-1)})$
        $q^{(j-\frac{1}{2})} = q^{(j-1)} + \frac{\epsilon}{2\rho_{j-1}} p^{(j-\frac{1}{2})}$
        $\rho_j = -\rho_{j-1} + 2G(q^{(j-\frac{1}{2})}, q^{(j-\frac{1}{2})})$
        $q^{(j)} = q^{(j-\frac{1}{2})} + \frac{\epsilon}{2\rho_j} p^{(j-\frac{1}{2})}$
        $p^{(j)} = p^{(j-\frac{1}{2})} - \frac{\epsilon}{2\rho_j} \nabla V(q^{(j)})$
    end
    $(q', p') = (q^{(l)}, p^{(l)})$
    draw $\alpha \sim \mathcal{U}[0, 1]$
    $\delta\mathcal{H} = \mathcal{H}(q', p') - \mathcal{H}(q^{(0)}, p^{(0)})$
    if $\alpha < \min\{1, \exp(-\delta\mathcal{H})\}$ **then**
        $(q_i, p_i) = (q', p')$
    else
        $(q_i, p_i) = (q_{i-1}, p_{i-1})$
    end
end
return $\{q_i, p_i\}_{i=0}^{nsamples}$

Chapter 7

Criteria for evaluating the improved HMC algorithm

If the distribution of the points generated by any MCMC algorithm is asymptotically convergent to the exact target distribution $\pi(x)$, then we can estimate the statistical properties such as mean, median, and quantiles to our distribution [10, 39]. The estimation improves as the number of samples approaches infinity. However, this not reasonable in practice. Instead, we generate a finite chain to give estimation of the statistical properties for the target distribution. The error arises from the truncation to finite chain both because of the shot–noise and correlation between successive elements of the chain. We can say that the finite chain is converged, if the statistical properties, sufficiently reflect those of the target distribution $\pi(x)$ with sufficient accuracy [9]. To achieve convergence of such chains, the following two requirements must be satisfied.

(1) The chain should fully traverse the region of high probability such that the correlation between successive elements does not bias the inferred distribution for $\pi(x)$.

(2) The estimation of statistical properties of the target distribution should be well defined with sufficient accuracy.

To achieve the second requirement of convergence, a level of accuracy for a given statistics must be specified. A common diagnostic criterion is the variation of the sample mean obtained form a finite chain.

There are many diagnostics tests to evaluate the convergence of MCMC chain in the literature. In this thesis we concentrate on a few of them namely the power spectral criteria, i.e. using the power spectral density to estimate the efficiency, convergence ratio, the number of independent samples, and the degree of the correlation in the chain.

7.1 Degree of the correlation criteria

Definition 7.1. The autocorrelation function $\rho(\tau)$ of a given chain $\{x_i\}_{i=1}^n$ describes the correlation between the successive elements x_i, and x_{i+l} of the chain at different lags l. The

autocorrelation function at lag l is defined in (7.1).

$$\begin{aligned}\rho(l) &= \frac{Cov(x_i, x_{i+l})}{Var(x_i)}, \\ &= \frac{\sum_{i=1}^{n-l}(x_i - \bar{x})(x_{i+l} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}.\end{aligned} \tag{7.1}$$

The ideal chain with no correlation between successive elements in the chain has autocorrelation function starting from unity and decays quickly to tauch the zero–line, see the left plot of Figure 7.1.

Definition 7.2. Integrated autocorrelation time τ_{int}, for chain $\{x_i\}_{i=1}^{\infty}$ that have autocorrelation function $\rho(\tau)$ is given in (7.2).

$$\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t). \tag{7.2}$$

The effective number of samples in a chain of length N can be approximated by $N/(2\tau_{int})$. The ideal chain, i.e. the chain with $\rho(\tau) = 0$ for all $\tau \neq 0$, and $\rho(0) = 1$ (uncorrelated case) has an integrated autocorrelation time $2\tau_{int} = 1$.

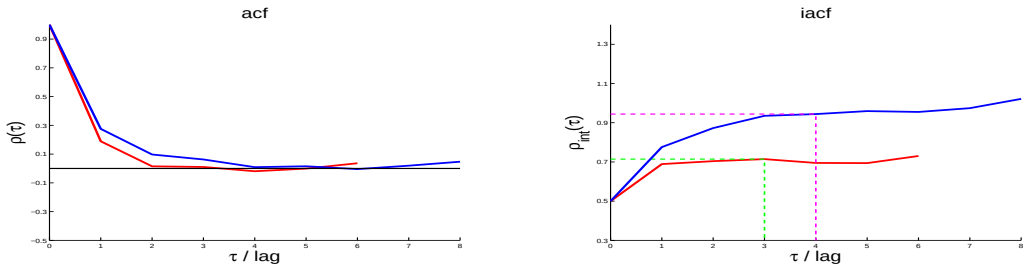


FIGURE 7.1: The figure shows how to compare two given chains. We are interested in the chain with less correlation between the elements (in red), with smallest τ_{int}.

Since the chain produced by any MCMC algorithm is finite, our goal is find a good estimator of the integrated autocorrelation time. Numerical estimation of the integrated autocorrelation time has many difficulties [44, 54]. For example, the autocorrelation become noisy as the t in equation (7.2) grows larger. Hence the sum in (7.2) diverges.

In this thesis, to estimate the integrated autocorrelation time τ_{int} and the autocorrelation function for a given chain, we will follow the methods described by [54] and use the MATLAB program associated with this paper. A typical output from the program with some adaptation is shown in Figure 7.1, where the τ_{int} corresponds to the value when curves values become more flat and stable.

7.1.1 Spectral analysis criteria

Let $\{x_n\}_{n=1}^{\pi}$ be the chain generated by MCMC algorithm, and $\tilde{X}(\kappa)$ denote the discrete Fourier transform this chain given in (7.3), and (7.4) which is the chain itself.

$$\tilde{X}(\kappa) = \sum_{n=-\infty}^{\infty} x_n e^{-in\kappa}, \tag{7.3}$$

$$x_n = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \tilde{X}(\kappa) e^{in\kappa} d\kappa. \tag{7.4}$$

Since the $\{x_n\}_{n=1}^{\infty} \subset \mathbb{R}$ It is easily to show that $\tilde{X}(\kappa) = \tilde{X}^*(-\kappa)$, where $\tilde{X}^*(\kappa)$ denotes the complex conjugate of $\tilde{X}(\kappa)$.

Definition 7.3. The non–normalized power spectral density $h(\kappa)$ of chain $\{x_n\}_{n=1}^{\infty}$ is defined in (7.5). Where $\tilde{X}_T(\kappa)$ is the discrete Fourier tansform of the chain in the period $[-T, T]$.

$$h(\kappa) = \lim_{T\to\infty} \frac{1}{2T} < |\tilde{X}_T(\kappa)|^2 >, \tag{7.5}$$

where the operator $< . >$ means the expected value. Assume that our chain $\{x_n\}_{n=1}^{\pi}$ can be represented by a continuous function $x(t)$, where $x(t_n) = x_n$, for all n. Then we can have the following definitions.

The autocovariance function $R(\tau)$ and the autocorrelation function $\rho(\tau)$ of $x(t)$ are given in (7.6) and (7.7), respectively. Note that in the definition of autocorrelation function in (7.1) we can generalize the formula in (7.1) to cover the one in (7.7), since any integral can be written as a sum.

$$\begin{aligned} R(\tau) &=< x(t)x(t-\tau) >, \\ &= \int_{-\pi}^{\pi} x(u)x(u-\tau)du, \end{aligned} \tag{7.6}$$

$$\begin{aligned} \rho(\tau) &= \frac{< x(t)x(t-\tau) >}{\sigma^2}, \\ &= \frac{1}{\sigma^2} \int_{-\pi}^{\pi} x(u)x(u-\tau)du, \end{aligned} \tag{7.7}$$

where σ^2 is variance of the chain, and hence the variance of $x(t)$, as we assumed. The normalized power spectral density $P(\kappa)$ [38] given by (7.8), will be used to infer information about the convergence of the chain.

$$P(\kappa) = \frac{h(\kappa)}{\sigma^2}. \tag{7.8}$$

Theorem 7.4. *The Wiener–Khinchine Theorem*
Let $\{x_n\}_{n=1}^{\infty}$ be a zero–mean MCMC chain with power spectral density $P(\kappa)$, and autocorrelation

function $\rho(\tau)$. Then $P(\kappa)$ is Fourier transform of $C(\tau)$, i.e.

$$\rho(\tau) = \frac{1}{2\pi}\int_{-\pi}^{\pi} P(u)\cos(\tau u)du. \tag{7.9}$$

The proof can be found for example in [38].

Now let the average of the independent sample of the chain be $\sigma_{\bar{x}}^2$, which gives the variance of the sample mean. A useful diagnostic is variation of the sample mean of finite chain length [9, 14]. This diagnostic is called the convergence ratio r, defined in (7.10).

$$r = \frac{\sigma_{\bar{x}}^2}{\sigma^2}. \tag{7.10}$$

To estimate the convergence ratio, we first consider a finite chain with mean zero, and of length N, i.e. $\{x\}_{n=1}^{N}$, then the sample mean of chain is give by

$$\bar{x}_N = \frac{1}{N}\sum_{n=1}^{N} x_n. \tag{7.11}$$

The variance of $\bar{x}_N$ can be expressed as

$$\begin{aligned} <\bar{x}_N^2> &= \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N} <x_n x_m>, \\ &= \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\frac{1}{2\pi}\int_{-\pi}^{\pi} P(k)\cos[k(m-n)]dk, \\ &= \frac{1}{N}\int_{-\pi}^{\pi}\frac{1}{2\pi N}P(k)\sum_{n=1}^{N}\sum_{m=1}^{N}\cos[k(m-n)]dk. \end{aligned} \tag{7.12}$$

The summation term in (7.12) can be written as

$$\begin{aligned} Re\left(\sum_{n=1}^{N}\sum_{m=1}^{N} e^{k(m-n)i}\right) &= Re\left(\sum_{n=1}^{N}\left(e^{-ki}\right)^n\sum_{m=1}^{N}\left(e^{-ki}\right)^m\right), \\ &= Re\left(\left[\frac{1-e^{-kNi}}{1-e^{-ki}}\right]\cdot\left[\frac{e^{kNi}-1}{e^{ki}-1}\right]\right), \\ &= Re\left(\left[\frac{1-e^{kNi}}{1-e^{ki}}\right]^2\cdot\left[\frac{e^{-kNi}}{e^{-ki}}\right]\right), \\ &= Re\left(\frac{e^{kNi}+e^{-kNi}-2}{e^{ki}+e^{-ki}-2}\right) = \frac{\sin^2[Nk/2]}{\sin^2[k/2]}. \end{aligned}$$

Then using this result in (7.12), we have

$$<\bar{x}_N^2> = \frac{1}{N}\int_{-\pi}^{\pi}\frac{1}{2\pi N}\frac{\sin^2[Nk/2]}{\sin^2[k/2]}P(k)dk. \tag{7.13}$$

Since (7.13) is a weighted average of the power spectral density, it becomes more concentrated around $k = 0$ as chain length become larger. Moreover, using the results in (7.15) and (7.14),

$$\lim_{N\to\infty} \frac{1}{2\pi N} \frac{\sin^2[Nt/2]}{\sin^2[t/2]} = \delta(t), \tag{7.14}$$

$$\int_{-\pi}^{\pi} \frac{1}{2\pi N} \frac{\sin^2[Nk/2]}{\sin^2[k/2]} dk = 1. \tag{7.15}$$

Then the estimation of sample mean of a long chain is given by

$$\sigma_{\bar{x}}^2 = < \bar{x}_N^2 > \approx \frac{1}{N} P(\kappa = 0). \tag{7.16}$$

Through out this thesis the power spectral density function $P(\kappa)$ is approximated in a discrete point using the discrete Fourier transform, computed by the fast Fourier transform (FFT) algorithm. Then power spectral density function for a chain of length N can be approximated by (7.17) using (7.3)

$$\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|. \tag{7.17}$$

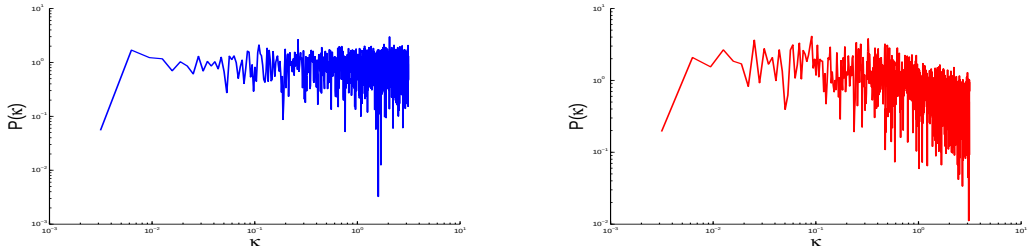


FIGURE 7.2: The discrete power spectral density of uncorrelated (white noise) chain for the Gaussian distribution with mean zero and unit variance (left figure). The discrete power spectral density function of an MCMC chain ($N = 1000$) for 2D Gaussian centered at the origin and with identity covariance (right figure).

The ideal chain produced from the target distribution has a white noise spectral, i.e. there are no correlation between the successive points, then the power spectral density is flat and $P(\kappa) = \sigma$ for all κ, where σ is the standard deviation of the target distribution. the power spectral density function for any MCMC chain follows the template in (7.18) [24, 43].

$$P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}, \tag{7.18}$$

where P_0 is the value of the white noise spectral density function of the chain, κ^* indicates where the white noise spectrum turnover at κ^*, to a different power law behavior, characterize by the parameter α, see Figure 7.2.

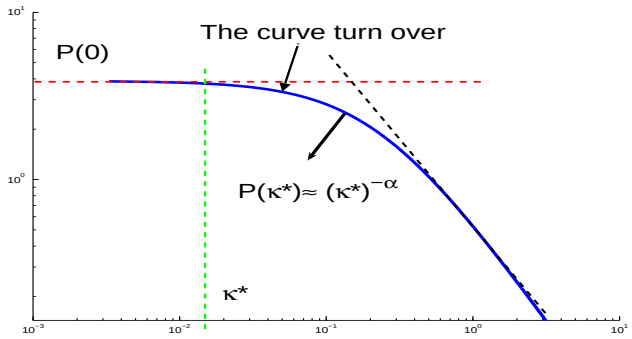


FIGURE 7.3: The power spectral density template for the MCMC chains, the curve is approximately power of two ($\alpha \approx 2$). The figure shows where the curve turnover is, before correlated sample are drawn.

Definition 7.5. Efficiency E of any MCMC chain is ratio of the number of independent points in MCMC chain to the number of MCMC iteration required to reach the same variance.

$$E = \lim_{N\to\infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)},$$

where σ_0^2, and $\sigma_{\bar{x}}^2$ are the variance of the target distribution and the chain sample mean, respectively. σ_0^2/N can be consider as the sample variance of best chain of length $N.E^{-1}$. form equation (7.16) we will get,

$$E = \frac{\sigma_0^2}{P_0} \qquad \text{where} \qquad P_0 = P(0).$$

For a chain of length N, the discrete power spectral density function of the template in equation (7.18) become $\tilde{P}_j$, where $j = 0, 1, 2, \ldots, N/2 - 1$. To fit this template we use the optimization of least square technique supported in MATLAB software, such as the function `lsqcurvefit`, in the range of the Fourier mode $1 \le j \le j_{max}$, for spectrum have a knee at $j^* = \kappa^*(N/2\pi)$, then an appropriate choice of limiting the point in the least square is $j_{max} \approx 10j^*$. In practice, it is good to use a predictor for the template parameters by fitting first an appropriate point-length, and then use this predictor parameters as a starting point.

Let $\sigma_{\bar{x}}^2(N)$ be the variance of sample mean, define by averaging over independent realization of a finite chain of length N. Then the variance of sample mean can be measured by the ratio, $r = \sigma_{\bar{x}}^2/\sigma_0^2$, where σ_0^2 denotes the variance of the target distribution, this ratio is called the convergence ratio. We require r to be below some tolerance value, for example 0.01.

7.1.2 Testing convergence of a chain

Once we know the template parameters P_0, α, and κ^* then we are ready to perform the convergence test, to insure that the MCMC chain achieves enough level of convergence. We check the following two requirements:

(1) the smallest value of κ, must be in the white noise regime $P(\kappa) \approx \kappa^0$, where $j^* = \kappa^*(N/2\pi) > 20$. This insures that the correlated points are not biased and indicates that the chain has explored the regions of high probability.

(2) the estimation of the convergence ration $r = \sigma^2_{\bar{x}}/\sigma^2_0 \approx P_0/N$ for chain with unit variance. To obtain a good accuracy we should generate a chain with, $r < 0.01$.

When the above requirements are met, then we are confident that the chain has converged and the chain can be stopped. We would run the chain longer to get better samples in order to reduce the shot–noise from the histogram, and find good approximation of the statistical properties of our interest. This is particularly relevant for lower dimensions where the chain can converge after relatively few steps. In practice, we find for high dimensional chain ($D \geq 8$), that there are enough samples by the time the test is passed [9]. To avoid saving more values in the chain we sometime save only the mth points in the original chain.

Chapter 8

Simulation and results

This chapter presents numerical experiment on the performance of the Hamiltonian Monte Carlo algorithm with, ordered over–relaxation (OHMC), and adaptive step–size using Störmer–Verlet scheme (SVHMC). The analysis compares these versions to the classical Hamiltonian Monte Carlo algorithm. The assessment is based on using the Gaussian distribution targets with uncorrelated variates in different dimensions. The performance of these versions are judged by the diagnostic tests discussed in Chapter (7).

8.1 Gaussian target and parameter estimation

In many applications, the problem of estimating the model parameter for given data can often be formulated as a least square problem, non–linear system of equation (linear system as especial case), or Bayesian inference problem. Assuming that $G : \mathbb{R}^l \to \mathbb{R}^n$ is a given mathematical model, where the vector $m \in \mathbb{R}^l$ is the model in question. Moreover, suppose that our data is given in the vector $d \in \mathbb{R}^n$. Then the relationship between these quantities is given in (8.1).

$$G(m) = d. \tag{8.1}$$

For simplicity consider the linear least square approach for the problem in (8.1), which can be written in form

$$\arg\min_{m \in \mathbb{R}^l} \|Gm - d\|^2. \tag{8.2}$$

Let the measurement error in particular observation d_i be denoted ϵ_i by

$$\epsilon_i = g_i^T m - d_i, \quad i = 1, 2, \ldots, n,$$

where $g_i \in \mathbb{R}^l$ is ith row in the matrix G, i.e. $G = [g_1 \quad g_2 \ldots g_n]^T$. It is reasonable to assume that the ϵ_i's are independently identically distributed (i.i.d.) with variance σ^2, then

$$Pr(\epsilon_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{\epsilon_i^2}{\sigma^2}\right), \quad i = 1, 2, \ldots, n. \tag{8.3}$$

The probability of the observation data d to match the actual model parameter m is given in (8.4), which is called the likelihood for a particular data m.

$$\begin{aligned} p(d|m) &= \prod_{i=1}^{n} Pr(\epsilon_i) = \prod_{i=1}^{l} Pr(\epsilon_i), \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{l}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(g_i^T m - d_i)^2\right). \end{aligned} \tag{8.4}$$

The distribution in (8.4) is proportional to Gaussian distribution with mean $\mu = m$, and covariance $\Sigma = \sigma^2(G^TG)^{-1}$. To show this we consider the sum

$$\begin{aligned} -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(g_i^T m - d_i)^2 &= -\frac{1}{2\sigma^2}(Gm-d)^T G^T G(Gm-d), \\ &= -\frac{1}{2\sigma^2}(d^T d - 2m^T G^T d + m^T G^T G m). \end{aligned} \tag{8.5}$$

If G^TG is symmetric and positive semidefinite, then the inverse of it exist and have the same properties. By defining the random vector $x \in \mathbb{R}^l$ by $x = (G^TG)^{-1}G^Td$, then we can rewrite (8.5) as

$$\begin{aligned} -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(g_i^T m - d_i)^2 &= -\frac{1}{2}\left[x^T\left(\sigma^{-2}G^TG\right)x - 2m^T\left(\sigma^{-2}G^TG\right)x + m^T\left(\sigma^{-2}G^TG\right)m\right], \\ &= -\frac{1}{2}(x-m)^T\left(\sigma^{-2}G^TG\right)(x-m), \\ &= -\frac{1}{2}(x-m)^T\Sigma^{-1}(x-m). \end{aligned} \tag{8.6}$$

For a suitable constant A the distribution in (8.4) can be written as

$$p(d|m) = A\exp\left(-\frac{1}{2}(x-m)^T\Sigma^{-1}(x-m)\right), \tag{8.7a}$$

$$x = (G^TG)^{-1}G^Td, \tag{8.7b}$$

$$\Sigma = \sigma^2(G^TG)^{-1}. \tag{8.7c}$$

Then transforming the data d, by the transformation $x = (G^TG)^{-1}G^Td$, we get realizations from the Gaussian distribution. There is thus a link between the problem in (8.7) and the least square problem in (8.2). By simulating the distribution in (8.7) we can estimate the stochastic distribution of model m in question.

8.2 Distributions for the numerical experiments

Equation (8.3) presents the univariate case where $A = \frac{1}{\sqrt{2\pi\sigma^2}}$, $\Sigma = (1/\sigma)I$, and $\epsilon = x - m$. For multivariates case (8.3) translate into (8.8) where $A = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}$.

$$\pi(x) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-m)^T\Sigma^{-1}(x-m)\right), \quad x \in \mathbb{R}^D, \tag{8.8}$$

$$\Sigma = I. \tag{8.9}$$

We have investigate several dimensions, in this thesis we present the results of $D = 64$, and 128.

8.3 Results from numerical experiments

The section is devoted to compare the performance of OHMC and SVHMC algorithms to the classical HMC algorithm. Then combine the OHMC and SVHMC algorithm in one algorithm which we call OSVHMC algorithm and compare it is performance to SVHMC algorithm.

The performance of these versions are evaluated in terms of the spectral analysis convergent test, as well as the degree of correlation of the chains, which are discussed in Chapter (6). In the simulation for each algorithm we will produce realizations from 64 and 128 dimensions (parameters) uncorrelated Gaussian distribution (8.8) with mean zero and unit covariance. Since we have more than one chain, one for each parameter, and all of the chains are identical, we will analyze the first chain (the chain for m_1) to get information about the diagnostics criterion. The true value for the parameters we want to estimate is $m_i = 0$ for all i.

8.3.1 Comparing OHMC and HMC algorithms

To see the effect of ordered over–relaxation, we run the over–relaxed version (OHMC) and the classical HMC algorithm with the same values for leapfrog step–size $\epsilon = 0.1$ and $L = 10$ leapfrog steps, and compare the results using the criteria in Chapter 6.

The power spectral analysis and the degree of the correlation are shown in Figures 8.1 and 8.2 for the chains generated by the two algorithms. The OHMC spends longer time on the white-noise regime than HMC, i.e. OHMC chain has more independent realizations from the target than HMC. This is given by calculating the length of the flat part of the power spectral density curve before the curve turns over to a different power region, which measured by the parameter κ^* (see equation (7.18)). These values can be read from Table 8.1 and Figure 8.1.

The numerical values for diagnostic tests are shown in Table 8.1. The parameter $P(0)$ is used to calculate the efficiency E and convergence ratio r of SVHMC and HMC algorithm. The ideal chain has an efficiency of approximately 1. As the table shows, OHMC has better efficiency than

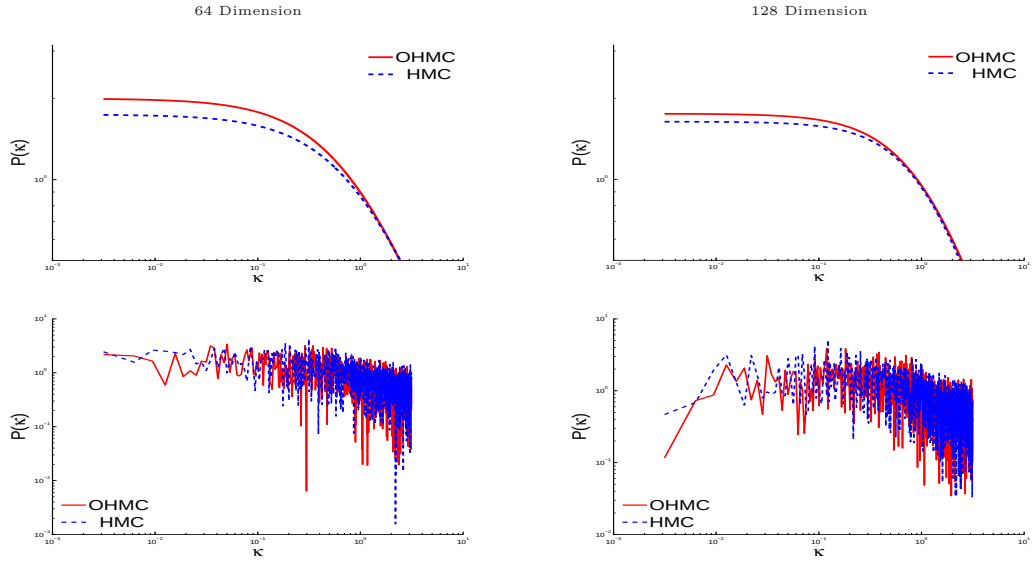


FIGURE 8.1: The discrete power spectral density (psd) and fitted template for the chains generated by the OHMC and HMC algorithm in 64-dimensions (the left column), and 128-dimensions (the right column).

TABLE 8.1: Summary of the power spectral and the degree of correlation convergence tests for OHMC and HMC algorithms in 64 and 128–dimensions.

	64 Dimension		128 Dimension	
	OHMC	HMC	OHMC	HMC
Acceptance Rate	0.993000	0.985000	0.9855000	0.975500
$P(0)$	3.106938	3.560673	2.9206225	3.125949
κ^*	0.985683	0.812680	1.2495117	1.145286
CPU time/Sec	561.224400	557.381400	1128.3454000	1117.776300
Efficiency E	0.321860	0.280845	0.3423927	0.319902
Convergence ratio r	0.001553	0.001780	0.0014603	0.001562
iacf τ_{int}	1.629816	1.845748	1.5333956	1.804368
Error of τ_{int}	0.176595	0.212930	0.1675069	0.208802
Error of estimation μ_1	0.038702	0.043258	0.0393236	0.043891
Estimation of m_1	0.048826	−0.054868	0.0026030	0.010457

HMC. All algorithms reached convergence based on $r < 0.01$ criterion as used in [9]. Also the CPU times for the algorithms are almost identical. However, the OHMC needs a few additional function evaluations in the ordered over–relaxation stage, see Algorithm 6.2.

In term of the efficiency, OHMC chain is about 1.2 times shorter than the HMC chain, i.e. $\frac{E_{OHMC}}{E_{HMC}} \approx 1.2$.. Note also that the we have better convergence of OHMC in estimating the true value for the parameter m in both dimensions.

These feature are seen much better in the autocorrelation function (acf) and the integrated autocorrelation function (iacf), which are shown in Figure 8.2. The bottom left row shows that the autocorrelation of OHMC dies off much faster than HMC.

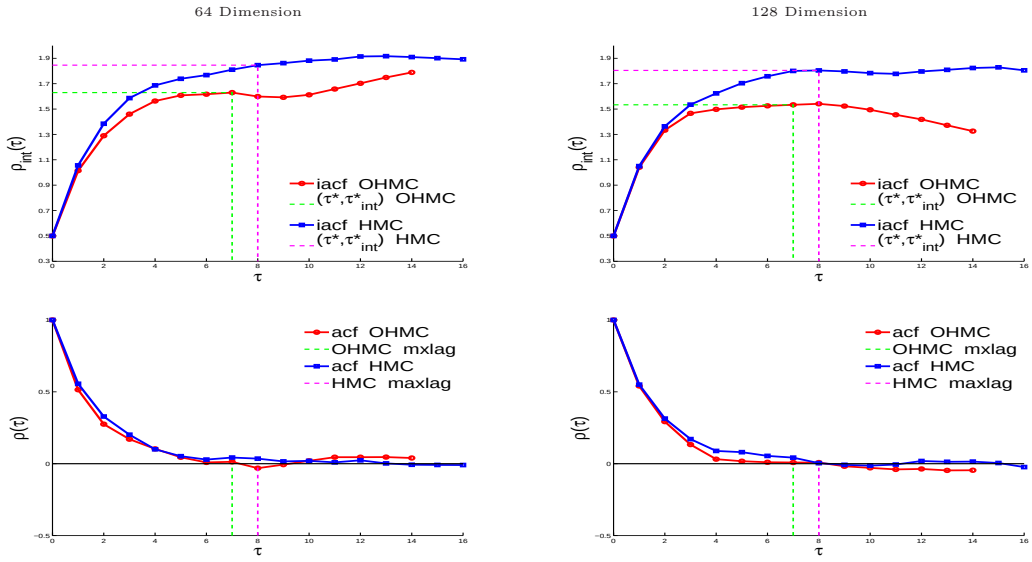


FIGURE 8.2: The integrated autocorrelation time and estimated value of τ_{int} for both OHMC and HMC algorithms are shown in the top row of the figure. The corresponding autocorrelation function for different lags is presented in the bottom row.

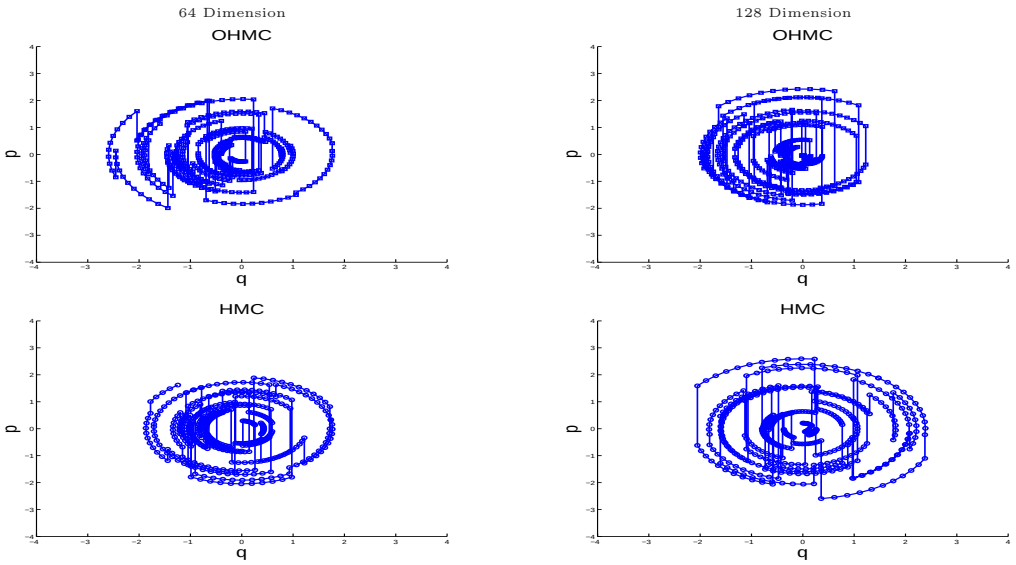


FIGURE 8.3: 500 trajectory points in the phase–space (q, p) for 64–dimensions (left column) and 128–dimensions (right column) from OHMC (top row) and HMC (bottom row).

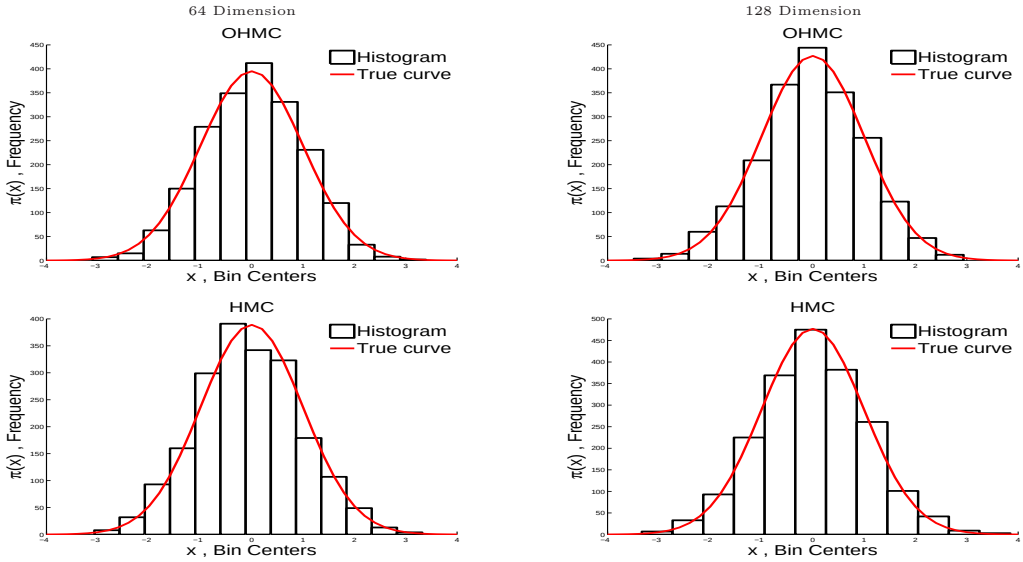


FIGURE 8.4: The histograms for OHMC (top row) and HMC (bottom plots) in 64 and 128–dimensions.

The number of independent samples in a chain of length N is given by $N/(2\tau_{int})$, where τ_{int} is the iacf. The chain generated by HMC has 614 independent samples, while there are 541 independent samples in the OHMC chain, taking into account the error of estimating the value of τ_{int} see Table 8.1. Looking at the trajectories and the histograms in Figures 8.3 and 8.4 respectively, for these algorithms, we observe a slight improvement in the OHMC algorithm over the HMC. The trajectories for OHMC are well concentrated in the region of high probability.

8.3.2 Comparing SVHMC and HMC algorithms

To compare the Störmer–Verlet discretization and the leapfrog scheme, we first need to optimize ϵ in HMC, and the starting value for the fictive variable ρ_0 in SVHMC. These can be obtained by running short chains of the algorithms with different values for these parameters, then by optimizing the efficiency, for example, we can find corresponding values for ϵ and ρ_0. We use the same number of simulation steps L.

We run 2000 iterations for each algorithm in 64 and 128–dimensions, and calculate the diagnostics criteria for these chains for the first parameter. The spectral analysis of these chains is shown in Figure 8.5. The flatter curve corresponds to the chain with better convergence properties.

From Table 8.2, the number of the effective independent samples for SVHMC is bout 1064 samples, while HMC has only about 541 samples. this means that we should generate twice the sample size of HMC to get the same performance as SVHMC. The ratio of the efficiency of

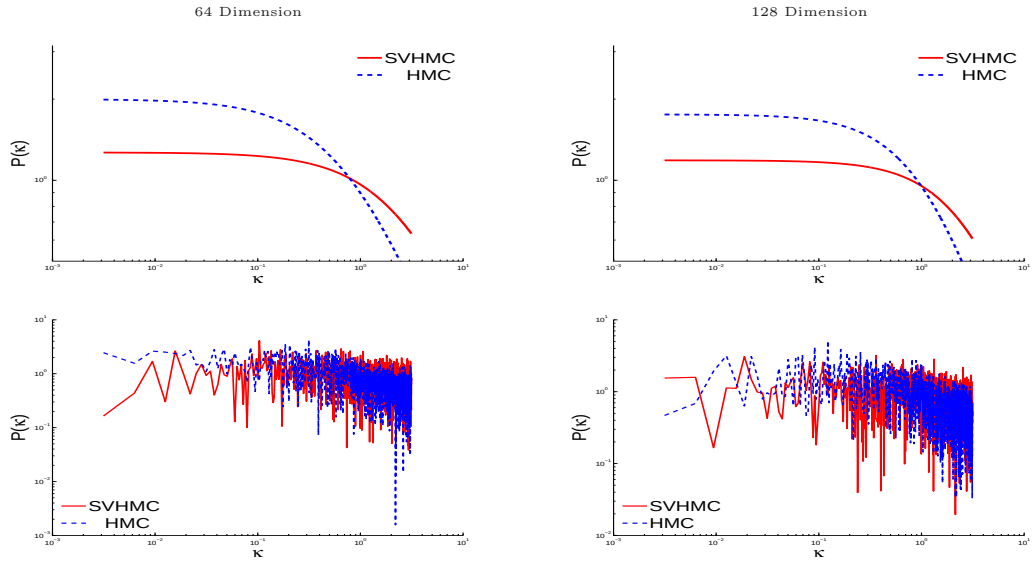


FIGURE 8.5: Discrete power spectral density (psd) in logarithmic scale (bottom row) with the best fit template (top row) for the chains generated by the SVHMC and HMC algorithm in 64 and 128–dimensions.

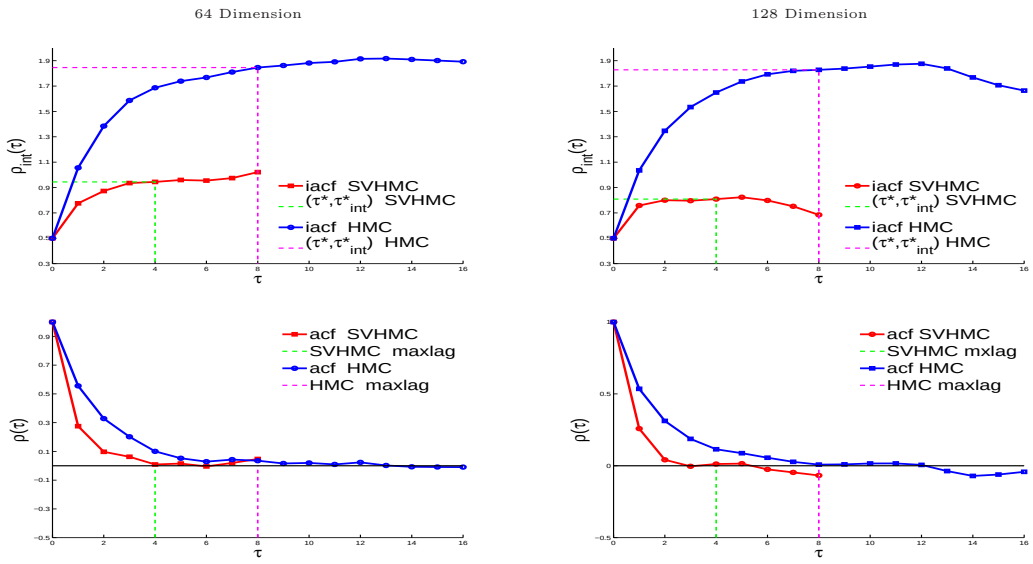


FIGURE 8.6: The integrated autocorrelation time τ_{int} and the autocorrelation function $\rho(\tau)$ for both SVHMC and HMC in 64 and 128–dimensions.

SVHMC to HMC algorithms about 1.6.

$$\frac{E_{SVHMC}}{E_{HMC}} \approx 1.6.$$

TABLE 8.2: Summary of diagnostics convergence tests for SVHMC and HMC algorithm in Gaussian distribution of 64 and 128–dimensions

	64 Dimension		128 Dimension	
	SVHMC	HMC	HMC	OHMC
Acceptance Rate	0.938500	0.985000	0.916000	0.975500
$P(0)$	2.2586970	3.560673	2.111930	3.125949
κ^*	3.1397510	0.812680	3.270751	1.145286
CPU time/Sec	669.3962000	557.381400	1568.014500	1117.776300
Efficiency E	0.4427331	0.280845	0.473500	0.319902
Convergence ratio r	0.0011293	0.001780	0.001055	0.001562
iacf τ_{int}	0.9440914	1.845748	0.856717	0.208802
Error of τ_{int}	0.0796167	0.212930	0.073130	1.804368
Error of estimation μ_1	0.0313669	0.043258	0.029080	0.043891
Estimation of m_1	0.0036804	−0.054868	0.034602	0.010457

For each trajectory, the momentum is drawn from the normal distribution (HMC) and the over–relaxed normal distribution (SVHMC). (The vertical jumps in plots of Figure 8.7) represent several steps along a trajectory of constant Hamiltonian value. The overall shape of the trajectories from SVHMC are more elliptical than HMC, which means that SVHMC visits more points in the high probability region in the phase–space than HMC.

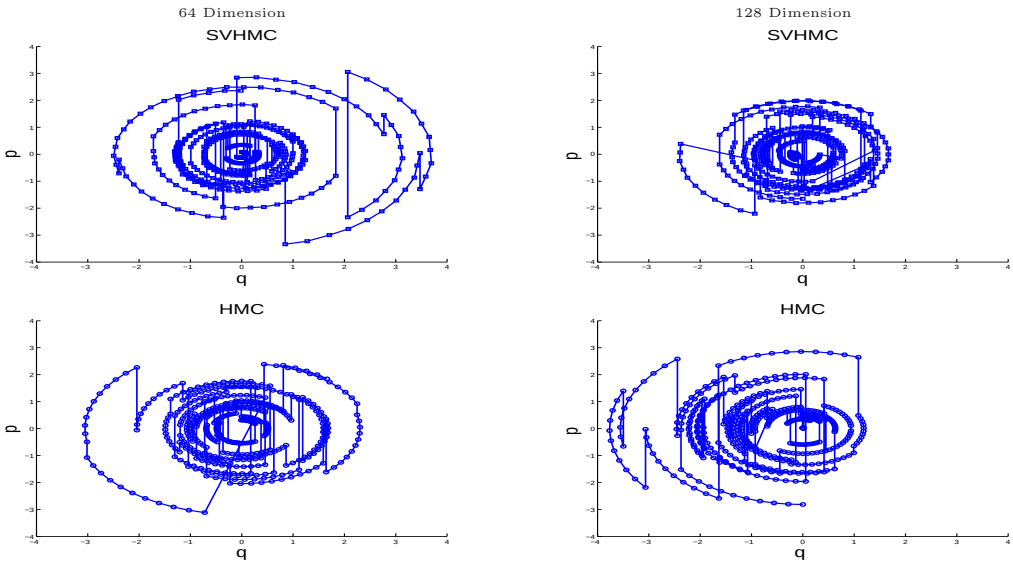


FIGURE 8.7: 500 trajectory points in the phase–space (q, p) for 64-dimensions (left column) and 128-dimensions (right column) from SVHMC (top row) and HMC (bottom row).

In addition, Figure 8.8 shows that the histograms for SVHMC match the marginal distribution of the first chain better than the HMC.

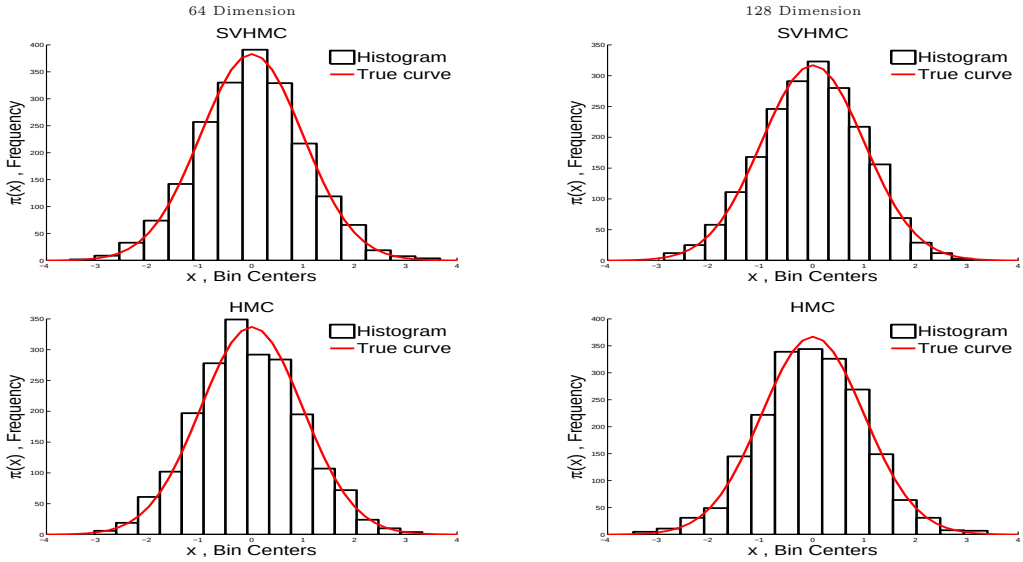


FIGURE 8.8: The histograms for SVHMC (top row) and HMC (bottom plots) for 64 and 128–dimensions.

Thus the SVHMC algorithm outperforms the classical HMC by having better convergence chain. The difference in the CPU time between SVHMC and HMC is largely due to the additional function evaluations in computing the fictive variable ρ_n. This difference can be reduced by running a shorter chain for SVHMC. The shorter chain will still have a better efficiency than HMC because the SVHMC algorithm will have more effective samples.

8.3.3 Comparing OSVHMC and SVHMC algorithms

Finally, we compare the hybrid of OHMC and SVHMC algorithm, i.e OSVHMC, to SVHMC to see the effect of the ordered over-relaxation on the Störmer–Verlet discretization.

TABLE 8.3: The convergence diagnostics and the efficiency criteria for OSVHMC and SVHMC.

	64 Dimension		128 Dimension	
	OSVHMC	SVHMC	OSVHMC	SVHMC
Acceptance Rate	0.921500	0.938500	0.916000	0.911000
$P(0)$	1.898972	2.258697	2.111930	2.180878
κ^*	4.886321	3.139751	3.270751	4.106211
CPU time/Sec	639.581200	669.396200	1568.014500	1626.932000
Efficiency E	0.526600	0.442733	0.473500	0.458530
Convergence ratio r	0.000949	0.001129	0.001055	0.001090
iacf τ_{int}	0.714289	0.944091	0.856717	0.810302
Error of the error	0.001119	0.001487	0.001379	0.001397
Error of estimation μ_1	0.026770	0.031366	0.029080	0.029453
Estimation of m_1	0.007290	0.003680	0.034602	−0.026153

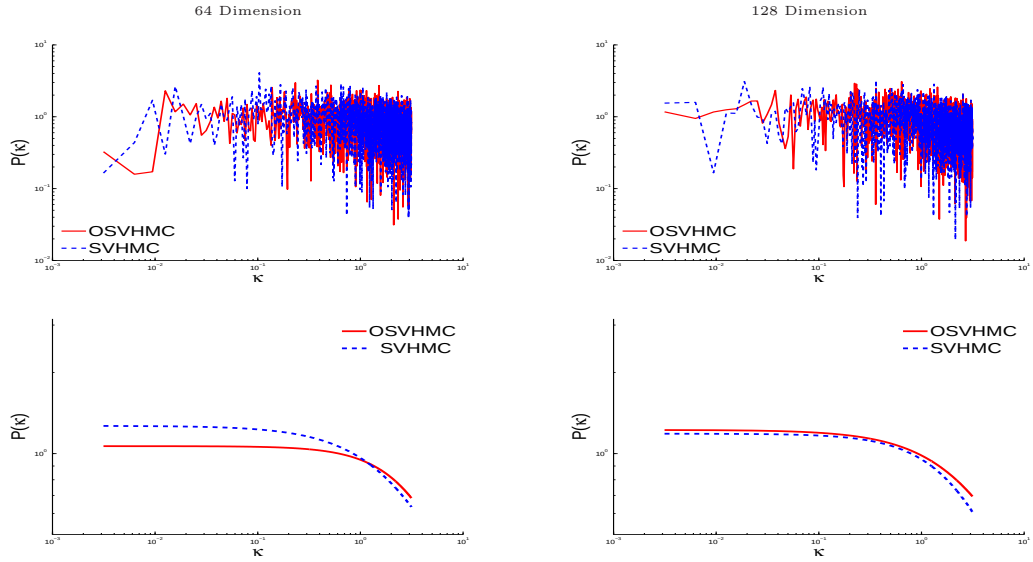


FIGURE 8.9: The power spectral density (psd) in logarithmic scale (bottom row), and approximated template curves defined by (7.18) for the chains generated by the SVHMC and HMC algorithm (top row), for 64-dimensions (left column) and 128-dimensions (right column).

Following the same arguments as in the previous sections, we observe from Figures 8.9, 8.10, 8.11, 8.12 and Table 8.3, that OSVHMC has better efficiency and converges faster than the SVHMC. Further, the number of effective samples in the OSVHMC are more than in the SVHMC.

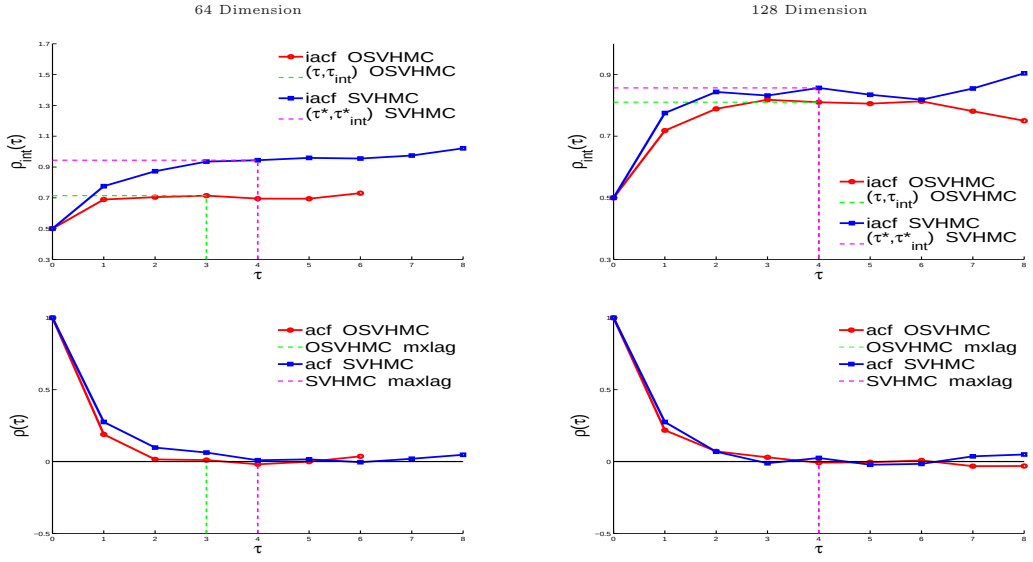


FIGURE 8.10: The integrated autocorrelation time and estimated value of τ_{int} for both OHMC and HMC algorithms are shown in the upper row of the figure. The corresponding autocorrelation functions are presented in the bottom row of this figure.

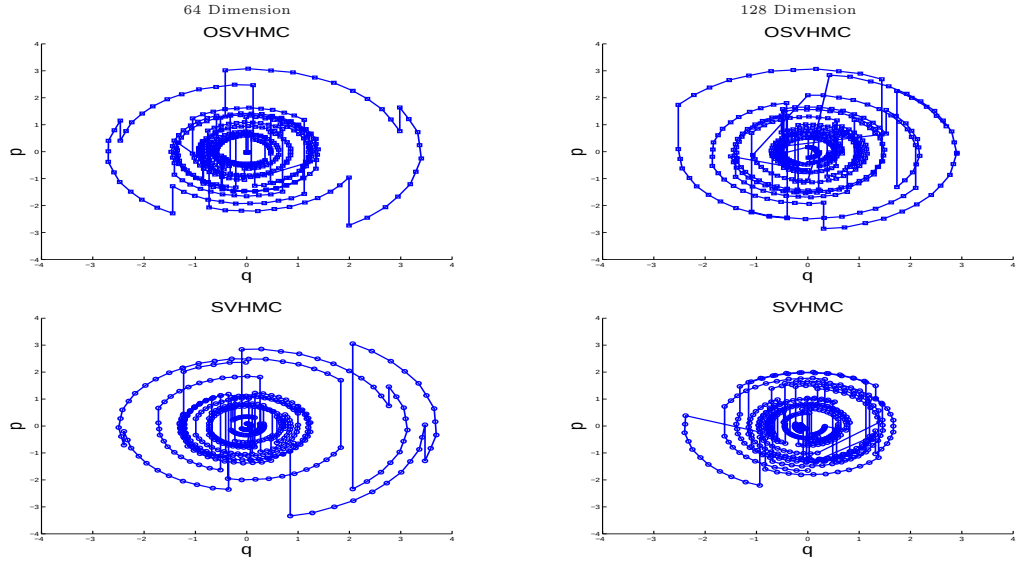


FIGURE 8.11: 500 trajectory points in the phase–space (q, p) for 64-dimensions (left column) and 128-dimensions (right column) for OSVHMC (top row) and SVHMC (bottom row).

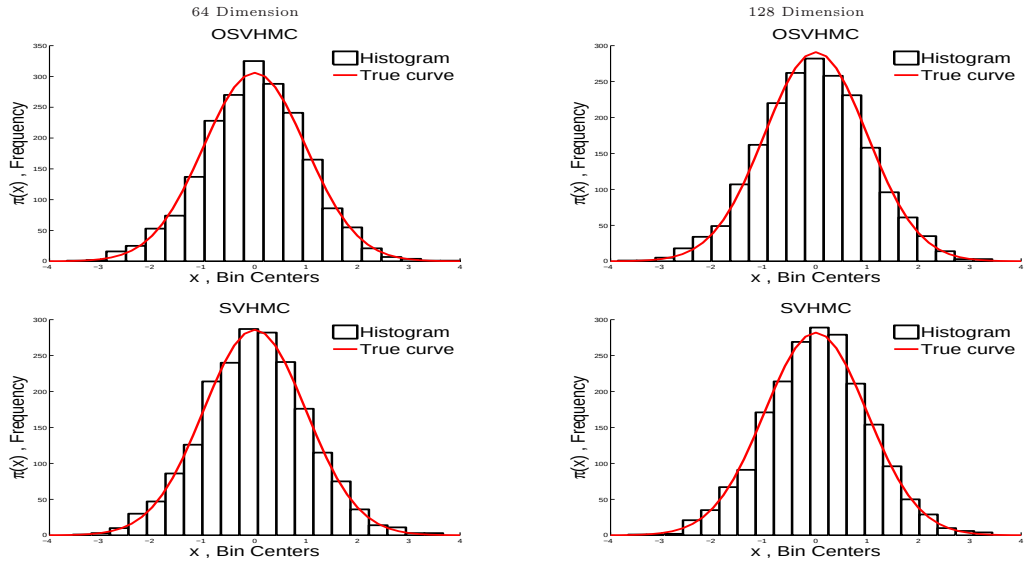


FIGURE 8.12: The histograms for OSVHMC (top) and SVHMC (bottom) for 64 and 128–dimensions.

8.4 Conclusion

The thesis has investigated methods to improve the Hamiltonian Monte Carlo (HMC) algorithm, by considering three techniques, which would enhance the performance of the algorithm. Firstly, by reducing the random walk in drawing the momentum variables, using ordered over–relaxation in the Gibbs sampling stage. This version, designated by OHMC, improves the number of effective samples by a factor $\sim 12\%$.

A second improvement involves reducing the error involved in simulating the Hamiltonian dynamics using the adaptive step–size with Störmer–Verlet discretization, which gives the SVHMC algorithm. The SVHMC algorithm outperforms the classical HMC algorithm with the leapfrog scheme by having $\sim 50\%$ more effective samples size. However, care must be taken when choosing the starting adaptive step–size in discretization of the simulation time.

Finally, the numerical experiments show that when ordered over–relaxation is applied to the SVHMC algorithm, the effect sample size is improved by $\sim 12\%$. This result is identical to applying ordered over–relaxation to the HMC algorithm.

Chapter 9

Suggestions for further works

Its is well established in the literature that MCMC algorithm propose small changes in the state vector at each iteration step. The result is:

(1) rare inter–modal moves,

(2) slow convergence of the chain, and

(3) high correlation between successive states.

The approach in the MSc thesis addresses points (1)–(2). Point (1) may be partially addressed by an approach based on using adaptive step–size. However, this remains to be proven.

When sampling from a multi–modal targets, it is desirable that the algorithm has the following properties:

(a) large proposal changes (not entirely random) in the state vectors are allowed,

(b) proposed new states are located in high–probability regions, and

(c) have high acceptance probabilities.

The HMC algorithm may boast good performance with respect to points (a)–(c). However its performance, when faced with multi–modal target distributions, this observation is based on preliminary studies, could be worse of than the classical MH algorithm.

One promising approach is to use the Mode Jumping Proposal, firstly suggested by [50], and which seeks to address points (a)–(c). above. The construction is based on firstly defining Q_0^{φ} and Q_1^{φ} as two proposal kernels on $\mathbb{R}^n$, with corresponding densities $q_0(y|x)$ and $q_1(y|x)$, respectively–(There are several methods for combining the proposal kernels into a single kernel Q on $\mathbb{R}^n$). The approach then follows a 3-step procedure:

Step (1) A move from $x \to T_0(x, \varphi) = x + \varphi$.

Step (2) Deterministic local optimization to determine the location of the minimum point $\mu(x+\varphi)$ of the energy function $U(x) = -\ln(\pi(x))$ in order to generate proposal state y. If $\mu(x)$ is the location of the minimum found with $x_{\text{start}} = x$, and $\Sigma(x) = \left[(\nabla^2 U)\mu(x)\right]^{-1}$, then $q_0^{\varphi}(y|x)$ is defined by (9.1), where $\mathrm{N}_n(\mu, \Sigma)(\cdot)$ is the pdf of a n-variate Gaussian with mean and covariance denoted by μ and Σ, respectively.

$$q_0^{\varphi}(y|x) = \mathrm{N}_n(\mu(T_0(x,\varphi)), \Sigma(T_0(x,\varphi)))(y). \tag{9.1}$$

Step (3) Deterministic local optimization to determine the location of the minimum point $\mu(y-\varphi)$. The choice of Q_1^{φ} is such that the reverse jump, from y to x, has high probability under the defined kernel. Thus if $\mu(T_0(x,\varphi))$ is relatively close to $T_0(x,\varphi)$, then $T_1(y,\varphi) = y - \varphi$ is most likely to be located within the same basin of attraction as the mode of x. Hence it reasonable to define $q_1^{\varphi}(x|y)$ by (9.2).

$$q_1^{\varphi}(x|y) = \mathrm{N}_n(\mu(T_1(y,\varphi)), \Sigma(T_y(y,\varphi)))(x). \tag{9.2}$$

Steps (1)–(2) above are designed to address the desirable characteristics of the chain, outlined in points (a)–(c). Incorporating the mode jumping into the general framework of the HMC algorithm implemented in this thesis could addresses the multi mode sampling drawback. An alternative approach has been proposed by [33].

Appendix A

Abstract accepted at Thiele conference

Abstract accepted at Efficient Monte Carlo: From Variance Reduction to Combinatorial Optimization A Conference on the Occasion of R.Y. Rubinstein's 70th Birthday. Sandbjerg Estate, Snderborg, Denmark 14-18 July 2008.

The Hamiltonian Monte Carlo Algorithm with Overrelaxation and Adaptive–Step Discretization– Numerical Experiments with Gaussian Targets

Mohammed Alfaki* Sam Subbey† Dag Haugland‡

April 11, 2008

Abstract

Using results from numerical experiments, we discuss the performance of the Hamiltonian Monte Carlo (HMC) algorithm with adaptive–step Störmer–Verlet discretization of the dynamic transitions, and ordered overrelaxation applied to the static transitions.

Our results show that the combined effect of overrelaxation and adaptive–step discretization results in an algorithm, which outperforms the classical Leapfrog HMC algorithm in sampling Gaussian targets with uncorrelated covariates.

We exemplify using Gaussian targets in 64 and 128 dimensions, and discuss the significance of our results in a more general context of parameter estimation involving high–dimensional Gaussian targets with uncorrelated covariates.

Keywords: Hamiltonian Monte Carlo, Overrelaxation, Symplectic integrator, Leapfrog, Störmer–Verlet, Gaussian targets, Parameter estimation.

*University of Bergen, Department of Informatics, email: mohammeda@ii.uib.no
†Inst. for Mar. Res., Bergen, Norway, email: samuels@imr.no
‡University of Bergen, Department of Informatics , email: dag@ii.uib.no

Appendix B

Abstract accepted at ECMOR XI conference

Abstract accepted at 11th European Conference on the Mathematics of Oil Recovery. 8 - 11 September 2008, Bergen, Norway

The Hamiltonian Monte Carlo Algorithm in Parameter Estimation and Uncertainty Quantification

Sam Subbey* Mohammed Alfaki† Dag Haugland‡

March 12, 2008

Abstract

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.

An example application to reservoir parameter estimation and uncertainty quantification is presented.

This paper differs from previous work in the following ways:

- Application of the HMC algorithm to parameter estimation and uncertainty quantification has not been reported in petroleum science literature. Previously reported traditional MCMC algorithms almost invariably suffer from inefficiency caused by the random walk nature of the Metropolis algorithm.
- We demonstrate improvement of the traditional HMC algorithm by application of a discretization scheme, which although reported in the Physics literature, has never been directly applied to the HMC algorithm.

1 Significance of Proposed Paper

This paper makes 3 distinct technical contributions to the knowledge base of the mathematics of oil recovery in the areas of

(1) (Assisted) History Matching:
Provides an efficient algorithms for rapid parameter estimation in history matching.

*Inst. for Mar. Res., Bergen, Norway, email: samuels@imr.no
†University of Bergen, Department of Informatics , email: mohammeda@ii.uib.no
‡University of Bergen, Department of Informatics , email: dag@ii.uib.no

(2) Uncertainty Quantification:
Facilitates rapid uncertainty quantification (especially in a Bayesian framework), where posterior distributions from which to sample are usually high-dimensional and complex. Usually in such cases, traditional MCMC techniques have slow convergence, low efficiency and are CPU intensive

(3) Stochastic Methods:
Applicable in other areas such as in rapid generation of stochastic realizations of porosity/permeability fields

Bibliography

[1] S. L. Adler. Over–relaxation Method for Monte Carlo Evaluation of the Partition Function for Multiquadratic Actions. *Physical Review D*, 23:2901–2904, 1981.

[2] H. C. Andersen. Molecular Dynamics Simulations at Constant Pressure and/or Temperature. *Journal of Chemical Physics*, 72:2384–2393, 1980.

[3] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier Academic Press, 2005.

[4] S. Blanes and C. J. Budd. Explicit Adaptive Symplectic (EASY) Integrators: a scaling invariant generalization of Levi–Civita and KS. *Celestial Mechanics and Dynamical Astronomy*, 89(4):383–405, 2004.

[5] M. P. Calvo, M. A. López-Marcos, and J. M. Sanz-Serna. Variable Step Implementation of Geometric Integrators. *Appl. Numer. Math.*, 28(1):1–16, 1998.

[6] P. Clifford. Lecture Notes of Mathematical Statistics. 2000.

[7] H. A. David and Herbert Aron David. *Order Statistics*. Wiley-Interscience, second edition, 1981.

[8] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195:216–222, 1987.

[9] J. Dunkley, M. Bucher, P. G. Ferreira, K. Moodley, and C. Skordis. Fast and Reliable MCMC for Cosmological Parameter Estimation. *arXiv,astro-ph.*, 0405462, 2004.

[10] A. Gelman, G. O. Roberts, and W. R. Gilks. Efficient Metropolis Jumping Rules. *Bayesian Statistics*, 5:599–607, 1996.

[11] W. R. Gilkks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, second edition, 1996.

[12] B. Goldman, M. Duncan, and J. Candy. Symplectic Integrators for Longterm Integrations in Celestial Mechanics. *Celestial Mechanics and Dynamical Astronomy*, 52(3):221–240, 1991.

[13] H. Goldstein. *Classical Mechanics*. Addison–Wesley, Reading, MA, 1980.

[14] A. Hajian. Efficient Cosmological Parameter Estimation with Hamiltonian Monte Carlo. *arXiv:*, 0608679v2 [astro-ph.], 2006.

[15] K. M. Hanson. Markov Chain Monte Carlo Posterior Sampling with Hamiltonian Methods. *IIn Proc. SPIE 4322*, Medical Image Processing, M. Sanka K. H. Hanson, eds.:456–467, 2001.

[16] K. M. Hanson, G. S. Gunningham, and R. J. Mckee. Uncertainty Assessment for Reconstruction Based on Deformable Geometry. *Int. J. Imaging Syst. Technol*, 8:506–516, 1997.

[17] E. Harier and G. Söderlind. Explicit, Time Reversible, Adaptive Step Size Control. *SIAM Journal of Scientific Computing*, 26(6):1838–1851, 2005.

[18] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57:97–109, 1970.

[19] T. Holder, B. Leimkuhler, and S. Reich. Explicit Variables Step-Size and Time-Reversible Integration. *Applied Numerical Mathematics*, 39(Issues 3-4):367–377, 2001.

[20] W. Huang and B. Leimkuhler. The Adaptive Verlet Method. *SIAM Journal on Scientific Computing*, 18:239–256, 1997.

[21] P. Hut, J. Makino, and S. McMillan. Building A Better Leapfrog. *The Astrophysical Journal*, 443:L93–l96, 2001.

[22] A. D. Kennedy, R. Edwards, H. Mino, and B. Pendleton. Tuning the Generalized Hybrid Monte Carlo Algorithm. *hep–lat/9509043*, 1995.

[23] V. Kolehmainen. *Novel Approaches to Image Reconstruction in Diffusion Tomography*. PhD thesis, 2001.

[24] A. Krakovská and S. Štolc. Fractal Complexity of EEG Signal. *Measurement Science Review*, 6(4):63–66, 2006.

[25] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge University Press, University of Leicester and Imperial College, London, 2004.

[26] D. Mackay. *Efficient Monte Carlo Methods*, chapter 30, pages 387–397. Cambridge University Press, 2003.

[27] W. L. Martinez and A. R. Martinez. *Computation Statistics Handbook with Matlab*. Chapman and Hall, 2002.

[28] K. Mosegaard and A. Tarantola. Monte Carlo Sampling of Solutions to Inverse Problems. *Geophysical Reseach*, 100, No.,B7:12431–12447, 1995.

[29] P. Müller. A Generic Approach to Posterior Integration and Gibbs Sampling. 1993.

[30] I. Murray. *Advances in Markov Chain Monte Carlo Methods*. PhD thesis, 2007.

[31] M. N. Rosenbluth N. Metropolis, A. W. Rosenbluth and E. Teller. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[32] R. M. Neal. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical report, University of Toronto, 1993.

[33] R. M. Neal. Sampling from Multimodal Distributions Using Tempered Transitions. Technical report, Department of Statistics, University of Toronto, 1994.

[34] R. M. Neal. Suppressing Random Walks in Markov Chain Monte Carlo Using Ordered Overrelaxation. Technical report, Department of Statistics, University of Toronto, 1995.

[35] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, New York, 1996.

[36] R. M. Neal. Suppressing Random Walks in Markov Chain Monte Carlo Using Ordered Overrelaxation, In M. I., Editor,. *Learning in Graphical Models*, pages 205–228, 1998.

[37] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2000.

[38] M. B. Priestley. *Spectral Analysis And Time Series*. Academic Press, 1996.

[39] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, second edition, 2000.

[40] K. Rossberg. *A first Course in Analytical Mechanics*. John Wiley & Sons, New York, 1983.

[41] J. M. Sanz-Serna and M. P. Calvo. *Numerical Hamiltonian Problems*. Chapman & Hall, London, 1994.

[42] M. B. Sevryuk. *Reversible Systems, Lecture Notes in Mathematics*, volume 1211. Springer-Verlag, 1986.

[43] D. Sigeti and W. Horsthemke. High–Frequency Power Spectra for Subject to Noise. *Physical Review A*, 35(5):2276–2282, 1987.

[44] A. Sokal. Monte Carlo Methods in Statistical Mechanics: Fundation and New Algorithm. *In Cours de Troisiéme Cycle de la Physique en Suisse Romande*, 1996.

[45] D. M. Stoffer. Variable Steps for Reversible Integration Methods. *Computing*, 55:1–22, 1995.

[46] A. Tarantola. *Inverse Problem Theroy and Methods for Model Parameter Estimation*. SIAM, 2005.

[47] A. Tarantola and K. Mosegaard. *Probabilistic Approach to Inverse Problems*, chapter 16, pages 237–265. Academic Press, 2002.

[48] H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, third edition, 1998.

[49] L. Tierney. Markov Chains for Exploring Posterior Distributions (with discussion). *Annals of Statistics*, 22:1701–1762, 1994.

[50] H. Tjelmeland and B. Hegstad. Mode Jumping Proposals in MCMC. Technical report, Norwegian University of Science and Technology, 1999.

[51] B. J. Torby. *Advanced Dynamics for Engineers*. Holt, R & W, 1984.

[52] C. R. Vogel. *Computation Methods for Inverse Problems*. Frontiers In Applied Mathematics, SIAM, Montana State University, 2002.

[53] B. Walsh. Markov Chain Monte Carlo and Gibbs Sampling. Lecture Notes for EEB 581. 2004.

[54] U. Wolff. Monte Carlo Errors with less Errors. *ELsevier, Computer Phyics Communications*, 156(2004):143–153, 2003.

[55] H. Yoshida. Construction of Higher Order Symplectic Integrators. *Phys. Lett. A*, 150:262–268, 1995.