UNIVERSITY OF CALIFORNIA,
IRVINE

Scalable Hamiltonian Monte Carlo via Surrogate Methods

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mathematics

by

Cheng Zhang

Dissertation Committee:
Professor Hongkai Zhao, Chair
Associate Professor Babak Shahbaba
Professor Jack Xin

2016
DEDICATION

To my grandpa,
grandma and my parents.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF ALGORITHMS</td>
<td>x</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>xi</td>
</tr>
<tr>
<td>CURRICULUM VITAE</td>
<td>xii</td>
</tr>
<tr>
<td>ABSTRACT OF THE DISSERTATION</td>
<td>xiv</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Outline and Contributions</td>
<td>6</td>
</tr>
<tr>
<td><strong>2 Markov Chain Monte Carlo</strong></td>
<td>9</td>
</tr>
<tr>
<td>2.1 Metropolis-Hastings Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Hamiltonian Monte Carlo</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 Hamiltonian Dynamics</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2 Leapfrog Scheme</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3 Model-based Hamiltonian Flow</td>
<td>17</td>
</tr>
<tr>
<td>2.2.4 Riemannian Manifold HMC</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Scalable MCMC</td>
<td>22</td>
</tr>
<tr>
<td>2.3.1 Stochastic Gradient MCMC</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2 Surrogate Method</td>
<td>27</td>
</tr>
<tr>
<td><strong>3 Precomputing Strategies</strong></td>
<td>30</td>
</tr>
<tr>
<td>3.1 Insights From an Illustrative Example</td>
<td>31</td>
</tr>
<tr>
<td>3.2 A General Framework</td>
<td>32</td>
</tr>
<tr>
<td>3.3 Grid Based Surrogate Methods</td>
<td>35</td>
</tr>
<tr>
<td>3.3.1 Naive Grid HMC</td>
<td>35</td>
</tr>
<tr>
<td>3.3.2 Sparse Grid HMC</td>
<td>38</td>
</tr>
<tr>
<td>3.4 Domain of Interest</td>
<td>45</td>
</tr>
<tr>
<td>3.5 Experiments</td>
<td>47</td>
</tr>
<tr>
<td>3.5.1 Logistic Regression</td>
<td>48</td>
</tr>
<tr>
<td>3.5.2 Banana-shaped Distribution</td>
<td>48</td>
</tr>
</tbody>
</table>
C Random Network Surrogate HMC 132
  C.1 Proof of Proposition 4.2 ............................................. 132

D Variational Hamiltonian Monte Carlo 135
  D.1 Proof of Theorem 5.1 .................................................. 135
  D.2 Proof of Theorem 5.3 .................................................. 136
  D.3 Proof of Proposition 5.1 ............................................. 136
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>2D Gaussian example: (a) the graph of the first component of $\nabla \mu U$. The function value at red point can be obtained by interpolation when the function values at three blue points are known. (b) HMC samples from the posterior distribution.</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>Force map of a logistic regression model</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Clenshaw-Curtis type sparse grids</td>
<td>40</td>
</tr>
<tr>
<td>3.4</td>
<td>Piecewise linear Nodal basis (a) and hierarchical functions (b) with support nodes $x^i_j \in X^i_\Delta, i = 1, 2, 3$ for the Clenshaw-Curtis grid</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>Piecewise linear interpolation: Nodal versus Hierarchical</td>
<td>42</td>
</tr>
<tr>
<td>3.6</td>
<td>Domains of interest using Laplace’s approximation for the logistic regression model</td>
<td>45</td>
</tr>
<tr>
<td>3.7</td>
<td>HMC vs GHMC: logistic regression</td>
<td>47</td>
</tr>
<tr>
<td>3.8</td>
<td>HMC vs GHMC: banana-shaped distribution</td>
<td>49</td>
</tr>
<tr>
<td>3.9</td>
<td>HMC vs sgHMC: Gaussian Process</td>
<td>51</td>
</tr>
<tr>
<td>3.10</td>
<td>HMC vs sgHMC: an elliptic PDE inverse problem</td>
<td>53</td>
</tr>
<tr>
<td>3.11</td>
<td>Comparing HMC and sgHMC under different dimensionalities on logistic regression models</td>
<td>54</td>
</tr>
<tr>
<td>3.12</td>
<td>Posterior samples for HMC, GHMC and GHMC-complete: logistic regression and banana-shaped distribution</td>
<td>57</td>
</tr>
<tr>
<td>3.13</td>
<td>The average log-likelihood on test data for a logistic regression model</td>
<td>59</td>
</tr>
<tr>
<td>3.14</td>
<td>Domains of Interest via Laplace’s approximation for the banana shaped distribution</td>
<td>60</td>
</tr>
<tr>
<td>3.15</td>
<td>Domains of Interest via early trajectories</td>
<td>60</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparing different surrogate approximations with an increasing number of observations $N = 10, 20, 40$ on target function $y = x^2/2$. The observation points are nested samples from the standard normal distribution. For FITC and random networks, we choose 5 inducing points and 5 hidden neurons respectively. FITC and random networks are all run 100 times and averaged to reduce the random effects on the results.</td>
<td>71</td>
</tr>
</tbody>
</table>
4.2 Comparing HMC and NNS-HMC based on a 2-dimensional banana-shaped distribution. The left panel shows the gradient fields (force map) for the original Hamiltonian flow (red) and the surrogate induced Hamiltonian flow (blue). The middle and right panel show the trajectories for HMC and NNS-HMC samplers. Both samplers start from the same point (red square) with same initial momentums. Blue points at the end of the trajectories are the proposals. The overall acceptance probability drops from 0.97 using HMC to 0.88 using NNS-HMC.

4.3 Comparing the efficiency of our random network surrogates and Gaussian process surrogates on a challenging 32 dimensional Gaussian target whose covariance matrix has an eigenvector \((1, 1, \ldots, 1)\top\) with a corresponding eigenvalue of 1.0, and all other eigenvalues are 0.01. We set the step size to keep the acceptance probability around 70% for HMC and use the same step size in all surrogate methods. For FITC and random networks, the number of inducing points and hidden neurons are all set to be 1000 to allow reasonably accurate approximation. We ran each algorithm ten times and plot the medians and 80% error bars.

4.4 Acceptance probability of surrogate induced Hamiltonian flow on simulated logistic regression models for different number of parameters, \(d\), and hidden neurons, \(s\). The step size is chosen to keep the acceptance probability around 70% for HMC. **Left:** Acceptance probability as a function of \(s\) (x-axis) and \(d\) (y-axis). **Middle:** Acceptance probability as a function of \(s\) for a fixed dimension \(d = 32\). **Right:** Acceptance probability as a function of \(d\) for a fixed \(s = 2000\).

4.5 Acceptance probability of the surrogate induced Hamiltonian flow based on a simulated logistic regression models with dimension \(d = 32\). **Left:** Acceptance probability as a function of the number of hidden neurons \(s\) (x-axis) and the number of training points \(N\) (y-axis). **Middle:** Acceptance probability as a function of \(N\) for a fixed \(s = 1000\). **Right:** Acceptance probability as a function of \(s\) for a fixed \(N = 1600\).

4.6 HMC vs RNS-HMC: Comparing one- and two-dimensional posterior marginals of \(\beta_1, \beta_{11}, \beta_{21}, \beta_{31}, \beta_{41}\) based on the logistic regression model with simulated data.

4.7 Median acceptance rate of ARNS-HMC along with the corresponding 90% interval (shaded area). The red line shows the average acceptance rate of standard HMC.

4.8 Relative error of mean as a function of running time.

5.1 Approximate posteriors for a varying number of hidden neurons. Exact posterior at bottom right.

5.2 KL-divergence and score matching squared distance between the surrogate approximation and the exact posterior density using an increasing number of hidden neurons.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>RMSE approximate posterior mean as a function of the number of likelihood evaluations for difference variational Bayesian approaches and our Variational HMC algorithm.</td>
</tr>
<tr>
<td>5.4</td>
<td>Final error of logistic regression at time T versus mixing rate for the mean (top) and covariance (bottom) estimates after 300 (left) and 3000 (right) seconds of computation. Each algorithm is run using different setting of parameters.</td>
</tr>
<tr>
<td>5.5</td>
<td>Convergence of Amari distance on the MEG data for HMC, SGLD and our Variational HMC algorithm.</td>
</tr>
</tbody>
</table>

111

113

115
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Comparing HMC with GHMC using a logistic regression model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>48</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparing HMC with GHMC using a banana-shaped distribution model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>50</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparing HMC with sgHMC using a Gaussian process model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>51</td>
</tr>
<tr>
<td>3.4</td>
<td>Comparing HMC with sgHMC using an elliptic PDE inverse problem. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>54</td>
</tr>
<tr>
<td>3.5</td>
<td>Comparing HMC with GHMC using a logistic regression model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>58</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparing HMC with GHMC using a banana-shaped distribution model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>58</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparing the algorithms using logistic regression models and an elliptic PDE inverse problem. For each method, we provide the acceptance probability (AP), the CPU time (s) for each iteration and the time-normalized ESS</td>
<td>86</td>
</tr>
</tbody>
</table>
## LIST OF ALGORITHMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Metropolis-Hastings Algorithm</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Hamiltonian Monte Carlo</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Naive Grid HMC</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>Sparse Grid HMC</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Extreme Learning Machine</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Random Network Surrogate HMC</td>
<td>78</td>
</tr>
<tr>
<td>4.3</td>
<td>Adaptive RNS-HMC</td>
<td>83</td>
</tr>
<tr>
<td>5.1</td>
<td>Variational Hamiltonian Monte Carlo</td>
<td>107</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

First, I would like to express my greatest appreciation and thanks to my advisor Professor Hongkai Zhao, for his insightful advice and tremendous support throughout my graduate career. Along the journey of my research, Professor Zhao has been a great mentor, teacher and friend, who not only pointed out important directions, offered countless support, but also gave me a great amount of freedom to explore my interests freely. I have been greatly benefited from the enormous discussions with Professor Zhao, when he can always rapidly reinterpret my questions in an intuitive manner and suggest ways to solve them. Moreover, I also greatly appreciate Professor Zhao’s extremely patient guidance and help on presentation and writing skills, as well as other perspectives essential for success beyond research. Without him, this thesis and most of my papers would have been impossible. Professor Zhao’s rigorous scientific attitude, humble personality, and commitment to perfection have made a profound impact on me and have encouraged me to do the same.

Second, I am also grateful to the other two members of my dissertation committee, Professor Babak Shahbaba and Professor Jack Xin. My current research started from a joint group meeting between Mathematics and Statistics departments, where Professor Shahbaba provided me a first research opportunity in computational statistics, introducing me into the area of scalable Bayesian inference. From then on, we have been extensively collaborated; in fact, the main body of this thesis is based on our joint work. It is truly a blessing to work with him who lead me into the field of Statistics, taught me essential techniques from his own experience and encouraged me to explore new exciting topics. I want to thank Professor Jack Xin for his intuitive physical and numerical insights which inspire many ideas in our work. I deeply appreciate Professor Shahbaba’s and Professor Xin’s constant encouragements and their generous help over the years.

Third, I have also greatly benefited from my interaction and collaboration with other professors at UCI, including Professor Long Chen, Jeffrey Streets, Yaming Yu, Ombao Hernando and Dan Gillen, to name a few. Of course, I would like to express my thankfullness to my peers who have helped to shape and enrich my graduate experience. I want to thank Rongjie Lai, a former Postdoc in Professor Zhao’s group, for his guidance and useful suggestions on the road to be an independent researcher. Thanks to Shiwei Lan and Alex Vandenberg-Rodes in Professor Shahbaba’s group for inspiring discussions and collaborations. I would also like to thank all members in the GAMS group, as well as all of my Chinese friends over the years.

Finally, I would like to thank my family for a lifetime of love and support. I owe an enormous debt of gratitude to my parents, for their understanding and constant encouragement, especially during strenuous times.

My graduate research and this thesis have been supported in part by National Science Foundation (NSF) grants DMS-1418422 and DMS-1622490.
CURRICULUM VITAE

Cheng Zhang

EDUCATION

Doctor of Philosophy in Mathematics 2016
University of California, Irvine

Master of Science in Mathematics 2011
Peking University

Bachelor of Science in Mathematics 2008
Peking University

RESEARCH EXPERIENCE

Graduate Research Assistant 2011–2016
University of California, Irvine

TEACHING EXPERIENCE

Teaching Assistant UC Irvine, California
Numerical Analysis Winter, 2016
Multivariable Calculus Spring, 2015
Linear Algebra Spring, 2015

SELECTED AWARDS

Outstanding graduates, School of Mathematical Sciences 2011

ACADEMIC REVIEWING

Bayesian Analysis 2016
Inverse Problems in Science and Engineering 2015
REFEREED JOURNAL PUBLICATIONS


ABSTRACT OF THE DISSERTATION

Scalable Hamiltonian Monte Carlo via Surrogate Methods

By

Cheng Zhang

Doctor of Philosophy in Mathematics

University of California, Irvine, 2016

Professor Hongkai Zhao, Chair

Markov chain Monte Carlo (MCMC) methods have been widely used in Bayesian inference involving intractable probabilistic models. However, simple MCMC algorithms (e.g., random walk Metropolis and Gibbs sampling) are notorious for their lack of computational efficiency in complex, high-dimensional models and poor scaling to large data sets. In recent years, many advanced MCMC methods (e.g., Hamiltonian Monte Carlo and Riemannian Manifold Hamiltonian Monte Carlo) have been proposed that utilize geometrical and statistical quantities from the model in order to explore the target distribution more effectively. The gain in the efficacy of exploration, however, often comes at a significant computational cost which hinders their application to problems with large data sets or complex likelihoods.

In practice, it remains challenging to design scalable MCMC algorithms that can balance computational complexity and exploration efficacy well. To address this issue, some recent algorithms rely on stochastic gradient methods by approximating full data gradients using mini-batches of data. In contrast, this thesis focuses on accelerating the computation of MCMC samplers based on various surrogate methods via exploring the regularity of the target distribution.

We start with a precomputing strategy that can be used to build efficient surrogates in relatively low-dimension parameter spaces. We then propose a random network surrogate
architecture which can effectively capture the collective properties of large data sets or complex models with scalability, flexibility and efficiency. Finally, we provide a variational perspective for our random network surrogate methods and propose an approximate inference framework that combines the advantages of both variational Bayes and Markov chain Monte Carlo methods. The properties and efficiency of our proposed methods are demonstrated on a variety of synthetic and real-world data problems.
Chapter 1

Introduction

Bayesian statistics has provided a principled and robust framework to create many important and powerful data analysis methods over the past several decades. Given a probabilistic model for the underlying mechanism of the observed data, Bayesian methods properly quantify uncertainty and reveal the landscape or global structure of the parameter space of target distribution. While conceptually simple, exact inference in many Bayesian models is often intractable in that no closed form solution is available. Therefore, in practice, people often resort to approximation methods among which variational Bayes (VB) (Jordan et al., 1999; Jaakkola, 2000) and Markov chain Monte Carlo (MCMC) (Metropolis et al., 1953; Neal, 1993; Robert and Casella, 2004) are the two most prominent choices.

Variational Bayesian inference (Jordan et al., 1999; Wainwright and Jordan, 2008) typically utilizes a parametrized distribution to approximate the target posterior distribution. By searching over a flexible family of distributions (often with specific forms or structures) to find a closest approximation to the target distribution, it transforms Bayesian inference problem into an optimization problem which can be typically solved by iteratively reducing their distance (usually measured by the Kullback-Leibler (KL) divergence) with respect to
the variational parameters. Variational Bayesian method can also be viewed as a natural extension of the classical Expectation-Maximization (EM) algorithm (Dempster et al., 1977) from a deterministic approach of finding the Maximum A Posterior (MAP) to a probabilistic approach of finding an approximation to the target posterior distribution for statistical inference/estimation (Neal and Hinton, 1998). As a byproduct, it also provides the optimal lower bound for the marginal likelihood (model evidence), which has been used extensively in Bayesian model selection. Further exploration on this method includes Beal and Ghahramani (2002); Girolami (2001); Xing et al. (2003); Bishop et al. (2003), to name a few.

Unlike variational Bayes, the MCMC approach is based on drawing a series of correlated samples by constructing a Markov chain with guaranteed convergence to the target distribution. Therefore, MCMC methods are asymptotically unbiased. Despite a high computational cost, MCMC often provides better approximations and therefore maintains a simple but powerful tool to perform Bayesian inference (Metropolis et al., 1953; Hastings, 1970; Geman and Geman, 1984; Robert and Casella, 2004). This dissertation will mainly focus on MCMC methods.

To guarantee the convergence to the target distribution, the Markov chain in MCMC should be specifically designed to meet the following two conditions: (i) The Markov chain is aperiodic and irreducible; (ii) The equilibrium distribution of the Markov chain is the target distribution. Fortunately, the above conditions could be automatically met if we choose a reversible transition kernel that satisfies the detailed balance condition. MCMC was first introduced by Metropolis et al. (1953) to simulate the distribution of states for a system of idealized molecules where a symmetric proposal distribution was used to suggest a possible move followed by an acceptance-rejection step. A natural choice of such proposals is Gaussian distribution centered at the current state, and the resulting MCMC sampler is called Random Walk Metropolis (RWM). Hastings (1970) later extended the algorithm to allow asymmetric proposals. Note that Gibbs sampling (Geman and Geman, 1984) can also be
seen as a special cyclic Metropolis-Hastings (M-H) algorithm where proposals are generated by sampling from the full conditional distributions, and parameters are updated sequentially, with guaranteed acceptance (Andrieu et al., 2003). More advanced variants of M-H algorithms are proposed by Gilks et al. (1995); Atchade and Perron (2005); Holden and Holden (2009); Giordani and Kohn (2010).

The most critical step in developing an efficient MCMC algorithm is the design of transition kernel. Ideally, we would like to design an MCMC algorithm to sample independent draws from the target distribution. The perfect scenario for the transition kernel, therefore, is to generate distant proposals, which can be regarded as independent from the current state, with high acceptance rate. This is extremely challenging for simple MCMC methods (e.g., RWM, Gibbs sampling) due to their random walk nature of movement, especially in complex, high-dimensional models. As a result, these methods may suffer from slow mixing and require a long time to converge to the target distribution.

Many attempts have been made to improve the mixing rate and facilitate convergence in MCMC. The slice sampling proposed by Neal (2003) introduced an auxiliary variable to represent a horizontal “slice” of the target distribution which opens a large sampling domain and thereby allows large moves in the parameter space. One of its generalizations, elliptical slice sampling (Murray et al., 2010), has shown significant improvement on efficiency for performing inference in models with multivariate Gaussian priors compared to simple MCMC methods that are commonly used.

Instead of explicitly specifying a large sampling domain, Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2011) uses an auxiliary momentum variable to construct a Hamiltonian flow which explores contours of constant energy in phase space. By simulating the Hamiltonian dynamics, distant proposals can be generated with high acceptance rate since the Hamiltonian flow preserves the total energy. Guided by the geometric information of the target distribution, HMC reduces the random walk behaviors of simple MCMC methods,
allowing for more efficient exploration of the parameter space. A complete introduction of HMC can be found in (Neal, 2011). The performance of HMC could be highly sensitive to two user-specified parameters: the trajectory length and step size. Hoffman and Gelman (2011); Wang et al. (2013) provided ways that can automatically adjust these hyper-parameters. Although HMC explores the parameter space more efficiently than simpler MCMC methods, it does not fully exploit the structure (i.e., geometric properties) of parameter space since dynamics are defined over Euclidean space. To address this issue, Girolami and Calderhead (2011) proposed a new method, called Riemannian Manifold HMC (RMHMC), that uses the Riemannian geometry of the parameter space (Amari and Nagaoka, 2000) to improve standard HMC’s efficiency by automatically adapting to the local structures. See Beskos et al. (2011); Lan et al. (2012); Shahbaba et al. (2014); Betancourt et al. (2014) for more recent work on HMC.

The high exploration efficiency of HMC came at a computational price. As modern data analysis usually involves computation with large scale data sets, the required computation of geometrical information in HMC and its variants, which typically need full scans of the entire data set, becomes a major bottleneck for more general applications. In recent years, stochastic gradient MCMC algorithms have been proposed and proven quite useful in scaling their counterparts to large data sets (Welling and Teh, 2011; Ahn et al., 2012; Chen et al., 2014; Ding et al., 2014; Ma et al., 2015). The key idea of these methods stems from stochastic gradient descent, where noisy estimates of the gradient based on small subsets of the data are utilized to allow for efficient optimization with massive data. However, the noise introduced by subsampling could lead to non-ignorable loss of accuracy, which in turn exerts a substantially negative impact on exploration efficiency (Betancourt, 2015). Therefore, in practice, it remains a challenge to design scalable MCMC algorithms that can find a good balance between computational complexity and exploration efficiency.

As an alternative approach to stochastic gradient MCMC methods, surrogate transition
method (Neal, 1996; Liu, 2001) aims at finding computationally cheaper surrogate functions to substitute the expensive target potential energy functions. Rasmussen (2003); Lan et al. (2015); Strathmann et al. (2015) use Gaussian processes or reproducing kernel Hilbert space (RKHS) to approximate the energy functions based on information gathered from pre-convergent samples. By exploiting smoothness or regularity in parameter space, which is true for most statistical models, surrogate method reduces the computation cost while maintaining comparable exploration efficacy. However, the usefulness of these methods is often limited to moderate dimensional problems by the cost of acquiring such surrogates with desired approximation accuracy.

This thesis develops a series of surrogate methods that can be potentially generalized to higher dimensional problems and investigates their properties and efficiency on a variety of synthetic and real-world data sets. In addition, we explore the connection between variational Bayesian inference and surrogate based MCMC methods, developing techniques that can combine the advantages of both approaches, and are applicable to design efficient approximate inference algorithms.

In particular, we propose precomputing strategies that can be used to build efficient surrogates in relative low-dimension parameter spaces. We then propose a random bases surrogate architecture which can provide effective approximation of the probabilistic model based on the collective behavior of the massive data. The randomized nonlinear basis functions combined with the computationally efficient learning process can incorporate correct criteria for an efficient implicit subsampling, resulting in both flexible and scalable approximations.

Finally, we provide a general variational framework for the random bases surrogate method and derive an efficient approximate inference method that combines the advantages of variational Bayes and MCMC algorithms together. The resulting algorithm is simple, efficient and significantly improves the performance of its VB and MCMC counterparts. We demonstrate the properties and efficiency of our algorithms on a variety of synthetic and real-world
1.1 Outline and Contributions

The thesis is organized as follows. Chapter 2 presents an overview of Markov chain Monte Carlo methods, including recent progresses on scalable MCMC: stochastic gradient MCMC methods and surrogate methods. Chapters 3-5 consist of the novel contributions of the thesis, and Chapter 6 concludes and discusses some open directions. The appendices provide additional proofs and derivations. In more details:

Chapter 2 provides an overview of Markov chain Monte Carlo methods. We first introduce the fundamental Metropolis-Hastings algorithm which provides a general framework for all MCMC methods. We then introduce Hamiltonian Monte Carlo, a state-of-the-art MCMC method, and its variants. We briefly review some of the recent progresses on scalable MCMC methods in the last two sections.

Chapters 3-5 describe the main contributions of this thesis.

Chapter 3 presents precomputing strategies that can be used to build efficient surrogates in relatively low-dimensional parameter spaces. These efficient surrogates, then, are utilized to speed up Hamiltonian Monte Carlo methods. We start with a toy example that motivates our idea. We then provide two practical implementations that work for problems of different dimensionality. Specific contributions include:

- We provide a framework where efficient numerical approximation methods can be used to design scalable MCMC algorithms.
- We prove that the effectiveness of the surrogate can be controlled by the approximation quality in terms of maximum norm.
Chapter 4 proposes a new surrogate method for Hamiltonian Monte Carlo which is based on random nonlinear bases, along with efficient learning algorithms. The randomized nonlinear basis functions combined with the computationally efficient learning process can incorporate correct criteria for an efficient implicit subsampling resulting in both flexible and scalable approximation, which provides a faster alternative to the GPs based surrogate methods. In detail:

- We propose a novel surrogate method that has the potential to generalize to problems of higher dimensionality than that other current surrogate methods can handle.

- We show that our proposed method is related to and can be extended to other surrogate functions such as generalized additive models and Gaussian process models by constructing the surrogate functions using different bases and optimization processes.

- Our method can be easily extended to other geometrically motivated methods such as Riemannian Manifold Hamiltonian Monte Carlo.

- For problems with a limited time budget, we propose an adaptive version of our method that substantially reduces the required number of training points. This way, the random network surrogate function could be utilized earlier and its approximation accuracy could be improved adaptively as more training points become available.

- We show that theoretically the learning procedure of our surrogate function is asymptotically equivalent to potential matching, which is itself a novel distribution matching strategy for unnormalized densities.

Chapter 5 provides a general variational framework for the random network surrogate method proposed in Chapter 4. Base on this, we derive an efficient approximate inference method that combines the advantages of variational Bayes and MCMC algorithms together. The key idea is to integrate fast variational approximation into the sampling procedure
so that the overall computational complexity can be reduced. The main contributions are summarized as follows:

- We derive an efficient approximate inference method that combines the advantages of variational Bayes and MCMC algorithms together.
- We derive a general variational framework for the random network surrogate method, which further reduces the computation cost by allowing to use the computationally fast surrogate in the M-H correction step in HMC. The modified HMC sampler will converge to the best approximation from an exponential family with pre-specified random sufficient statistics. This variational perspective distinguishes our approach from the existing surrogate methods on accelerating HMC.
- We derive a new training procedure for the random network surrogate, based on our new variational framework. The random network surrogate is trained by minimizing a squared distance between the gradient of the surrogate and the gradient of the target (log-posterior), a procedure that resembles score matching. In contrast to score matching, the new training procedure mitigates the restriction on the training data by including the information from the target distribution explicitly.

Chapter 6 concludes and discusses some open directions for future research.
Chapter 2

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a popular approximate inference method originated from the landmark paper by Metropolis et al. (1953), where the fundamental idea of evolving a Markov chain to sample from a target distribution was introduced and used to simulate the distribution of states for a system of idealized molecules. From then on, this simple and powerful idea, later known as the Metropolis algorithm, together with its variations and extensions have played a significant role in statistics, economics, physics and computer sciences over the past several decades.

Theoretically speaking, MCMC algorithms can be used to generate random samples from any target distributions known up to a normalizing constant. However, the resulting samples are often highly correlated and the estimates obtained from these samples tend to have greater variances than those obtained from independent samples. To overcome these limitations, many advanced MCMC methods have been proposed which utilize geometrical information from the target distribution to facilitate convergence and sampling efficiency. The gain in the efficiency of sampling, however, often comes at a significant computational price which hinders their application to problems with large data sets or complicated likelihoods.
Therefore, it remains challenging to design scalable MCMC algorithms that can strike a good balance between computational complexity and sampling efficiency.

In this chapter, we provide background about Markov chain Monte Carlo methods and some of its recent advances on sampling efficiency and scalability. Section 2.1 presents the fundamental Metropolis-Hasting algorithm which is the cornerstone of all MCMC methods. Section 2.2 then introduces one of the state-of-the-art MCMC algorithms, the celebrated Hamiltonian Monte Carlo algorithm, and its variants. In section 2.3, we discuss some of the recent progress on scalable MCMC methods.

2.1 Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm is the most popular MCMC method (Metropolis et al., 1953; Hastings, 1970). As shown in later sections, most practical MCMC algorithms can be viewed as special cases or extensions of this algorithm.

Suppose that the target distribution is $\pi(\cdot)$. We want to simulate a Markov chain in the state space $\theta$ so that the equilibrium distribution of this chain is the target distribution $\pi$. To this end, we need to derive a transition probability (kernel) $T(\cdot|\cdot)$ so that the equilibrium can be reached. A sufficient, but not necessary condition, to ensure that $\pi$ is the desired stationary distribution is the following detailed balance condition (Andrieu et al., 2003):

$$\pi(\theta)T(\theta'|\theta) = \pi(\theta')T(\theta'|\theta)$$

(2.1)

Given the current state $\theta$, the M-H algorithm proposes a candidate state $\theta^*$ according to some tractable proposal distribution $q(\theta^*|\theta)$. The Markov chain then accepts the proposal
Algorithm 2.1 Metropolis-Hastings Algorithm

**Input:** Initialize $\theta^{(0)}$, transition kernel $q(\cdot|\cdot)$

for $n = 0$ to $N - 1$ do

Sample $u \sim U_{[0,1]}$

Propose $\theta^* \sim q(\theta^*|\theta^{(n)})$

if $u < \alpha(\theta^*, \theta^{(n)}) = \min \left\{ 1, \frac{\pi(\theta^*)q(\theta^{(n)}|\theta^*)}{\pi(\theta^{(n)})q(\theta^*|\theta^{(n)})} \right\}$ then

$\theta^{(n+1)} = \theta^*$

else

$\theta^{(n+1)} = \theta^{(n)}$

end if

end for

$\theta^*$ with the following probability

$$\alpha(\theta, \theta^*) = \min \left\{ 1, \frac{\pi(\theta^*)q(\theta|\theta^*)}{\pi(\theta)q(\theta^*|\theta)} \right\} \quad (2.2)$$

, otherwise it remains at $\theta$. The detailed balance condition (2.1) can be easily verified (Andrieu et al., 2003). We summarize the Metropolis-Hastings algorithm in Algorithm 2.1

When the proposal distribution $q(\cdot|\cdot)$ is symmetric, the Metropolis-Hastings algorithm reduces to the Metropolis algorithm. One such choice is the Gaussian random perturbation, $q(\theta^*|\theta) \sim N(\theta, \sigma^2I)$, and the resulting MCMC sampler is called Random Walk Metropolis (RWM). Note that Gibbs sampling (Geman and Geman, 1984) can also be viewed as a special case of MCMC algorithms with guaranteed acceptance (Andrieu et al., 2003).

The sampling efficiency of the M-H algorithm is largely determined by the proposal distribution $q(\cdot|\cdot)$. It is well known that simple proposal kernels (e.g., those used in RWM and Gibbs sampling) can be very inefficient for complex, high-dimensional distributions: successive samples may exhibit high autocorrelation, due to the random walk nature of the movement. As a result, these methods may suffer from slow mixing and require a long time to converge to the target distribution.
2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) was first proposed by (Duane et al., 1987) as a combination of the MCMC and molecular dynamics approaches. The statistical applications of HMC began with Neal (1996) where it was used for neural network models. Introducing an auxiliary momentum variable, HMC reduces the random walk behavior by proposing states following a Hamiltonian flow which preserves the target distribution. By leveraging the geometric information of the target distribution, e.g., the gradient, HMC is able to generate distant proposals with high acceptance probabilities, enabling more efficient exploration of the parameter space than simple MCMC methods that use standard random-walk proposals.

2.2.1 Hamiltonian Dynamics

Hamiltonian dynamics is a classic mechanics where the motion of a particle in a closed system can be described by a set of differential equations (Hamilton equations) according to the law of conservation of energy. The basic idea behind Hamiltonian Monte Carlo is that one can simulate a Hamiltonian dynamical system to generate trail moves in a MCMC sampler. Guided by the Hamiltonian dynamics, the generated proposals follow the target distribution more closely, and therefore suppress the diffusive behavior of simple random-walk proposals. In what follows, we give a brief overview of Hamiltonian dynamics and its properties. One can find a more detailed review in Neal (2011).

Consider a hockey puck sliding over a frictionless surface of varying height. The potential energy is based on the height of the surface at the current position of the puck, \( \theta \in \mathbb{R}^d \), and we denote it by \( U(\theta) \). The kinetic energy is based on the momentum of the puck, \( r \in \mathbb{R}^d \) and its Mass, \( M \), and generally is assumed to have a quadratic form \( K(r) = \frac{1}{2} r^\top M^{-1} r \). As the puck moves on an positive slope, the kinetic energy decreases as the potential energy...
increases. The remaining momentum keeps pushing the puck upward until the kinetic energy becomes zero. The puck then slides back down the hill with kinetic energy increasing and potential energy decreasing. The total energy of the this dynamical system is conserved, which is known as the *Hamiltonian*.

**Definition 2.1 (Hamiltonian).** The Hamiltonian $H$ is defined as the total energy, the sum of the potential and kinetic energy:

$$H(\theta, r) = U(\theta) + K(r) \quad (2.3)$$

According to the law of conservation of energy, the evolution of the state $(\theta, r)$ is governed by the *Hamilton equations* (2.4) as follows.

**Definition 2.2 (Hamiltonian Dynamics).** Assume that the Hamiltonian, $H(\theta, r)$, is differentiable, Hamiltonian dynamics is defined by the following Hamilton equations:

$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial r_i}$$
$$\frac{dr_i}{dt} = -\frac{\partial H}{\partial \theta_i} \quad (2.4)$$

for $i = 1, \ldots, d$

The Hamiltonian dynamics of (2.4) defines a flow, parameterized by a trajectory length $s \in \mathbb{R}$, which is a map $\phi^H_s : \mathbb{R}^{2d} \to \mathbb{R}^{2d}, \quad (\theta(t_0), r(t_0)) \mapsto (\theta(t_0 + s), r(t_0 + s)), \quad \forall t_0, s \in \mathbb{R}$.

Alternatively, we can rewrite the Hamilton equations (2.4) into a matrix form by combining the position vector $\theta$ and the momentum vector $r$ into the state vector $z := (\theta, r)$:

$$\frac{dz}{dt} = J\nabla_z H(z)$$
where
\[
J = \begin{bmatrix}
0_{d \times d} & I_{d \times d} \\
-I_{d \times d} & 0_{d \times d}
\end{bmatrix}
\]
is a symplectic matrix.

Hamiltonian dynamics has several fundamental properties that are of great importance in the construction of MCMC.

**Reversibility.** The Hamiltonian flow \( \phi^H_s \) defines a one-to-one mapping from the state at time \( t_0, z(t_0) \) to the state at time \( t_0 + s, z(t_0 + s) \), whose inverse \( \phi^{-H}_s \) can be obtained by simply negating the time derivatives in equations (2.4). When the kinetic energy is in the quadratic form, the inverse mapping can also be obtained by negating \( r \), applying \( \phi^H_s \), and then negating \( r \) again.

**Proposition 2.1 (Time Reversibility).** Hamiltonian dynamics (2.4) is time reversible.

**Proof.** Let \( z'(t) = (\theta(t_0 + s - t), -r(t_0 + s - t)) = I z(t_0 + s - t), 0 \leq t \leq s \) denote the inverse flow, where \( I = \begin{bmatrix}
I_{d \times d} & 0_{d \times d} \\
0_{d \times d} & -I_{d \times d}
\end{bmatrix} \). Then it satisfies the following differential equations

\[
\frac{dz'}{dt} = -I \frac{dz}{dt} = -I J \nabla z' H(z') = J \nabla z H(z')
\]

which is identical to Hamiltonian dynamics (2.4). Let \( \nu \) be the operator on the state space that keeps \( \theta \) and negates \( r \), then

\[
\phi^{-H}_s(z(t_0 + s)) = z(t_0) = \nu \circ z'(s) = \nu \circ \phi^H_s(z'(0)) = \nu \circ \phi^{-H}_s \circ \nu(z(t_0 + s))
\]

therefore, we have \( \phi^{-H}_s = \nu \circ \phi^H_s \circ \nu \).

**Remark.** Note that \( \nu \circ \phi^H_s \circ \nu \circ \phi^H_s = \phi^{-H}_s \circ \phi^H_s = I \). Denote \( \phi^H_s = \nu \circ \phi^H_s \), then \( \phi^H_s \circ \phi^H_s = I \Rightarrow \phi^{-H}_s = \phi^H_s \).
The reversibility of Hamiltonian dynamics can be used to prove the reversibility of the Markov chain transitions, which is important to show the stationarity of the resulting Markov chain.

**Volume Preservation.** The second fundamental property of Hamiltonian dynamics is that the Hamiltonian flow \( \phi^H_t \) preserves volume in the state space \( z = (\theta, r) \). It follows from a stronger property of Hamiltonian system: symplecticity.

**Definition 2.3 (Symplectic Transformation).** A differentiable map \( g : U \to \mathbb{R}^{2d} \) is called symplectic if the Jacobian matrix \( \frac{\partial g}{\partial z} \) is everywhere symplectic, i.e.,

\[
\left( \frac{\partial g}{\partial z} \right)^\dagger J^{-1} \left( \frac{\partial g}{\partial z} \right) = J^{-1}
\]

**Theorem 2.1 (Poincaré 1899).** Let the Hamiltonian \( H(\theta, r) \) be a twice continuously differentiable function on \( U \subset \mathbb{R}^{2d} \). Then, for each fixed \( s \), the flow \( \phi^H_s \) is a symplectic transformation wherever it is defined.

**Proof.** See Appendix A.1 for a detailed proof.

**Proposition 2.2 (Volume Preservation).** Hamiltonian dynamics (2.4) is volume preserving

**Proof.** It follows from Theorem 2.1 that the Jacobian matrix of \( \phi^H_s \) satisfies

\[
\left( \frac{\partial \phi^H_s}{\partial z_0} \right)^\dagger J^{-1} \left( \frac{\partial \phi^H_s}{\partial z_0} \right) = J^{-1}
\]

which implies \( \det(\partial \phi^H_s/\partial z_0) = 1 \), and therefore volume conservation.

This property simplifies the computation in acceptance probability for Metropolis updates since we don’t need to account for any change in volume.
Conservation of the Hamiltonian  The last property of the dynamics is that it preserves the total energy, the Hamiltonian. As a result, the new states generated from the Hamiltonian flow will always be accepted if (2.4) is analytically solved.

Proposition 2.3 (Conservation of the Hamiltonian). Hamiltonian dynamics (2.4) is energy conservative.

Proof. It follows from Hamilton equations (2.4)

\[
\frac{dH}{dt} = \frac{\partial H}{\partial \theta} \cdot \frac{d\theta}{dt} + \frac{\partial H}{\partial r} \cdot \frac{dr}{dt} = \frac{\partial H}{\partial \theta} \cdot \frac{\partial H}{\partial r} - \frac{\partial H}{\partial r} \cdot \frac{\partial H}{\partial \theta} = 0
\]

In practice, Hamiltonian dynamics (2.4) might not be analytically solvable and we need to resort to numerical integrators. However, high acceptance probability can still be maintained by controlling the discretization error carefully. Moreover, the reversibility and volume preservation properties can be maintained exactly for certain discretizations and the corresponding numerical methods are called geometric integrators (Leimkuhler and Reich, 2004; Girolami and Calderhead, 2011).

2.2.2 Leapfrog Scheme

One of the commonly used geometric integrators is the leapfrog scheme. Without loss of generality, we assume that the kinetic energy \( K(r) = \frac{1}{2} r^T M^{-1} r \). However, the scheme below can be applied with any form for the kinetic energy. Discretizing time with some
small step size $\varepsilon$, the *leapfrog* scheme works as follows:

$$
\begin{align*}
    r^{(t+1/2)} &= r^{(t)} - \frac{\varepsilon}{2} \nabla_\theta U(\theta^{(t)}) \\
    \theta^{(t+1)} &= \theta^{(t)} + \varepsilon M^{-1} r^{(t+1/2)} \\
    r^{(t+1)} &= r^{(t+1/2)} - \frac{\varepsilon}{2} \nabla_\theta U(\theta^{(t+1)})
\end{align*}
$$

(2.5)

It is also straightforward to check that the *leapfrog* scheme is time reversible and volume preserving. The time reversibility comes from the symmetry of the scheme. Starting at $z(t + \varepsilon) = (\theta(t + \varepsilon), r(t + \varepsilon))$, one can get back to $z(t) = (\theta(t), r(t))$ by simply negating $r$ and apply the scheme (2.5) and then negating $r$ again, which is similar to the continuous case. The scheme also preserves volume exactly, since all updates in (2.5) are shear transformations.

In HMC, the state of $\theta$ at the end of the trajectory after running the *leapfrog* scheme for $L$ steps with step size $\varepsilon$ is used as a trail move and a Metropolis correction step is taken to decide whether to accept it or not. The scheme (2.5) introduces local and global discretization errors. The local error, the error after one step that moves from time $t$ to time $t + \varepsilon$, is $O(\varepsilon^3)$ and the global error, the error after simulating for some fixed time interval $s$ (requires $s/\varepsilon$ steps), is $O(\varepsilon^2)$. Due to the introduced discretization error, the Hamiltonian $H$ can not keep invariant and the acceptance probability could be less than 1. For more detailed discussion on the numerical properties of the *leapfrog* scheme, see (Leimkuhler and Reich, 2004; Neal, 2011).

### 2.2.3 Model-based Hamiltonian Flow

In Bayesian statistics, posterior samples of the model parameters is usually the focus of interest for inference or prediction purpose. The model parameters take the role of the position variable, $\theta$. To define the dynamics, we augment the state space with a fictitious momentum variable $r$ which is of the same dimension as $\theta$. 

17
Assume the target posterior distribution has density $\pi(\theta|D)$, where $D$ is the observed data. Motivated by statistical mechanics, we define the corresponding potential energy function as the negative log of the density $\pi(\theta|D)$

$$U(\theta) = -\log(\pi(\theta|D)) = -[\log(p(\theta)) + \log(L(\theta|D))] \tag{2.6}$$

where $p(\theta)$ is the prior density, and $L(\theta|D)$ is the likelihood function.

For the fictitious momentum $r$, we assign a kinetic energy function

$$K(r) = \frac{1}{2}r^\top M^{-1}r \tag{2.7}$$

which implies $r$ follows a zero-mean multivariate Gaussian distribution

$$\pi(r) \propto \exp(-K(r)) \Rightarrow r \sim \mathcal{N}(0, M).$$

where $M$ is a mass matrix and is often set to identity, $I$. The fictitious Hamiltonian $H$, therefore, is defined as the total energy of the system and the joint density of $(\theta, r)$ is

$$\pi(\theta, r) \propto \exp(-U(\theta) - K(r)) \propto \pi(\theta|D) \cdot \pi(r) \tag{2.8}$$

Note that $\theta$ and $r$ are independent in (2.8) when $M$ is constant.

HMC samples from the joint distribution of $\theta$ and $r$ defined by (2.8) as follows: i) given the current position $\theta$, it samples a random momentum variable $r$ from the conditional distribution, the independent Gaussian distribution $\mathcal{N}(0, M)$; ii) Hamiltonian dynamics then is simulated for $L$ steps using the leapfrog scheme, with a step size $\varepsilon$, starting from the current state $z = (\theta, r)$. The new state at the end of the trajectory (with momentum variable being negated) is used as a proposal $z^* = (\theta^*, -r^*) = \varphi^H_s(z)$; iii) The proposed state $z^*$ is accepted
as the next state of the Markov chain with the following probability

\[
\alpha_{\text{hmc}}(z, z^*) = \min \left\{ 1, \frac{\pi(z^*) \delta_{\varphi^H_s(z^*)}(z)}{\pi(z) \delta_{\varphi^H_s(z)}(z)} \right\} = \min \{ 1, \exp(-H(z^*) + H(z)) \} \tag{2.9}
\]

where \( \delta \) is the Dirac delta function. i), ii) and iii) together make one iteration in HMC. Since \( \theta \) and \( r \) are independent, we can simply drop the momentum samples \( r \) and the position samples \( \theta \) follow the marginal distribution \( \pi(\theta | D) \).

Remark. The randomness of the proposing mechanism for HMC comes from drawing a random momentum variable \( r \) in step i). After that, the algorithm runs a deterministic ODE system (Hamiltonian dynamical system), which transfers some of the randomness from the momentum to the position, to generate proposals. The computation in (2.9) has been simplified since \( |dz^*|/|dz| = 1 \), which follows from volume preservation.

The following theorem ensures that HMC leaves the joint distribution (2.8) invariant.

**Theorem 2.2.** The Markov chain generated by the HMC algorithm has joint distribution (2.8) as its stationary distribution.

**Proof.** Let \( z' \) denote the next state, it suffices to verify the detail balance condition (2.1) for \( z' = z^* \).

\[
\pi(z) T(z'|z) = \pi(z) \delta_{\varphi^H_s(z)}(z') \alpha_{\text{hmc}}(z, z') |dz|
\]

\[
= \pi(z) \delta_{\varphi^H_s(z)}(z') |dz| \min \left\{ 1, \frac{\pi(z') \delta_{\varphi^H_s(z')}(z)}{\pi(z) \delta_{\varphi^H_s(z)}(z)} |dz'| \right\}
\]

\[
= \min \{ \pi(z) \delta_{\varphi^H_s(z)}(z') |dz|, \pi(z') \delta_{\varphi^H_s(z')}(z) |dz'| \}
\]

\[
= \pi(z') \delta_{\varphi^H_s(z')}(z) |dz'| \min \left\{ 1, \frac{\pi(z) \delta_{\varphi^H_s(z)}(z') |dz|}{\pi(z') \delta_{\varphi^H_s(z')}(z) |dz'|} \right\}
\]

\[
= \pi(z') \delta_{\varphi^H_s(z')}(z) |dz'| \alpha_{\text{hmc}}(z', z) |dz'| = \pi(z') T(z'|z')
\]

The last equality follows from the reversibility property, \( \varphi^H_{-s} = \varphi^H_s \).
Algorithm 2.2 Hamiltonian Monte Carlo

**Input:** Starting position $\theta^{(1)}$ and step size $\varepsilon$

for $t = 1, 2, \ldots, T$ do

- Resample momentum $r$
  \[ r^{(t)} \sim \mathcal{N}(0, M) \]
  \[ (\theta_0, r_0) = (\theta^{(t)}, r^{(t)}) \]

- Simulate discretization of Hamiltonian dynamics:
  for $l = 1$ to $L$ do
    \[ r_{l-1} \leftarrow r_{l-1} - \frac{\varepsilon}{2} \nabla_\theta U(\theta_{l-1}) \]
    \[ \theta_l \leftarrow \theta_{l-1} + \varepsilon M^{-1} r_{l-1} \]
    \[ r_l \leftarrow r_{l-1} - \frac{\varepsilon}{2} \nabla_\theta U(\theta_l) \]
  end for

  \[ (\theta^*, r^*) = (\theta_L, -r_L) \]

- Metropolis-Hasting correction:
  \[ u \sim \mathcal{U}_{[0,1]} \]
  \[ \rho = \exp[H(\theta^{(t)}, r^{(t)}) - H(\theta^*, r^*)] \]
  if $u < \min(1, \rho)$ then
    \[ \theta^{(t+1)} = \theta^* \]
  else
    \[ \theta^{(t+1)} = \theta^{(t)} \]
  end if
end for

The HMC algorithm is summarized in Algorithm 2.2. In recent years, many variants of HMC have been developed to make the algorithm more flexible and generally applicable in a variety of settings. For example, methods proposed in Hoffman and Gelman (2011); Wang et al. (2013) enable automatically tuning of the hyper-parameters such as the step size $\varepsilon$ and the number of *leapfrog* steps $L$, saving the amount of tuning-related headaches. Riemannian Manifold HMC (Girolami and Calderhead, 2011) further improves standard HMC’s efficiency by automatically adapting to local structures using Riemannian geometry of parameter space, as we detail in the next section.

### 2.2.4 Riemannian Manifold HMC

Although HMC explores the target distribution more efficiently than random walk Metropolis, it does not fully exploit the geometric structures of the underlying probabilistic model.
since a flat metric (i.e., \( M = I \)) is used. Using more geometrically motivated methods could substantially improve sampling algorithms’ efficiency. Recently, Girolami and Calderhead (2011) proposed a new method, called Riemannian Manifold HMC (RMHMC), that exploits the Riemannian geometry of the target distribution to improve standard HMC’s efficiency by automatically adapting to local structures. To this end, instead of the identity mass matrix commonly used in standard HMC, they use a position-specific mass matrix \( M = G(\theta) \).

More specifically, they set \( G(\theta) \) to the Fisher information matrix, and define Hamiltonian as follows:

\[
H(\theta, r) = U(\theta) + \frac{1}{2} \log \det G(\theta) + \frac{1}{2} r^T G(\theta)^{-1} r = \phi(\theta) + \frac{1}{2} r^T G(\theta)^{-1} r 
\]  

(2.10)

where \( \phi(\theta) := U(\theta) + \frac{1}{2} \log \det G(\theta) \). Note that standard HMC is a special case of RMHMC with \( G(\theta) = I \). Based on this dynamic, they propose the following HMC on Riemannian manifold:

\[
\frac{d\theta}{dt} = \nabla_r H(\theta, r) = G(\theta)^{-1} r \\
\frac{dr}{dt} = -\nabla_\theta H(\theta, r) = -\nabla_\theta \phi(\theta) + \frac{1}{2} \nu(\theta, r) 
\]  

(2.11)

where the \( i \)th element of the vector \( \nu(\theta, r) \) is

\[
(\nu(\theta, r))_i = -r^T \partial_i (G(\theta)^{-1}) r = (G(\theta)^{-1} r)^T \partial_i G(\theta) G(\theta)^{-1} r 
\]

with the shorthand notation \( \partial_i = \partial/\partial \theta_i \) for partial derivative.

The above dynamic is non-separable (it contains products of \( \theta \) and \( r \)), and the resulting proposal generating mechanism based on the standard leapfrog method is neither time-reversible nor symplectic. Therefore, the standard leapfrog algorithm cannot be used for the above dynamic (Girolami and Calderhead, 2011). Instead, we can use the Stömer-Verlet
\( r^{(t+1/2)} = r^{(t)} - \frac{\varepsilon}{2} \left[ \nabla_{\theta} \phi(\theta^{(t)}) - \frac{1}{2} \nu(\theta^{(t)}, r^{(t+1/2)}) \right] \) \hspace{1cm} (2.12)

\( \theta^{(t+1)} = \theta^{(t)} + \frac{\varepsilon}{2} \left[ G^{-1}(\theta^{(t)}) + G^{-1}(\theta^{(t+1)}) \right] r^{(t+1/2)} \) \hspace{1cm} (2.13)

\( r^{(t+1)} = r^{(t+1/2)} - \frac{\varepsilon}{2} \left[ \nabla_{\theta} \phi(\theta^{(t+1)}) - \frac{1}{2} \nu(\theta^{(t+1)}, r^{(t+1/2)}) \right] \) \hspace{1cm} (2.14)

The resulting map is 1) deterministic, 2) reversible, and 3) volume-preserving. However, it requires solving two computationally intensive implicit equations, (2.12) and (2.13), at each leapfrog step. As the dimension increases, the gain in the exploration efficiency brought by the local geometrical structure could be offset by the involved expensive computation and the overall efficiency of RMHMC might drop.

### 2.3 Scalable MCMC

The high exploration efficiency of HMC comes at a computational price. As modern data analysis usually involves computation with large scale data sets, the required computation of geometrical information in HMC and its variants, which typically need full scans of the entire data set,

\[ \nabla U(\theta) = -\sum_{x \in D} \nabla \log p(x|\theta) - \nabla \log p(\theta) \]

becomes a major bottleneck for more general applications. Therefore, devising scalable variants of the MCMC algorithms has become a hotspot in recent MCMC literature.

#### 2.3.1 Stochastic Gradient MCMC

In recent years, stochastic gradient MCMC algorithms have been proposed and proven quite useful in scaling their counterparts to large scale data sets (Welling and Teh, 2011; Ahn
et al., 2012; Chen et al., 2014; Ding et al., 2014; Ma et al., 2015). The key idea of these methods, which stems from stochastic gradient descent, is to subsample the data and use stochastic gradients in place of full-data gradients in the Hamiltonian dynamics simulations. Due to the noise introduced by subsampling, the convergence to the right target distribution for stochastic gradient MCMC methods often relies on stochastic differential equations of the following form

$$dz = f(z)dt + \sqrt{2D(z)} \, dW(t)$$

(2.15)

where $f(z)$ denotes the deterministic drift term, $D(z)$ is a positive semidefinite diffusion matrix and $W(t)$ is a multidimensional Wiener process. The goal, of course, is to choose appropriate $f(z)$ and $D(z)$ such that the stationary distribution of (2.15) is $\pi^*(z) \propto \exp(-H(z))$.

Ma et al. (2015) devises a complete recipe for constructing SDEs with the correct stationary distribution as follows

$$f(z) = -[D(z) + Q(z)]\nabla H(z) + \Gamma(z), \quad \Gamma_i(z) = \sum_j \frac{\partial}{\partial z_j} (D_{ij}(z) + Q_{ij}(z))$$

(2.16)

where $Q(z)$ is a skew-symmetric curl matrix representing the deterministic traversing effects. Matrices $D(z)$ and $Q(z)$ can be adjusted to improve convergence to the target posterior distribution.

**Theorem 2.3** (Ma et al. 2015). The SDE in (2.15) has a stationary distribution $\pi^*(z) \propto \exp(-H(z))$ if $f(z)$ is restricted to the form of (2.16), with $D(z)$ positive semidefinite and $Q(z)$ skew-symmetric. If $D(z)$ is positive definite, or if ergodicity can be proven, then the stationary distribution is unique. Conversely, any continuous Markov process with desired stationary distribution $\pi^*(z)$ corresponds to an SDE as in (2.15) with $f(z)$ defined in (2.16).

Proof. The evolution of the distribution $\pi(z, t)$ of (2.15) is characterized by the Fokker-
Planck equation

\[ \partial_t \pi(z, t) = -\nabla \cdot (f(z) \pi(z, t)) + \nabla \nabla^\top : (D(z) \pi(z, t)) \quad (2.17) \]

where \( : \) represents a matrix double dot product \( X:Y = \text{Tr}(X^\top Y) \). (2.17) can be further transformed into a more compact form (Yin and Ao, 2006; Shi et al., 2012)

\[ \partial_t \pi(z, t) = \nabla \cdot ([D(z) + Q(z)][\pi(z, t) \nabla H(z) + \nabla \pi(z, t)]) \quad (2.18) \]

It is easy to verify that \( \pi^*(z) \propto \exp(-H(z)) \) is invariant under (2.18) since

\[ \exp(-H(z)) \nabla H(z) + \nabla \exp(-H(z)) = 0 \]

. If the process is ergodic, this stationary distribution is unique. For the proof of the converse part, see (Ma et al., 2015)

In practice, numerical simulation requires an \( \varepsilon \)-discretization of the SDE, leading to a full-data update

\[ z^{(t+1)} = z^{(t)} - \varepsilon_t \left[ \left( D(z^{(t)}) + Q(z^{(t)}) \right) \nabla H(z^{(t)}) + \Gamma(z^{(t)}) \right] + \mathcal{N}(0, 2\varepsilon_t D(z^{(t)})) \quad (2.19) \]

The gradient of \( H(z) \) involves the gradient of \( U(\theta) \), which can be estimated unbiasedly from a minibatch \( \tilde{\mathcal{D}} \) sampled uniformly at random from \( \mathcal{D} \)

\[ \nabla \tilde{U}(\theta) = -\frac{|\mathcal{D}|}{|\tilde{\mathcal{D}}|} \sum_{x \in \tilde{\mathcal{D}}} \nabla \log p(x|\theta) - \nabla \log p(\theta), \quad \tilde{\mathcal{D}} \subset \mathcal{D} \]

Assumed that we have independent data, by the central limit theorem,

\[ \nabla \tilde{U}(\theta) \approx \nabla U(\theta) + \mathcal{N}(0, V(\theta)) \quad (2.20) \]
Here, $V$ is the covariance of the stochastic gradient noise, which depends on the current model parameter $\theta$ and minibatch sample size. Therefore, we obtain a noisy Hamiltonian gradient $\nabla \tilde{H}(z) = \nabla H(z) + \left[ N(0, V(\theta)), 0 \right]^T$, which if plugged in (2.19) in place of $\nabla H(z)$, would result in an additional noise term. Assume that we have an estimate $\hat{B}_t$ of the variance of this additional noise, satisfying $2D(z^{(t)}) - \varepsilon_t \hat{B}_t \succeq 0$ (note that with small $\varepsilon_t$, this is always true and the stochastic gradient noise is dominated by the injected noise). Accounting for the stochastic gradient noise in the update, we have

$$ z^{(t+1)} = z^{(t)} - \varepsilon_t \left[ (D(z^{(t)}) + Q(z^{(t)})) \nabla \tilde{H}(z^{(t)}) + \Gamma(z^{(t)}) \right] + N(0, \varepsilon_t (2D(z^{(t)}) - \varepsilon_t \hat{B}_t)) $$

(2.21)

Note that the noise introduced by subsampling is multiplied by $\varepsilon_t$, which implies that the dynamics in (2.21) reduces to those of (2.19) as $\varepsilon_t$ is annealed. In this case, (2.21) yields the correct stationary distribution. This SDE based approach saves the computation for a costly M-H correction. However, annealing the step size $\varepsilon_t$ hinders the exploration efficiency. In practice, people often use a finite, small step size, resulting in a biased sampler (Chen et al., 2014; Ding et al., 2014).

It turns out that some of the recently developed stochastic gradient MCMC methods can be cast within the framework (2.21)

**Stochastic Gradient Langevin Dynamics (SGLD)** Scalable MCMC algorithms starts from SGLD (Welling and Teh, 2011), which uses the following first order (no momentum) Langevin dynamics

$$ \theta^{(t+1)} = \theta^{(t)} - \varepsilon_t D \nabla \tilde{U}(\theta^{(t)}) + N(0, 2\varepsilon_t D) $$

(2.22)

Note that SGLD corresponds to taking $z = \theta$ with $H(\theta) = U(\theta), D(\theta) = D, Q(\theta) = 0$ and $\hat{B}_t = 0$. Using Laplacian approximation of the posterior offers a more sophisticated estimate for the introduced noise $\hat{B}_t$, which leads to a variant of SGLD, the stochastic gradient Fisher scoring algorithm (Ahn et al., 2012)
Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) The SGHMC algorithm is introduced in Chen et al. (2014) which uses the second order Langevin dynamics

\[
\begin{align*}
\theta^{(t+1)} &= \theta^{(t)} + \varepsilon_t M^{-1} r^{(t)} \\
r^{(t+1)} &= r^{(t)} - \varepsilon_t \nabla \tilde{U}(\theta^{(t)}) - \varepsilon_t C M^{-1} r^{(t)} + \mathcal{N}(0, \varepsilon_t (2C - \hat{B}_t)) \\
\end{align*}
\] (2.23)

This method fits into the framework (2.21) with

\[
z = (\theta, r), \quad H(\theta, r) = U(\theta) + \frac{1}{2} r^\top M^{-1} r, \quad D(\theta, r) = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}, \quad Q(\theta, r) = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix}
\]

and $\hat{B}_t$ is an estimate of $V(\theta_t)$. By adding the friction term $CM^{-1} r$, the dynamics keeps the posterior distribution invariant.

Stochastic Gradient Nosé-Hoover Thermostat (SGNHT) The SGNHT method (Ding et al., 2014) incorporates ideas from thermodynamics to adaptively adjust the friction term by introducing an additional thermostat variable $\xi$ and uses the following dynamics

\[
\begin{align*}
\theta^{(t+1)} &= \theta^{(t)} + \varepsilon_t r^{(t)} \\
r^{(t+1)} &= r^{(t)} - \varepsilon_t \nabla \tilde{U}(\theta^{(t)}) - \varepsilon_t C M^{-1} r^{(t)} + \mathcal{N}(0, \varepsilon_t (2a I - \varepsilon_t \hat{B}_t)) \\
\xi^{(t+1)} &= \xi^{(t)} + \varepsilon_t \left( \frac{1}{d} r^{(t)} \nabla \right) - 1\right) \\
\end{align*}
\] (2.24)

To cast (2.24) within the framework, we can set $z = (\theta, r, \xi)$, $H(z) = U(\theta) + \frac{1}{2} r^\top r + \frac{1}{2d} (\xi - a)^2$

\[
D(z) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a I & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad Q(z) = \begin{bmatrix} 0 & -I & 0 \\ I & 0 & r/d \\ 0 & -r^\top/d & 0 \end{bmatrix}
\]

Other than unifying the existing stochastic gradient MCMC methods, the framework (2.21) can also be used to develop new samplers. See Ma et al. (2015) for more details.
Although stochastic gradient MCMC methods can scale to large data sets, the noise introduced by subsampling could lead to non-ignorable loss of accuracy, which in turn exerts a substantially negative impact on exploration efficiency (Betancourt, 2015). Therefore, in practice, it remains a challenge to design scalable MCMC algorithms that can find a good balance between computational complexity and exploration efficiency.

2.3.2 Surrogate Method

As an alternative approach to stochastic gradient MCMC methods, surrogate transition methods (Neal, 1996; Liu, 2001) aim at finding computationally cheaper surrogate functions to substitute the expensive target potential energy functions. By exploiting smoothness or regularity in parameter space, which is true for most statistical models, surrogate method reduces the computation cost while maintaining comparable exploration efficiency, without requiring small or annealing step sizes.

Given a cheap surrogate function $U^*(\theta)$, one can construct efficient Markov transition kernel $T_s(\cdot|\cdot)$ which leaves $\pi^*(\theta) \propto \exp(-U^*(\theta))$ invariant. That is, the detailed balance condition

$$\pi^*(\theta)T_s(\theta'|\theta) = \pi^*(\theta'T_s(\theta'|\theta')$$

is satisfied. $T_s$, then, can be used to generate proposals for a Markov chain that has the true target distribution $\pi(\theta)$ as its stationary distribution (Liu, 2001).

**Proposition 2.4.** The target distribution $\pi(\theta)$ is the stationary distribution for a Markov chain simulated according to the following procedure: given the current state $\theta$, let $\vartheta_0 = \theta$ and recursively sample $\vartheta_i \sim T_s(\cdot|\vartheta_{i-1})$ for $i = 1, \ldots, k$. Then, accept the proposal $\theta^* = \vartheta_k$ with the following probability

$$\alpha_s(\theta, \theta^*) = \min\left\{ 1, \frac{\pi(\theta^*)\pi^*(\theta)}{\pi(\theta)\pi^*(\theta^*)} \right\}$$
Proof. As before, denote the next state for the Markov chain as $\theta'$. It suffices to show that the detailed balance condition is satisfied for $\theta' = \theta^*$. Note that the $k$-step transition function can be formally written as

$$T_s^{(k)}(\theta_k|\theta_0) = \int \cdots \int T_s(\theta_1|\theta_0) \cdots T_s(\theta_k|\theta_{k-1}) d\theta_1 \cdots \theta_{k-1}$$

Since $T_s$ leaves $\pi^*$ invariant, we have

$$\pi^*(\theta) T_s(\theta_1|\theta_0) \cdots T_s(\theta_k|\theta_{k-1}) = T_s(\theta_0|\theta_1) \pi^*(\theta_1) T_s(\theta_2|\theta_1) \cdots T_s(\theta_k|\theta_{k-1})$$

$$= \cdots \cdots \tag{2.25}$$

$$= \pi^*(\theta^*) T_s(\theta_{k-1}|\theta_k) \cdots T_s(\theta_0|\theta_1)$$

Integrating out $\theta_1, \ldots, \theta_{k-1}$ on both sides of (2.25), we have

$$\pi^*(\theta) T_s^{(k)}(\theta^*|\theta) = \pi^*(\theta^*) T_s^{(k)}(\theta^*|\theta^*)$$

Therefore,

$$\pi(\theta) T(\theta^*|\theta) = \pi(\theta) T_s^{(k)}(\theta^*|\theta) \alpha_s(\theta, \theta^*)$$

$$= \pi^*(\theta) T_s^{(k)}(\theta^*|\theta) \min \left\{ \frac{\pi(\theta)}{\pi^*(\theta)}, \frac{\pi(\theta^*)}{\pi^*(\theta^*)} \right\}$$

$$= \pi^*(\theta^*) T_s^{(k)}(\theta^*|\theta^*) \min \left\{ \frac{\pi(\theta)}{\pi^*(\theta)}, \frac{\pi(\theta^*)}{\pi^*(\theta^*)} \right\}$$

$$= \pi(\theta^*) T(\theta^*|\theta^*)$$

Proposition 2.4 provides a general framework of accelerating MCMC methods by using efficient surrogate transition kernels. Therefore, it can be adopted to scale MCMC methods to large data sets or complex models. The efficiency of surrogate method depends on how fast
the surrogate transition kernel $T_s$ is and how close to the target $\pi(\theta)$ the surrogate induced distribution $\pi^*(\theta)$ is, another trade-off between computational complexity and approximation accuracy. It could be particularly useful if the redundancy of large scale data sets or complex models can be properly exploited to construct efficient surrogate transition kernels with good approximation quality.

There are many ways to devise efficient transition kernels that can preserve $\pi^*(\theta)$. For example, one can use the Metropolis rule on $\pi^*(\theta)$. In this thesis, we take a more sophisticated design which leverages the Hamiltonian dynamics corresponding to $\pi^*(\theta)$. Similar ideas have been exploited in Rasmussen (2003); Lan et al. (2015); Strathmann et al. (2015) where surrogate functions are devised based on Gaussian processes and reproducing kernel Hilbert space (RKHS). However, the usefulness of these methods is often limited to moderate dimensional problems due to the expensive computational cost of acquiring such surrogates with desired approximation accuracy. In what follows, we develop a series of surrogate methods that can be potentially generalized to higher dimensional problems. In addition, we provide a variational perspective of the surrogate based MCMC methods and develop techniques that can combine the advantages of both variational Bayes and MCMC approaches.
Chapter 3

Precomputing Strategies

Markov chain Monte Carlo (MCMC) (Metropolis et al., 1953) algorithms play an important role in statistical inference problems with intractable probability distributions. Recently, many MCMC algorithms such as Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2011) and its variants leverage Hamiltonian dynamics to define a transition kernel that efficiently explores a target distribution. These algorithms, however, tend to be computationally intensive which would limit their usefulness, especially for big data problems due to repetitive evaluations of functions and statistical quantities that depend on the data. This issue occurs in many statistical computing problems. In this chapter, we propose a novel surrogate method that exploits smoothness (regularity) in parameter space to improve computational efficiency of MCMC algorithms. Our proposed method is based on precomputing the required geometric information on a set of grids before running sampling algorithm and approximating the geometric information for the current location of the sampler using the precomputed information at each iteration of HMC. We start with an illustrative example in section 3.1. Section 3.2 then presents a general framework that incorporates our proposed precomputing strategies into HMC algorithm. In section 3.3, two grid based precomputing methods are introduced to construct efficient surrogate functions, which work for problems
3.1 Insights From an Illustrative Example

We start with a simple example to motivate our approach. Consider a bivariate Gaussian distribution with known covariance matrix and a conjugate prior

\[ Y|\mu \sim \mathcal{N}(\mu, \Sigma), \quad \mu \sim \mathcal{N}(\mu_0, \Sigma_0) \]

Note that in this case, the posterior distribution has a closed form so that MCMC is not required. However, we use this example to motivate our method. For this problem, the potential energy function and its gradient are given by

\[ U(\mu) = \frac{1}{2} \sum_{i=1}^{N} (y_n - \mu)^T \Sigma^{-1} (y_n - \mu) + \frac{1}{2} (\mu - \mu_0)^T \Sigma_0^{-1} (\mu - \mu_0) \]  \hspace{1cm} (3.1)  

\[ \frac{\partial U}{\partial \mu} = N \Sigma^{-1} (\mu - \bar{Y}) + \Sigma_0^{-1} (\mu - \mu_0), \quad \bar{Y} = \frac{1}{N} \sum_{i=1}^{N} y_i/N \]  \hspace{1cm} (3.2)  

In the gradient function (4.13), all the information about the parameter is contained in one single value \( \bar{Y} \) (i.e., the sufficient statistic for \( \mu \)). Therefore, if we precompute \( \bar{Y} \), gradient computation of the potential energy function \( U \) could be reduced to a simple matrix vector multiplication. Moreover, the gradient function itself is a linear function. In this 2D case, the essential information of \( \frac{\partial U}{\partial \mu} \) can be captured by its function values at three non-collinear points (left panel of Fig.3.1). On the other hand, samples from the posterior distribution are concentrated around the high density region where the neighborhood of one sample is
Figure 3.1: 2D Gaussian example: (a) the graph of the first component of $\nabla_{\mu}U$. The function value at red point can be obtained by interpolation when the function values at three blue points are known. (b) HMC samples from the posterior distribution.

frequently visited in the simulations of Hamiltonian dynamics (right panel of Fig.3.1). We use these insights to develop a method that can approximate the gradient function using precomputed values in order to accelerate standard HMC.

### 3.2 A General Framework

If we could solve Hamilton’s equations (2.4) analytically, the acceptance probability of new proposals in HMC would be exactly one (i.e., each proposal is accepted) because of the conservation of the Hamiltonian. In practice, however, analytical solution is usually not available and people often resort to numerical integrators (e.g., the leapfrog method) to simulate the Hamiltonian flow. Due to the introduced discretization error, the acceptance probability may be less than one. The trade-off between the accuracy of the proposal-generating mechanisms and Metropolis acceptance probability can go beyond time discretization. Therefore, we ask the following question: *can we properly approximate the proposal-generating mechanism in order to reduce computational complexity while keeping the acceptance probability at a reasonable level?* We answer this question in the remaining part of this section.
Note that the kinetic energy takes a quadratic form in (2.7), we can rewrite Hamilton’s
equations as follows:

\[
\frac{d\theta_i}{dt} = [M^{-1}\mathbf{r}]_i \\
\frac{dr_i}{dt} = -\frac{\partial U}{\partial \theta_i}
\] (3.3)

The routine of the trajectory for one iteration is determined by both the random sampling
of the initial momentum and the negative gradient of the potential energy function, which
is called force in Physics,

\[ \mathbf{F} = -\nabla_\theta U(\theta) \]

The random momentum enables the scheme to explore the target distribution stochastically,
and the fictitious force guides the sampler in the right direction. Following the fictitious
force field, the diffusive behavior of random walk proposals is suppressed so that the entire
sampling method would be more efficient than simple MCMC methods using random walk
proposals. However, the computation of the true force \( \mathbf{F} \) could be quite expensive for
problems with large scale data sets or complex models. To alleviate this issue, we propose
to construct a Hamiltonian dynamical system, at this time for the proposal step only, using
an alternative Hamiltonian function,

\[
H^*(\theta, \mathbf{r}) = U^*(\theta) + K(\mathbf{r})
\] (3.4)

where \( U^* \) is an approximation to the true potential energy \( U \), whose negative gradient \( \mathbf{F}^* \)
(which is an approximation to the true force function \( \mathbf{F} \)) provides guidance similar to the
true force \( \mathbf{F} \) but can be computed much faster. As a result, simulating this alternative
Hamiltonian dynamics offers a more efficient transition kernel which allows us to develop
scalable HMC algorithms. This way, our method can be viewed as a surrogate method (Neal,
1996; Liu, 2001; Rasmussen, 2003) that leverages the Hamiltonian dynamics corresponding
to \( \pi^*(\theta) \propto \exp(-U^*(\theta)) \), with an intuitive physical explanation. Note that the dynamical
system induced by the alternative Hamiltonian \( H^* \) is also reversible and volume preserving,
the convergence to the correct target distribution therefore can be guaranteed if we use the original Hamiltonian when calculating the acceptance probability of the proposals.

**Proposition 3.1.** If we run a Markov chain according to the following procedure: i) given the current position $\theta$, samples a random momentum variable $r \sim N(0, M)$; ii) starting from the current state $z = (\theta, r)$, simulate the following Hamiltonian dynamics

\[
\frac{d\theta}{dt} = M^{-1}r, \quad \frac{dr}{dt} = -\nabla_\theta U^*(\theta)
\] (3.5)

for $L$ steps using the leapfrog scheme, with a step size $\varepsilon$. Use the state at the end of the trajectory (with momentum variable being negated) as a proposal $z^* = (\theta^*, -r^*) = \varphi^H_\varepsilon(z)$; iii) Accept the proposed $z^*$ as the next state of the Markov chain with the following probability

\[
\alpha^*_\text{hmc}(z, z^*) = \min \left\{ 1, \frac{\pi(z^*) \varphi^H_\varepsilon(z^*) |dz^*|}{\pi(z) \varphi^H_\varepsilon(z) |dz|} \right\} = \min \left\{ 1, \exp(-H(z^*) + H(z)) \right\}
\]

, the Markov chain has a stationary distribution $\pi(z) \propto \exp(-U(\theta)) \cdot \pi(r)$ whose marginal distribution of $\theta$ is the target distribution $\pi(\theta)$.

**Proof.** Note that (3.5) is a valid Hamiltonian dynamics. Therefore, it is also reversible and volume preserving and so is the corresponding leapfrog scheme. Similarly to Theorem 2.2, the detailed balance condition is satisfied (suffices to verify for $z' = z^*$)

\[
\pi(z)T(z'|z) = \pi(z)\varphi^H_\varepsilon(z')|dz'| = \pi(z)\varphi^H_\varepsilon(z)|dz| \min \left\{ 1, \frac{\pi(z') \varphi^H_\varepsilon(z') |dz'|}{\pi(z) \varphi^H_\varepsilon(z) |dz|} \right\} = \min \{ \pi(z') \varphi^H_\varepsilon(z') |dz'|, \pi(z) \varphi^H_\varepsilon(z) |dz| \} = \pi(z')T(z'|z')
\]

The last equality follows from the reversibility property, $\varphi^H_{-\varepsilon} = \varphi^H_\varepsilon$. \hfill \Box

**Remark.** The acceptance probability $\alpha^*_\text{hmc}$ in Proposition 3.1 is a little bit different from
the acceptance probability $\alpha_s$ in Proposition 2.4 for surrogate methods. That is because we are using the Hamiltonian dynamics (3.5) to generate proposals only, instead of providing a full transition kernel that keeps $\pi^*(\theta) \propto \exp(-U^*(\theta))$ invariant. In addition, Hamiltonian dynamics (3.5) preserves the Hamiltonian $H^*$ defined in (3.4), rather than the true Hamiltonian $H$. Therefore, the approximation quality of $U^*$ has to be controlled to maintain high acceptance probability.

3.3 Grid Based Surrogate Methods

Note that Hamiltonian dynamics (3.5) only involves the approximate force field, it suffices to find an approximate force function, $F^* = -\nabla_\theta U^*(\theta)$, directly. In this section, we introduce two grid based approximation methods that work for problems of different dimensionalities. They all adopt the precomputing ideas discussed in section 3.1.

3.3.1 Naive Grid HMC

To approximate the force function, one could simply use a piecewise constant function, which corresponds to a piecewise linear approximation of the potential energy. In most cases, the high density region of the posterior distribution can be covered by a finite domain $D$, henceforth called “domain of interest”. If we partition $D$ with a fine grid, justification of an appropriate piecewise constant approximation to the force function $F$ is guaranteed by the smooth dependence of $F$ (or $U$) in parameter space. For a 2-dimensional problem, suppose our domain of interest is $D = [a, b] \times [c, d]$. Given the grid points

$$x_i = a + i\Delta x, \quad y_j = c + j\Delta y, \quad i, j = 0, 1, \ldots, N_p$$
where $\Delta x$, $\Delta y$ are the corresponding grid sizes, for each cell, $C_{i,j} = [x_{i-1}, x_i] \times [y_{j-1}, y_j]$, we approximate the force function by its value at the center of the grid, $c_{i,j} = (x_{i-1/2}, y_{j-1/2})^T$:

$$F^s(\theta) = F_{i,j} \triangleq F(c_{i,j}), \text{ if } \theta \in C_{i,j}$$

By the smoothness of $F$, $\|F^s - F\|_\infty \to 0$ as $\Delta x, \Delta y \to 0$. Therefore, we can always find some fine grid to achieve the desired approximation accuracy.

Figure 3.2 shows a piecewise constant approximation to the force function of a logistic regression model with design matrix $X = (1, X_1)$ and true parameter $\beta_T = (-1, 1)^T$, where $X_1$ follows standard normal distribution. The binary responses $Y = (y_1, y_2, \ldots, y_N)^T$ are
sampled independently from Bernoulli distributions

\[ y_i \sim \text{Bernoulli}(p_i), \quad p_i = \frac{\exp(x_i^\top \beta)}{1 + \exp(x_i^\top \beta)} \]

Therefore, the likelihood function is

\[ L(\beta | X, Y) \propto \prod_{i=1}^{N} p_i^{y_i} (1 - p_i)^{1-y_i} \]

and the potential energy function and the force function are

\[
U(\beta) = -\sum_{i=1}^{N} \left[ y_i x_i^\top \beta - \log(1 + \exp(x_i^\top \beta)) \right] \tag{3.6}
\]

\[
F(\beta) = -\nabla_\beta U(\beta) = X^\top (Y - P) \tag{3.7}
\]

where \( P = (p_1, p_2, \ldots, p_N)^\top \). It can be seen from the graph that: (i) the approximate force “map” does point to the right direction so that it provides valid geometric information for HMC; (ii) the approximate force function also changes smoothly, which means that the numerical stability of the leap-frog scheme can be maintained with approximately the same step size as standard HMC. As a result, the proposed scheme with piecewise constant force functions would be consistent and stable. Therefore, we can precompute the piecewise constant function \( F^* \) in advance. When evaluating the force function in the simulation of the Hamiltonian dynamics system, we locate the cell \((i, j)\) for the current parameter \( q \) and read the approximate function value \( F^*(\theta) \) from the precomputed force map. We summarize this approach in Algorithm 3.1 and refer to it as Grid HMC (GHMC).

Our initial results showed that the Naive Grid HMC method would work well for low dimensional problems. However, its extension to high dimensional problems could be problematic because as the number of parameters increases, the number of grid nodes at which we need to evaluate the approximate force map grows exponentially. In other words, the method will
Algorithm 3.1 Naive Grid HMC

**Input:** Starting position $\theta^{(1)}$ and step size $\varepsilon$
Precompute the approximate force map $F^* (\theta) : F^*_{i,j} = - (\nabla_\theta U(c_{i,j}))$

for $t = 1, 2, \cdots, T$ do

Resample momentum $r$
$r^{(t)} \sim N(0, M)$, $(\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$

Simulate discretization of Hamiltonian dynamics:

for $l = 1$ to $L$ do

$r_{l-1} \leftarrow r_{l-1} + \frac{\varepsilon}{2} F^*(\theta_{l-1})$
$\theta_{l} \leftarrow \theta_{l-1} + \varepsilon M^{-1} r_{l-1}$
$r_{l} \leftarrow r_{l} + \frac{\varepsilon}{2} F^*(\theta_{l})$
end for

$(\theta^*, r^*) = (\theta_L, -r_L)$

Metropolis-Hasting correction:

$u \sim U[0,1]$
$\rho = \exp (H(\theta^{(t)}, r^{(t)}) - H(\theta^*, r^*))$

if $u < \min(1, \rho)$ then

$\theta^{(t+1)} = \theta^*$
else

$\theta^{(t+1)} = \theta^{(t)}$
end if
end for

encounter the curse of dimensionality.

### 3.3.2 Sparse Grid HMC

Instead of employing a full grid discretization, the sparse grid interpolation method uses a special discretization technique to approximate a smooth function over a sparse grid of points (Bungartz and Griebel, 2004; Klimke and Wohlmuth, 2005; Barthelmann et al., 2000). More specifically, it uses a hierarchical basis (a representation of a discrete function space that is equivalent to the conventional nodal basis) and a sparse tensor product construction. Discretization on sparse grids employs $\mathcal{O}(N \cdot \log(N)^{d-1})$ grid points only, where $d$ denotes the dimension and $N$ denotes the number of grid points at the boundary in each coordinate direction (i.e., the mesh size is $h = 1/N$). Using piecewise linear basis functions, the interpolation accuracy could be $\mathcal{O}(N^{-2} \cdot (\log N)^{d-1})$ with respect to the $L_2$ norm and $L_\infty$ norm,
under certain smoothness conditions. This is in contrast to a full grid methods, which need \( O(N^d) \) grid points to achieve an approximation accuracy of \( O(N^{-2}) \). Therefore, the curse of dimensionality of full grid methods is overcome to some extent.

### Smolyak’s formula

Assume that we want to approximate the smooth functions \( f : [0, 1]^d \rightarrow \mathbb{R} \) using a finite number of function values (at support nodes). For the one dimensional case, the interpolation formula is given by

\[
U^i(f) = \sum_{j=1}^{m_i} f(x^i_j) \cdot a^i_j
\]

where \( i \in \mathbb{N} \) is the level, \( X^i = \{ x^i_j \in [0, 1] | j = 1, \ldots, m_i \} \) are the support nodes in level \( i \), and \( a^i_j \in C([0, 1]) \) are the basis functions in level \( i \). We could use the following tensor product for multidimensional cases:

\[
(U^{i_1} \otimes \cdots \otimes U^{i_d})(f) = \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d}) \cdot (a^{i_1}_{j_1} \otimes \cdots \otimes a^{i_d}_{j_d})
\] (3.8)

However, the above product formula requires a large number \( (m_{i_1} \cdots m_{i_d}) \) of support nodes, which are sampled on the full grid.

In contrast, Smolyak’s formulas (Smolyak, 1963) uses a sparse tensor product construction. With \( U^0 = 0 \), define

\[
\Delta^i = U^i - U^{i-1}, \quad \forall \ i \in \mathbb{N}
\]

Moreover, we put \( |\mathbf{i}| = i_1 + \cdots + i_d \) for the level vector \( \mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}^d \). Then Smolyak’s algorithm is given by

\[
A_{q,d}(f) = \sum_{|\mathbf{i}| \leq q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f) = A_{q-1,d}(f) + \sum_{|\mathbf{i}| = q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f)
\]

(3.9)
for integers $q \geq d$, where $A_{d-1,d} = 0$. As shown in the following sections, Smolyak’s algorithm needs much smaller number of grid points, resulting in efficient “sparse grid” interpolation schemes.

**Sparse Grid and Multivariate Hierarchical Structure**

There are many possibilities to construct nested sparse grids. For example, the Clenshaw-Curtis type sparse grid $H^{CC}$ is constructed from the following formulas. Here, the $x^i_j$ are defined as

$$
x^i_j = \begin{cases} 
(j - 1)/(m_i - 1), & j = 1, \ldots, m_i, m_i > 1 \\
0.5, & j = 1, m_i = 1
\end{cases}
$$

In order to obtain nested sets of points, the number of nodes is given by

$$m_1 = 1 \quad \text{and} \quad m_i = 2^{i-1} + 1 \text{ for } i > 1$$

Figure 3.3 shows the Clenshaw-Curtis type sparse grids $H^{CC}$ in two and three dimensional spaces. Compared to full grids, the number of sparse grid points grows much slower with increasing dimension $d$. More specifically, we have the following lemma

**Lemma 3.1.** Let $W_i = \Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d}$ denote the hierarchical increments in (3.9) and $H_{q,d}$ denote the set of employed grid points for $A_{q,d}$. The number of inner grid points in $H_{n+d-1,d}$
is given by

$$|\hat{H}_{n+d-1,d}| = \sum_{i=0}^{n-1} 2^i \cdot \binom{d-1+i}{d-1}$$

$$= (-1)^d + 2n \cdot \sum_{i=0}^{d-1} \binom{n+d-1}{i} \cdot (-2)^{d-1-i}$$

$$= 2^n \cdot \left( \frac{n^{d-1}}{(d-1)!} + O(n^{d-2}) \right)$$

Thus, we have

$$|\hat{H}_{n+d-1,d}| = O(N \cdot \log(N)^{d-1}) \quad (3.10)$$

where $N = 2^n$ is the number of points in each individual coordinate direction.

Proof. See a detailed proof in Appendix B.1

Depending on the choice of sparse grids, different piecewise linear basis functions $a$ can be used for the univariate interpolation formulas $U^i(f)$. For the Clenshaw-Curtis grid, we have

$$a_1^i(x) = 1, \quad a_j^i(x) = \begin{cases} 1 - (m_i - 1) \cdot |x - x_j^i|, & |x - x_j^i| < 1/(m_i - 1), \\ 0, & \text{otherwise} \end{cases} \quad (3.11)$$

for $i > 1$ and $j = 1, \ldots, m_i$. Nodal basis functions can not be used in Smolyak’s formula (3.9) directly since they do not accommodate hierarchical structures. However, with the selection of nested sets of points, we can easily transform the univariate nodal basis into the hierarchical one. By definition, we have

$$\Delta^i(f) = U^i(f) - U^{i-1}(f)$$

$$= \sum_{j=1}^{m_i} f(x_j^i) \cdot a_j^i - \sum_{j=1}^{m_i} U^{i-1}(f)(x_j^i) \cdot a_j^i$$

$$= \sum_{j=1}^{m_i} (f(x_j^i) - U^{i-1}(f)(x_j^i)) \cdot a_j^i$$
Figure 3.4: Piecewise linear Nodal basis (a) and hierarchical functions (b) with support nodes $x^i_j \in X^i_\Delta, i = 1, 2, 3$ for the Clenshaw-Curtis grid

Figure 3.5: Piecewise linear interpolation: Nodal versus Hierarchical

since $f(x^i_j) - U^{i-1}(f)(x^i_j) = 0, \forall x^i_j \in X^{i-1}$,

$$\Delta^i(f) = \sum_{x^i_j \in X^i_\Delta} \left( f(x^i_j) - U^{i-1}(f)(x^i_j) \right) \cdot a^i_j$$ \hspace{1cm} (3.12)

From (3.12) we note that for all $\Delta^i(f)$, only the basis functions belonging to the grid points that have not yet occurred in a previous set $X^k, 1 \leq k \leq i - 1$ are involved. Note that the hierarchical basis functions span the the same finite-dimensional function space as the standard nodal point basis functions. See Figure 3.4 for a comparison of the nodal and the hierarchical basis functions. Figure 3.5 shows the construction of the interpolation formula using nodal basis functions and function values versus using hierarchical basis functions and hierarchical surpluses for a univariate function $f$. Figures 3.4 and 3.5 are reproductions based on Klimke and Wohlmuth (2005).
Applying the tensor product formula (3.8) with $\Delta^i$ given in (3.12), the hierarchical update in the Smolyak algorithm (3.9) now can be rewritten as

$$
\Delta A_{q,d}(f) = \sum_{|i|=q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f)
$$

$$
= \sum_{|i|=q} \sum_{x_{i_1}^{j_1} \in X_{\Delta}^{i_1}} \cdots \sum_{x_{i_d}^{j_d} \in X_{\Delta}^{i_d}} (f(x_{i_1}^{j_1}, \ldots, x_{i_d}^{j_d}) - A_{q-1,d}(f)(x_{i_1}^{j_1}, \ldots, x_{i_d}^{j_d})) \cdot (a_{i_1}^{j_1} \otimes \cdots \otimes a_{i_d}^{j_d})
$$

(3.13)

The coefficients in (3.13), also called hierarchical surpluses

$$
\omega_{n,i} \overset{\Delta}{=} f(x_i^n) - A_{n+d-1,d}(f)(x_i^n)
$$

were introduced by Bungartz (1998) and can be used to obtain an estimate of the current approximation error and terminate the algorithm automatically when a desired accuracy is reached.

**Accuracy of Piecewise Multilinear Interpolation**

Using piecewise linear basis functions, sparse grid interpolation based on Smolyak’s algorithm provides comparable approximation accuracy (up to logarithmic factors) to full grid methods, if the target function $f$ has bounded second mixed derivatives. That is, $f \in H^2_{\text{mix}}$ with

$$
H^2_{\text{mix}} := \{ f : [0, 1]^d \to \mathbb{R}, D^\alpha f \in L_2([0, 1]^d), |\alpha|_{\infty} \leq 2 \}
$$

where $\alpha \in \mathbb{N}_0^d, |\alpha|_{\infty} = \max\{\alpha_1, \ldots, \alpha_d\}$ and

$$
D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}, \quad |\alpha| = \sum_{i=1}^d \alpha_i
$$
Algorithm 3.2 Sparse Grid HMC

**Input:** Starting position \( \theta^{(1)} \) and step size \( \varepsilon \)

Precompute the hierarchical surpluses for Smolyak’s formula \( A_{n+d−1,d} \) of the true potential energy \( U \)

for \( t = 1, 2, \ldots, T \) do

Resample momentum \( r \)

\( r^{(t)} \sim \mathcal{N}(0, M), (\theta_0, r_0) = (\theta^{(t)}, r^{(t)}) \)

Simulate discretization of Hamiltonian dynamics

for \( l = 1 \) to \( L \) do

\( r_{l-1} \leftarrow r_{l-1} - \frac{\varepsilon}{2} \nabla A_{n+d−1,d}(U)(\theta_{l-1}) \)

\( \theta_l \leftarrow \theta_{l-1} + \varepsilon M^{-1} r_{l-1} \)

\( r_l \leftarrow r_l - \frac{\varepsilon}{2} \nabla A_{n+d−1,d}(U)(\theta_l) \)

end for

\((\theta^*, r^*) = (\theta_L, -r_L)\)

Metropolis-Hasting correction:

\( u \sim \mathcal{U}[0,1] \)

\( \rho = \exp(H(\theta^{(t)}, r^{(t)}) - H(\theta^*, r^*)) \)

if \( u < \min(1, \rho) \) then

\( \theta^{(t+1)} = \theta^* \)

else

\( \theta^{(t+1)} = \theta^{(t)} \)

end if

end for

For ease of presentation, we assume \( f \) is zero on the boundary, that is, \( f \in H^{2}_{0, \text{mix}} \). According to Bungartz (1998); Barthelmann et al. (2000); Bungartz and Griebel (2004), we have the following theorem

**Theorem 3.1.** For a function \( f \in H^{2}_{0, \text{mix}} \), the orders of the interpolation error for the sparse grid methods hold

\[
\| f - A_{n+d−1,d}(f) \|_2 = \mathcal{O}(N^{-2} \cdot \log(N)^{d−1}) \quad (3.14)
\]

\[
\| f - A_{n+d−1,d}(f) \|_\infty = \mathcal{O}(N^{-2} \cdot \log(N)^{d−1}) \quad (3.15)
\]

Together with Lemma 3.1, we see that sparse grids need much less points in higher dimensions than naive full grids to achieve a similar approximation quality.
Using sparse grid interpolation (3.9) based on Smolyak’s algorithm, we can generalize our Naive Grid HMC method to relatively higher dimensional problems. The hierarchical surpluses for the energy function $U$ can be precomputed with certain type of sparse grid and $\nabla A_{n+d-1,d}(U)$ can be called to replace the gradient computation (see Algorithm 3.2).

### 3.4 Domain of Interest

The grid needs to be specified over a finite domain of interest such that there is a good balance between the cost and efficiency of the precomputing strategy. That is, we need to find an appropriate bounded domain $D$ that covers most of the high density areas without creating cells that are rarely visited by the sampler.

Note that for points outside of $D$, one can still use the standard HMC method; that is, there will not be any computational saving for these points. More specifically, the overall potential energy function can be presented as follows:

$$
U^*(\theta) = U(\theta)(1 - 1_D(\theta)) + \hat{U}(\theta)1_D(\theta)
$$

(3.16)
where $U$ is the original potential, $\hat{U}$ is the precomputed approximation over $D$, and $\mathbb{1}$ is the indicator function. Therefore, within the identified domain the energy function is approximated; whereas, the energy function remains exact outside of the domain. Note that $U^*$ might not be continuous on the boundary of $D$. However, the discretized Hamiltonian dynamics is still time reversible and volume preserving (each of the equations are shear transformations)

$$\begin{align*}
\boldsymbol{r}^{(t+1/2)} &= \boldsymbol{r}^{(t)} - \frac{\varepsilon}{2} \nabla_{\theta} U^*(\theta^{(t)}) \\
\theta^{(t+1)} &= \theta^{(t)} + \varepsilon M^{-1} \boldsymbol{r}^{(t+1/2)} \\
\boldsymbol{r}^{(t+1)} &= \boldsymbol{r}^{(t+1/2)} - \frac{\varepsilon}{2} \nabla_{\theta} U^*(\theta^{(t+1)})
\end{align*}$$

Therefore, simulating the above induced dynamics system and using the original Hamiltonian in the computation of acceptance probability guarantees the convergence to the correct target distribution (see proposition 3.1). Finally, it is easy to show that the chain remains ergodic and can move between the two domains. To see this, notice that within the first step of the leapfrog, $\theta$ is updated as follows:

$$\begin{align*}
\theta_1 \sim N(\theta - \frac{\varepsilon^2}{2} \nabla_{\theta} U^*(\theta), \varepsilon^2 M)
\end{align*}$$

where the support is $\mathbb{R}^d$; therefore, the sampler has non-zero probability to move inside and outside of the domain.

To find the domain of interest, we use Laplace’s approximation,

$$\theta | Y \sim N(\hat{\theta}, J^{-1}(\hat{\theta}))$$

where $\hat{\theta}$ is the posterior mode which can be estimated using fast optimization methods, and $J(\hat{\theta}) = H_U(\hat{\theta})$ is the Hessian matrix at the point. Given a pre-specified probability, $p$, we
can find a domain with probability \( p \) based on the above normal approximation.

Figure 3.6 shows the domain of interest \( R \) for a logistic regression model. It can be seen that for different data sizes (\( N = 100 \) and \( 1000 \)), the corresponding domains of interest are adjusted automatically to capture the high density regions of the posterior distribution.

When the high density region is irregular and can not be represented well by a rectangular box, this might not be an efficient approach. Later, we will discuss a more general approach for such cases.

### 3.5 Experiments

In this section, we compare our proposed methods to standard HMC using several experiments in terms of sampling efficiency. We define sampling efficiency as time-normalized effective sample size (ESS). Given \( B \) MCMC samples for each parameter, we calculate the corresponding ESS = \( B[1 + 2\Sigma_{k=1}^{K}\gamma(k)]^{-1} \), where \( \Sigma_{k=1}^{K}\gamma(k) \) is the sum of \( K \) monotone sample autocorrelations (Geyer, 1992). We use the minimum ESS over all parameters normalized by
the CPU time, $s$ (in seconds), as the overall measure of efficiency: $\min(\text{ESS})/s$. The sparse grid interpolation is implemented using Matlab package \texttt{spinterp} (Klimke and Wohlmuth, 2005).

Empirical results show that both GHMC and Sparse Grid HMC (sgHMC) provide substantial improvement over standard HMC in terms of efficiency while maintaining relatively high acceptance rates.

### 3.5.1 Logistic Regression

For our first example, we sample $N = 100$ data points from a logistic regression model discussed in section 3.3.1 and choose the domain of interest to be $[-3, 0.5] \times [-0.5, 3]$ and set the grid size to 0.1. Figure 3.7 shows posterior samples using standard HMC and GHMC. Note that they both converge to the target distribution and explore the parameter space quite well. Table 3.1 compares the performance of these algorithms based on 3200 MCMC iterations after burning the first 800 iterations. As we can see, GHMC outperforms standard HMC in terms of time-normalized ESS.

Table 3.1: Comparing HMC with GHMC using a logistic regression model. For each method, we provide the acceptance rate (AR), the CPU time ($s$) for each iteration and the time-normalized ESS

<table>
<thead>
<tr>
<th>Method</th>
<th>AR</th>
<th>$\text{ESS}(\beta_0, \beta_1)$</th>
<th>$s$/Iteration</th>
<th>min ESS/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>0.9225</td>
<td>(3200, 3200)</td>
<td>7.0157E-4</td>
<td>1425.3707</td>
</tr>
<tr>
<td>GHMC</td>
<td>0.7981</td>
<td>(3200, 3200)</td>
<td>3.318E-4</td>
<td>3013.9031</td>
</tr>
</tbody>
</table>

### 3.5.2 Banana-shaped Distribution

The potential energy function for the logistic regression model is quite similar to a Gaussian distribution model, where the resulting force function is relatively smooth. To investigate
GHMC’s ability to explore the parameter space with a more complicated geometry, we construct a banana-shaped posterior distribution of $\beta = (\beta_1, \beta_2|y)$ based on the following model:

$$
y|\beta \sim \mathcal{N}(\beta_1 + \beta_2^2, \sigma_y^2)
$$

$$
\beta \sim \mathcal{N}(0, \sigma_\beta^2)
$$

The data $\{y_i\}_{i=1}^{100}$ are generated with $\beta_1 + \beta_2^2 = 1$, $\sigma_y = 2$, $\sigma_\beta = 1$. The potential energy function is

$$
U(\beta) = \sum_{i=1}^{N}(y_i - \beta_1 - \beta_2^2)^2 + \frac{\beta_1^2 + \beta_2^2}{2\sigma_\beta^2}
$$

and the force function is

$$
F(\beta) = -\nabla_\beta U(\beta) = \sum_{i=1}^{N}(y_i - \beta_1 - \beta_2^2) \cdot \left(1 \over \sigma_y^2 \beta_2^2 \over \sigma_\beta^2 \right)
$$

Here, we choose the domain of interest to be $[-4, 4] \times [-4, 4]$ and set grid size to 0.1. Figure 3.8 shows the samples for the posterior distribution using standard HMC and GHMC. As before,
both methods converge to the target distribution and explore the parameter space quite well.
Even though Banana-shaped distribution is more distorted and the force function is more
complex, $\tilde{F}$ (grid size 0.1) still provide a good approximation to the true force function.
Table 3.2 compares the performance of these algorithms based on 3200 MCMC iterations
after burning the first 800 iterations. As before, GHMC outperforms standard HMC in terms
of time-normalized ESS.

Table 3.2: Comparing HMC with GHMC using a banana-shaped distribution model. For
each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and
the time-normalized ESS

<table>
<thead>
<tr>
<th>Method</th>
<th>AR</th>
<th>ESS($\beta_1$, $\beta_2$)</th>
<th>s/Iteration</th>
<th>min ESS/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>0.9353</td>
<td>(2403, 1191.6)</td>
<td>3.8703E-4</td>
<td>962.1346</td>
</tr>
<tr>
<td>GHMC</td>
<td>0.6587</td>
<td>(893.8862, 766.2423)</td>
<td>1.4498E-4</td>
<td>1651.5917</td>
</tr>
</tbody>
</table>

3.5.3 Gaussian Process Model

For our third example, we use a Gaussian process model. Posterior sampling for these mod-
els tends to be quite difficult due to the computation cost associated with inverting the
covariance matrix. See Neal (1998); Rasmussen (1996) for more details on Gaussian pro-
cesses. Here we construct a 2D Gaussian process with zero mean and the squared exponential
covariance function,

$$Y \sim \mathcal{N}(0, \Sigma), \quad \Sigma_{ij} = \eta \cdot \exp\left(-l \|x_i - x_j\|^2\right) + J \cdot \delta_{ij}$$

where $\eta, l, J$ are positive hyperparameters with log-normal priors.

$$\log(\eta) \sim \mathcal{N}(-1, 1), \quad \log(l) \sim \mathcal{N}(-1, 1), \quad \log(J) \sim \mathcal{N}(-1, 1)$$
Figure 3.9: HMC vs sgHMC: Gaussian Process

Table 3.3: Comparing HMC with sgHMC using a Gaussian process model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

<table>
<thead>
<tr>
<th>Method</th>
<th>AR</th>
<th>ESS(\eta, l, J)</th>
<th>s/Iteration</th>
<th>min ESS/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>0.9472</td>
<td>(1021.7, 1784.8, 3200)</td>
<td>2.3547E-1</td>
<td>1.3559</td>
</tr>
<tr>
<td>sgHMC</td>
<td>0.7066</td>
<td>(828.7, 1380.0, 3200)</td>
<td>2.9851E-2</td>
<td>8.6752</td>
</tr>
</tbody>
</table>

Let \( \eta = \log(\eta), \ l = \log(l), \ J = \log(J) \), the potential energy function is

\[
U(\tilde{\eta}, \tilde{l}, \tilde{J}) = \frac{1}{2} \log(|\Sigma|) + \frac{1}{2} Y^T \Sigma^{-1} Y + \frac{1}{2} \left[ (\tilde{\eta} + 1)^2 + (\tilde{l} + 1)^2 + (\tilde{J} + 1)^2 \right]
\]

and the force function is

\[
F(\beta) = -\frac{\partial U}{\partial \beta} = \frac{1}{2} \text{tr} \left( \Sigma^{-1} \frac{\partial \Sigma}{\partial \beta} \right) - \frac{1}{2} Y^T \Sigma^{-1} \frac{\partial \Sigma}{\partial \beta} \Sigma^{-1} Y + \beta + 1, \quad \beta = (\tilde{\eta}, \tilde{l}, \tilde{J})^T
\]

The domain of interest for \( \beta = (\tilde{\eta}, \tilde{l}, \tilde{J})^T \) is set to be \([-1.6, 1.6] \times [-1.6, 1.6] \times [-1.2, 0.4] \) where we train a sparse grid interpolator to replace the force function. Figure 3.9 shows
the samples from the posterior distribution given by standard HMC and sgHMC. Table 3.3 compares the performance of the two algorithms based on 3200 MCMC iterations after 800 burn-in iterations. As we can see, sgHMC substantially outperforms standard HMC.

### 3.5.4 Elliptic PDE Inverse Problem

Our last example is a canonical inverse problem involving inference of the diffusion coefficient in an elliptic PDE (Dashti and Stuart, 2011; Conard et al., 2014). The forward model is to solve a two dimensional elliptic PDE

\[ \nabla_x \cdot (c(x, \theta) \nabla_x u(x, \theta)) = 0 \quad (3.19) \]

where \( x = (x_1, x_2) \in [0, 1]^2 \) is the spatial coordinate. The boundary conditions are

\[ u(x, \theta)|_{x_2=0} = x_1, \quad u(x, \theta)|_{x_2=1} = 1 - x_1 \]

\[ \frac{\partial u(x, \theta)}{\partial x_1} \bigg|_{x_1=0} = 0, \quad \frac{\partial u(x, \theta)}{\partial x_1} \bigg|_{x_1=1} = 0 \]

This PDE provides a simple model of steady-state flow in porous media. The coefficient \( c \) represents the permeability of a porous medium while \( u \) represents the pressure head. In this inverse problem, the objective of interest is to infer the unknown diffusion coefficient conditioned on observation data where Bayesian approach can be naturally adopted. In our numerical simulation, (4.14) is solved using standard continuous GFEM with bilinear basis functions on a uniform \( 30 \times 30 \) quadrilateral mesh.

A log-Gaussian process prior is given to the diffusivity field \( c(x) \) with mean zero and an isotropic squared-exponential covariance kernel:

\[ C(x_1, x_2) = \sigma^2 \exp \left( -\frac{\|x_1 - x_2\|^2}{2\ell^2} \right) \]
for which we choose variance $\sigma^2 = 1$ and a length scale $\ell = 0.2$. With this prior, the field can be easily parameterized with a Karhunen-Loeve (K-L) expansion:

$$c(x, \theta) \approx \exp \left( \sum_{i=1}^{d} \theta_i \sqrt{\lambda_i} v_i(x) \right)$$

where $\lambda_i$ and $v_i(x)$ are the eigenvalues and eigenfunctions of the integral operator defined by the kernel $C(\cdot|\cdot)$, and the parameter $\theta_i$ are endowed with independent standard normal priors, $\theta_i \sim \mathcal{N}(0, 1)$, which are the targets of inference. To reduce the dimension of this inference problem, the Karhunen-Loeve expansion is truncated at the first five modes ($d = 5$) and the corresponding mode weights ($\theta_1, \ldots, \theta_5$) are conditioned on data. Data are generated by combining observations of the solution field (solve the PDE on a finer 51-by-51 grid) on a uniform $11 \times 11$ grid covering the unit square with additive independent Gaussian noise.

$$y_j = u(x_j, \theta) + \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, 0.1^2)$$

Consistent with results from previous examples, sgHMC performs substantially better than HMC (Table 3.4).

Figure 3.10: HMC vs sgHMC: an elliptic PDE inverse problem
Table 3.4: Comparing HMC with sgHMC using an elliptic PDE inverse problem. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

<table>
<thead>
<tr>
<th>Method</th>
<th>AR</th>
<th>ESS</th>
<th>s/Iteration</th>
<th>min ESS/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>0.7719</td>
<td>(991.8, 2091.2, 2831.0)</td>
<td>2.02E-1</td>
<td>1.5343</td>
</tr>
<tr>
<td>sgHMC</td>
<td>0.6141</td>
<td>(855.7, 1325.7, 1937.5)</td>
<td>6.1952E-2</td>
<td>4.3165</td>
</tr>
</tbody>
</table>

### 3.5.5 Computational Efficiency as Dimension Increases

As the dimensionality of parameter space increases, the efficiency of the sparse grid interpolation decreases in general. More specifically, it requires more grid points to maintain the quality of approximation which in turn mitigates the benefit of using sparse grid interpolation. To investigate the performance of sgHMC under different dimensionalities, we apply it to large scale ($N = 10^5$) logistic regression models in different dimensions. We choose the step size to keep the acceptance rate around 70% for HMC and collect 4000 samples after 1000 burn-in iterations. Both algorithms are run 10 times and averaged to reduce the random effects on the results.

![Comparison of HMC and sgHMC](image)

Figure 3.11: Comparing HMC and sgHMC under different dimensionalities on logistic regression models

From Figure 3.11, we can see that sgHMC maintains efficient in mediate dimensions. As
the dimensionality increases, the efficiency of sparse grid interpolation drops and the computation gain in speed eventually fails to offset the loss in approximation accuracy (see the acceptance probability in the left panel). At a dimensionality around 40, HMC surpasses sgHMC in our current approach on these examples.

Remark. Note that the efficiency of sparse grid interpolation only depends on the regularity of the target function and the dimensionality of parameter space, which makes it scalable to large scale learning problems and computationally intensive models. Even though so far our sgHMC algorithm can not generalize to extremely high dimensional problems, it can find applications on many important problems with moderate dimensionality and expensive function evaluations, such as learning hyper-parameters and Bayesian uncertainty quantification for differential equations.

3.6 Approximate Target Distribution

So far, our proposed method has been based on using the exact target distribution and approximating the proposal generating mechanism only. We can improve the computation speed even more by using grid approximation for $U$ in the correction step (accept/reject step) as well. In this case, the resulting sampler actually samples from an approximate distribution

$$Q(\theta) \propto \exp(-U^*(\theta))$$

instead of the target posterior distribution $\pi(\theta)$. The bound of the difference between these two distributions measured by the Kullback-Leibler divergence is shown in the following theorem.

Theorem 3.2. If $U$ and $V$ are energy functions corresponding to probability distributions $P$ and $Q$, that is

$$P(\theta) \propto \exp(-U(\theta)), \quad Q(\theta) \propto \exp(-V(\theta))$$
then the Kullback-Leibler divergence between $P$ and $Q$ is bounded by

$$D_{KL}(P\|Q) \leq 2\|U - V\|_{\infty}$$

**Proof.**

$$D_{KL}(P\|Q) = \int_{\mathbb{R}^D} P(\theta) \ln \left( \frac{P(\theta)}{Q(\theta)} \right) d\theta$$

$$= \int_{\mathbb{R}^D} P(\theta)(V(\theta) - U(\theta)) d\theta + \int_{\mathbb{R}^D} P(\theta) \ln \left( \frac{I_Q}{I_P} \right) d\theta$$

where

$$I_P = \int_{\mathbb{R}^D} \exp(-U(\theta)) d\theta, \quad I_Q = \int_{\mathbb{R}^D} \exp(-V(\theta)) d\theta$$

since

$$I_Q = \int_{\mathbb{R}^D} \exp(-V(\theta)) d\theta = \int_{\mathbb{R}^D} \exp(-U(\theta)) \cdot \exp(-(V(\theta) - U(\theta))) d\theta$$

$$\leq \exp(\|V - U\|_{\infty}) \cdot \int_{\mathbb{R}^D} \exp(-U(\theta)) d\theta = \exp(\|V - U\|_{\infty}) \cdot I_P$$

we have

$$D_{KL}(P\|Q) \leq \|V - U\|_{\infty} \cdot \int_{\mathbb{R}^D} P(\theta) d\theta + \|V - U\|_{\infty} \cdot \int_{\mathbb{R}^D} P(\theta) d\theta = 2\|V - U\|_{\infty}$$

Note that if the potential energy function $U$ is a smooth function, $\|U^* - U\|_{\infty} \to 0$ as the grid size goes to 0. By Theorem 3.2, the resulting sampler will eventually sample from the target distribution $\pi(\theta)$.

We apply this method, called GHMC-complete, to the logistic regression and banana-shaped distribution examples discussed above. The posterior samples are shown in Figures 3.12. As
we can see, the posterior samples given by GHMC-complete in both cases match the exact samplers (HMC and GHMC) quite well. With appropriate grid size (around the step size used in HMC), GHMC-complete can provide a high quality approximation to the standard HMC sampler. Moreover, computational efficiency has been substantially improved due to the fast computation of potential energy function in the correction step (Tables 3.5 and 3.6).

For the logistic regression example, Figure 3.13 shows the prediction accuracy vs. the run time for the three algorithms based on a test set. As we can see, the prediction accuracy (measured in terms of the average log-likelihood on the test data) of GHMC-complete increases faster compared to the other two methods. For computationally intensive models, the advantage of GHMC-complete will be more significant since the computation cost of the potential energy function becomes more expensive.
Table 3.5: Comparing HMC with GHMC using a logistic regression model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS.

<table>
<thead>
<tr>
<th>Method</th>
<th>AR</th>
<th>ESS(β₀, β₁)</th>
<th>s/Iteration</th>
<th>min ESS/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>0.9225</td>
<td>(3200, 3200)</td>
<td>7.0157E-4</td>
<td>1425.3707</td>
</tr>
<tr>
<td>GHMC</td>
<td>0.7981</td>
<td>(3200, 3200)</td>
<td>3.318E-4</td>
<td>3013.9031</td>
</tr>
<tr>
<td>GHMC-complete</td>
<td>0.7931</td>
<td>(3191.8, 3200)</td>
<td>2.9237E-4</td>
<td>3411.5275</td>
</tr>
</tbody>
</table>

Table 3.6: Comparing HMC with GHMC using a banana-shaped distribution model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS.

<table>
<thead>
<tr>
<th>Method</th>
<th>AR</th>
<th>ESS(β₁, β₂)</th>
<th>s/Iteration</th>
<th>min ESS/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>0.9353</td>
<td>(2403, 1191.6)</td>
<td>3.8703E-4</td>
<td>962.1346</td>
</tr>
<tr>
<td>GHMC</td>
<td>0.6587</td>
<td>(893.8862, 766.2423)</td>
<td>1.4498E-4</td>
<td>1651.5917</td>
</tr>
<tr>
<td>GHMC-complete</td>
<td>0.6697</td>
<td>(980.1443, 796.2977)</td>
<td>1.2279E-4</td>
<td>2026.6108</td>
</tr>
</tbody>
</table>

3.7 Discussion

Due to its ability of producing distant proposals with high acceptance probability, HMC can provide rapid exploration of the parameter space when sampling from the posterior distribution. However, the gradient computation to obtain essential geometric information prevents its application on computationally intensive problems. To address this issue, we have proposed a relaxed framework, where HMC can take advantage of the smoothness of the potential energy function $U$ in parameter space to accelerate computation by using grid-based precomputing strategies. The key idea is to approximate the force field generated by the potential energy function $U$ through interpolation of those precomputed field at grid points in each HMC iteration. Based on these ideas, two simple grid based algorithms, Naive Grid HMC and Sparse Grid HMC, are proposed and evaluated on several problems. Empirical results show that our approach can capture the main information needed for HMC’s implementation at a lower computational cost. As a result, our method tends to be more effective than standard HMC.
While quite effective in relatively low dimensional problems, extension of grid-based HMC to high dimensional problems could be quite challenging. Future research direction could involve finding effective strategies to alleviate this issue.

Another direction is to find more efficient method to locate the domain of interest. In subsection 3.4 we used Laplace’s approximation for this purpose. As shown in Figure 3.14, this strategy might not be effective when the resulting Gaussian distribution is not a good approximation for the target distributions. An alternative and more general approach is based on following the trajectories of the burn-in samples. Even though MCMC samplers might not converge to the target distribution in the early stage, those trajectories can capture the high density region to some extent. Figure 3.15 shows the cells visited by those early trajectories for the logistic regression example and the banana shaped distribution example.
The proposed precomputing strategy is not limited to HMC only. In fact, it can be integrated with other MCMC methods involving expensive computation of redundant information. For example, Fisher information matrices can be precomputed at each cell center to accelerate Riemannian Manifold HMC (Girolami and Calderhead, 2011).
Chapter 4

Random Network Surrogate

Hamiltonian Monte Carlo

Along with the development in stochastic gradient MCMC algorithms, in recent years several methods have been proposed to construct surrogate Hamiltonians to scale HMC to massive data or complex models. In chapter 3, we introduce two grid based piecewise interpolative surrogate methods based on precomputing strategies (Zhang et al., 2015b). Such grid based methods, however, could be difficult to extend to higher dimensional spaces due to the use of structured grids. Alternatively, we can use Gaussian process models, which are commonly used as surrogate models for emulating expensive-to-evaluate functions, to learn the target functions from early evaluations (training data) (Rasmussen, 2003; Lan et al., 2015; Meeds and Welling, 2014). However, naive (but commonly used) implementations of Gaussian process models have high computation cost associated with inverting the covariance matrix, which grows cubically as the size of the training set increases. This is especially crucial in high dimensional spaces, where we need large training sets in order to achieve a reasonable level of accuracy. Recently, scalable Gaussian processes using inducing point methods (Snelson and Ghahramani, 2006; Quinonero-Candela and Rasmussen, 2005) have been introduced
to scale up GPs to larger datasets. While these methods have been quite successful in reducing computational cost, the tuning of inducing points could still be problematic in high dimensional spaces (see a more detailed discussion in section 4.2).

The key in developing surrogate functions is to effectively capture the collective properties of large scale data sets or complex models with scalability, flexibility and efficiency. In this chapter, we propose to use random nonlinear bases along with efficient learning algorithms to construct surrogate functions that provide effective approximation of the probabilistic model based on the collective behavior of large scale data sets or complex likelihoods. The randomized nonlinear basis functions combined with the computationally efficient learning process can incorporate correct criteria for an efficient implicit subsampling resulting in both flexible and scalable approximations (Huang et al., 2006b,a; Rahimi and Recht, 2007, 2008). Because our method can be presented as a special case of shallow random networks implemented in HMC, we refer to it as random network surrogate Hamiltonian Monte Carlo; however, we will show that our proposed method is related to (and can be extended to) other surrogate methods based on generalized additive models and Gaussian process models, by constructing the surrogate functions using different bases and optimization processes. Our proposed method provides a natural framework to incorporate surrogate functions in the sampling algorithms such as HMC, and it can be easily extended to geometrically motivated methods such as Riemannian Manifold HMC.

In the sequel, we begin with a brief introduction of shallow random network architectures and their approximation properties in section 4.1. Section 4.2 presents and discusses the connection between random network approximations and (sparse) Gaussian processes. In section 4.3, We validate our choice of using geometrical information collected in the history of the Markov chain as training data and show that theoretically our learning procedure of the surrogate function is asymptotically equivalent to potential matching, a novel unnormalized distribution matching strategy similar to score matching (Hyvärinen, 2005; Strathmann
et al., 2015). Next, in section 4.4 we show how the random network surrogate can be used to simulate surrogate induced Hamiltonian flow and accelerate HMC. We develop an adaptive model in section 4.5 to further cut the time budget by utilizing the random network surrogate earlier and improving the approximation quality on the fly. In section 4.6, we use simulated and real-world data to evaluate the performance of our method. Conclusion and discussion on some possible future research directions are made in section 4.7.

4.1 Shallow Random Network Approximation

A typical shallow network architecture (i.e., a single-hidden layer feedforward scalar-output neural network) with $s$ hidden units, a nonlinear activation function $a$, and a scalar (for simplicity) output $z$ for a given $d$-dimensional input $\theta$ is defined as

$$z(\theta) = \sum_{i=1}^{s} v_{i} a(\theta; \gamma_{i}) + b$$

(4.1)

where $\gamma_{i}$ is the $i$th hidden node parameter, $v_{i}$ is the output weight for the $i$th hidden node, and $b$ is the output bias. Given a training data set

$$\mathcal{T} = \{(\theta^{(j)}, t^{(j)})| \theta^{(j)} \in \mathbb{R}^d, t^{(j)} \in \mathbb{R}, j = 1, \ldots, N\}$$

the neural network can be trained by finding the optimal model parameters $W = \{\gamma_{i}, v_{i}, i = 1, \ldots, s\} \cup \{b\}$ to minimize the mean square error cost function,

$$C(W|\mathcal{T}) = \frac{1}{N} \sum_{j=1}^{N} \| z(\theta^{(j)}) - t^{(j)} \|^2$$

(4.2)

The most popular algorithm in machine learning to optimize (4.2) is back-propagation (Rumelhart et al., 1986). However, as a gradient descent-based iterative algorithm, back-
Algorithm 4.1 Extreme Learning Machine

Input: Given a training set $\mathcal{T} = \{(\theta_j, t_j) | \theta_j \in \mathbb{R}^d, t_j \in \mathbb{R}, j = 1, \ldots, N\}$, activation function $a(\theta; \gamma)$ and hidden node number $s$

Step 1: Randomly assign hidden node parameters $\gamma_i, i = 1, \ldots, s$
Step 2: Calculate the hidden layer output matrix $H$

$$H_{ji} = a(\theta_j; \gamma_i), \quad i = 1, \ldots, s, \quad j = 1, \ldots, N$$

Step 3: Calculate the output weight $v$

$$v = H^\dagger t, \quad t = (t_1, t_2, \ldots, t_N)^T$$

where $H^\dagger$ is the Moore-Penrose generalized inverse of matrix $H$

propagation is usually quite slow and can be trapped at some local minimum since the cost function is nonlinear, and for most cases, non-convex. Motivated by the fact that randomization is computationally cheaper than optimization, alternative methods based on random nonlinear bases have been proposed (Huang et al., 2006b; Rahimi and Recht, 2007). These methods drastically decrease the computation cost while maintaining a reasonable level of approximation accuracy. The key feature of random networks is that they reduce the full optimization problem into standard linear regression by mapping the input data to a randomized feature space and then apply existing fast algebraic training methods (e.g., by minimizing squared error) to find the output weight. Given the design objective, algebraic training can achieve exact or approximate matching of the data at the training points. Compared to the gradient descent-based techniques, algebraic training methods can reduce computational complexity and provide better generalization properties. A typical algebraic approach for single-hidden layer feedforward random networks is extreme learning machine (ELM) (Huang et al., 2006b), which is summarized in Algorithm 4.1.

Using randomized nonlinear features, ELM estimates the output weight by finding the least-squares solution to the resulting linear equations system $Hv = t$. Note that presented this way, our method can also be regarded as a random version of Generalized Additive Model
(GAM). In practice, people could add regularization to improve stability and generalizability.

### 4.1.1 Approximation with Random Bases

As shown in Rahimi and Recht (2008), functions in (4.1) with randomly assigned bases are flexible enough to provide accurate approximations for other well-studied classes of functions (e.g., Reproducing Kernel Hilbert Space). Let \( \{a(\cdot; \gamma) : \gamma \in \Gamma \} \) be a family of functions on a compact set \( \Theta \subset \mathbb{R}^d \) with parameter \( \gamma \) specified over the set \( \Gamma \). Let \( p \) be a distribution on \( \Gamma \), consider a rich class of functions of the following form

\[
    f(\theta) = \int_{\Gamma} \alpha(\gamma) a(\theta; \gamma) \, d\gamma \tag{4.3}
\]

where \( |\alpha(\gamma)| \leq C|p(\gamma)|, \forall \gamma \in \Gamma \) for some constant \( C \). Define a norm \( \|f\|_p = \sup_{\gamma} \left| \frac{\alpha(\gamma)}{p(\gamma)} \right| \) and the set

\[
    \mathcal{F}_p = \left\{ f(\theta) = \int_{\Gamma} \alpha(\gamma) a(\theta; \gamma) \, d\gamma \mid \|f\|_p < \infty \right\} \tag{4.4}
\]

The following theorem shows that a given \( f \in \mathcal{F}_p \) can be approximated within \( \mathcal{O}(\|f\|_p / \sqrt{s}) \) by a function of the form

\[
    z(\theta) = \sum_{i=1}^{s} v_i a(\theta; \gamma_i) \tag{4.5}
\]

where \( \gamma_1, \ldots, \gamma_s \) are sampled iid from \( p(\gamma) \). See Rahimi and Recht (2008) for a detailed proof.

**Theorem 4.1** (Rahimi 2008). Let \( \mu \) be any probability measure on \( \Theta \), and define the norm \( \|f\|^2_\mu = \int_{\Theta} f^2(\theta) \mu(d\theta) \). Suppose \( a \) satisfies \( \sup_{\theta, \gamma} |a(\theta; \gamma)| \leq 1 \). Fix \( f \in \mathcal{F}_p \). Then, \( \forall \delta > 0 \), with probability at least \( 1 - \delta \) over \( \gamma_1, \ldots, \gamma_s \) drawn iid from \( p \), there exist \( v_1, \ldots, v_s \) so that the function

\[
    z(\theta) = \sum_{i=1}^{s} v_i a(\theta; \gamma_i)
\]
satisfies

\[ \|z - f\|_\mu < \frac{\|f\|_p}{\sqrt{s}} \left( 1 + \sqrt{2 \log \frac{1}{\delta}} \right) \]  

(4.6)

The activation function \( a \) and probability distribution \( p(\gamma) \) on \( \Gamma \) together defines a Reproducing Kernel Hilbert Space (RKHS) \( \mathcal{H} \) with the following kernel \( k \) on \( \Theta \times \Theta \)

\[ k(\theta, \theta') = \int_{\Gamma} p(\gamma) a(\theta; \gamma) a(\theta'; \gamma) d\gamma \]  

(4.7)

which is clearly positive definite. Alternatively, the following proposition shows that \( \mathcal{H} \) can be constructed based on functions of the form (4.3).

**Proposition 4.1.** Let the space \( \hat{\mathcal{H}} \) be the completion of the set of functions of the form (4.3) such that

\[ \int_{\Gamma} \frac{\alpha(\gamma)^2}{p(\gamma)} d\gamma < \infty \]

with the inner product

\[ \langle f, g \rangle = \int_{\Gamma} \frac{\alpha(\gamma) \beta(\gamma)}{p(\gamma)} d\gamma \]

where \( g(\theta) = \int_{\Gamma} \beta(\gamma) a(\theta; \gamma) d\gamma \). Then \( \hat{\mathcal{H}} = \mathcal{H} \).

**Proof.** See a proof in Rahimi and Recht (2008). \( \square \)

Note that \( \forall f \in \mathcal{F}_p \),

\[ \int_{\Gamma} \frac{\alpha(\gamma)^2}{p(\gamma)} d\gamma = \int_{\Gamma} \frac{\alpha(\gamma)^2}{p(\gamma)^2} p(\gamma) d\gamma \leq \|f\|_p^2 < \infty \]

Therefore, \( \mathcal{F}_p \) is a subset of RKHS \( \mathcal{H} \). In fact, Rahimi and Recht (2008) shows that \( \mathcal{F}_p \) is a dense subset of \( \mathcal{H} \).

**Theorem 4.2.** Let \( \mathcal{F}_p \) and \( \mathcal{H} \) be defined as above for a given activation function \( a(\theta; \gamma) \) and probability density \( p(\gamma) \). Then \( \mathcal{F}_p \) is dense in \( \mathcal{H} \).
Proof. By the property of RKHS, $H$ is the completion of the set of all finite linear combinations of the form

$$f(\theta) = \sum_i v_i k(\theta; \theta_i), \quad \theta_i \in \Theta$$

(4.8)

with the inner product satisfying $\langle k(\cdot, \theta_i), k(\cdot, \theta_j) \rangle = k(\theta_i, \theta_j)$. Using (4.7), we can rewrite (4.8) in the following form

$$f(\theta) = \sum_i v_i \int_{\Gamma} p(\gamma) a(\theta; \gamma) a(\theta_i; \gamma) \, d\gamma = \int_{\Gamma} p(\gamma) \sum_i v_i a(\theta_i; \gamma) a(\theta; \gamma) \, d\gamma = \int_{\Gamma} a_f(\gamma) a(\theta; \gamma) \, d\gamma$$

where $a_f(\gamma) = p(\gamma) \sum_i v_i a(\theta_i; \gamma)$ \Rightarrow $\left| \frac{a_f(\gamma)}{p(\gamma)} \right| \leq \sum_i |v_i|, \forall \gamma \in \Gamma$, implying that $f \in \mathcal{F}_p$. \qed

Since certain RKHSs are known to fit a rich class of (density) functions arbitrarily well, approximating these RKHSs with a small number of random bases allows for efficient surrogate construction with comparable approximation accuracy.

4.1.2 Choice of Nonlinearity

There are many choices for nonlinear activation functions in random networks. Different types of activation functions can be used for different learning tasks. Here, we focus on random networks with two typical types of nonlinear nodes:

- **Additive nodes:**

  $$a(\theta; \gamma) = a(w \cdot \theta + d), \quad w \in \mathbb{R}^d, \quad d \in \mathbb{R}, \quad \gamma = \{w, d\}$$

  where $w$ and $d$ are the weight vector and the bias of the hidden node.
Radial basis functions (RBF) nodes:

\[ a(\theta; \gamma) = a \left( -\frac{\|\theta - c\|^2}{2\ell^2} \right), \quad c \in \mathbb{R}^d, \ell \in \mathbb{R}^+, \quad \gamma = \{c, \ell\} \]

where \(c\) and \(\ell\) are the center and width of the hidden node.

With randomly assigned input weights and biases composed linearly inside the nonlinear activation function, additive nodes form a set of basis functions, whose level sets are hyperplanes orientated by \(w_i\) and shifted by \(d_i\) respectively. Random networks with additive nodes tend to reflect the global structure of the target function. On the other hand, RBF nodes are almost compactly supported (can be adjusted by the width \(\ell\)) rendering good local approximation for the corresponding random networks.

4.2 Connection to GPs and Sparse GPs

It is worth noting the connection between networks with RBF nodes and Gaussian processes models (Rasmussen, 1996; Neal, 1998). Given a training data set

\[ T = \{ (\theta^{(j)}, t^{(j)}) | \theta^{(j)} \in \mathbb{R}^d, t^{(j)} \in \mathbb{R}, j = 1, \ldots, N \} \]

and using squared exponential covariance function

\[ K(\theta^{(j)}, \theta^{(j')}) = \sigma_f^2 \exp \left( -\frac{\|\theta^{(j)} - \theta^{(j')}\|^2}{2\ell^2} \right), \quad \gamma = \{\sigma_f, \ell\} \]

the standard GP regression with a Gaussian noise model has the following marginal likelihood

\[ p(t|Q, \gamma) = \mathcal{N}(t|0, K_N + \sigma^2 I) \]
where \( Q = \{ \theta^{(j)} \}_{j=1}^N \), \( t = \{ t^{(j)} \}_{j=1}^N \), \( [K_N]_{jj'} = K(\theta^{(j)}, \theta^{(j')}) \) is the covariance matrix and \( \sigma \) is the noise parameter. Prediction on a new observation \( \theta^* \) is made according to the conditional distribution

\[
p(t^*|\theta^*, \mathcal{T}, \gamma) = \mathcal{N}(t^*|k^*_M(K_N + \sigma^2 I)^{-1}t, K_{ss} - k^*_M(K_N + \sigma^2 I)^{-1}k_* + \sigma^2)
\]

where \( [k_*]_j = K(\theta^{(j)}, \theta^*) \) and \( K_{ss} = K(\theta^*, \theta^*) \).

On the other hand, if we use \( K(\theta^{(j)}, \cdot) \) as the \( j \)th hidden node, \( j = 1, 2, \ldots, N \), the output matrix becomes \( H = K_N \), and the output weight learned by algebraic approach to a regularized least square problem is

\[
\hat{v} = \arg \min_v \| Hv - t \|^2 + \sigma^2 v^\top K_N v = (K_N + \sigma^2 I)^{-1}t
\]

Therefore, such a network provides the same prediction point estimate as the above full GP models. This way, a Gaussian process model can be interpreted as a self-organizing RBF network where new hidden nodes are added adaptively with new observations. This is also an alternative point of view to Neal (1996) where GP models were shown to be equivalent to single hidden-layer neural networks with infinite many hidden nodes.

Notice that the above GP model scales cubically with the number of data points \( N \) which limits the application of GPs to relatively small datasets. To derive a GP model that is computationally tractable for large datasets, sparse GPs based on inducing point methods (Snelson and Ghahramani, 2006; Quinonero-Candela and Rasmussen, 2005) have been previously proposed. These sparse models introduce a set of inducing points \( \tilde{Q} = \{ \tilde{\theta}^{(i)} \}_{i=1}^M \) and approximate the exact kernel \( K(\theta, \theta') \) by an approximation \( \tilde{K}(\theta, \theta') \) for fast computation. For example, the fully independent training conditional (FITC) (Snelson and Ghahramani,
2006) method uses the approximate kernel

$$\tilde{K}_{\text{FITC}}(\theta, \theta') = k_\theta^v K_M^{-1} k_{\theta'} + \delta_{\theta \theta'}(K(\theta, \theta') - k_\theta^v K_M^{-1} k_{\theta'})$$

where $K_M$ is the exact covariance matrix for the $M$ inducing points and $k_\theta, k_{\theta'}$ are the exact covariance matrices between $\theta, \theta'$ and the inducing points. Given the same training data set $T$, the marginal likelihood is

$$p(t|Q, \gamma) = \mathcal{N}(t|0, K_{NM}K_M^{-1}K_{MN} + \Lambda + \sigma^2 I)$$

Here, $\Lambda$ is a diagonal matrix with $\Lambda_{jj} = K_{jj} - k_j^v K_M^{-1} k_j$ that adjusts the diagonal elements to the ones from the exact covariance matrix $K_N$. Similarly, predictions can be made according to the conditional distribution

$$p(t^*|\theta^*, T, \gamma) = \mathcal{N}(t^*|k_{\theta}^*\Sigma_M^{-1} K_{MN}(\Lambda + \sigma^2 I)^{-1} t, K_{\theta\theta} - k_{\theta}^* (K_M^{-1} - \Sigma_M^{-1}) k_{\theta} + \sigma^2)$$

where $\Sigma_M = K_M + K_{MN}(\Lambda + \sigma^2 I)^{-1} K_{NM}$. The computation cost is reduced to $O(M^2 N)$ for learning and after that the predictive mean costs $O(M)$ per new observation. The hyperparameters $\theta, \sigma$ and inducing points $Q$ can be tuned to maximize the marginal likelihood. However, in high dimension the tuning of $Q$ becomes infeasible.

On the other hand, if we use the inducing points $\bar{Q}$ as the centers of a RBF network, the output matrix $H = K_{NM}$. Given the diagonal matrix $D = \Lambda + \sigma^2 I$, the output weight estimated by the algebraic approach to a weighted least square problem plus a regularization term is

$$\hat{v} = \arg \min_v \|D^{-\frac{1}{2}}(Hv - t)\|^2 + v^t K_M v = \Sigma_M^{-1} K_{MN} (\Lambda + \sigma^2 I)^{-1} t$$

Therefore, the same predictive mean can be reproduced if we use the inducing points as centers and use the same hyperparameter configuration in our random network with RBF
Figure 4.1: Comparing different surrogate approximations with an increasing number of observations $N = 10, 20, 40$ on target function $y = x^2/2$. The observation points are nested samples from the standard normal distribution. For FITC and random networks, we choose 5 inducing points and 5 hidden neurons respectively. FITC and random networks are all run 100 times and averaged to reduce the random effects on the results.

Figure 4.1 compares random networks with different node types and related GP methods on fitting a simple function $y = x^2/2$ which corresponds to the negative log-density of the standard normal distributions. We used softplus function $\sigma(x) = \log(1 + \exp(x))$ in additive nodes and exponential square kernels in RBF nodes and GP methods. As we can see from the graph, random networks generally perform better than GP methods when the number of observations is small. The randomness in the configuration of hidden nodes force networks to learn more globally. In contrast, GP models are more local and need more data to generalize well. By introducing sparsity, FITC tends to generalize better than full GP, especially on small datasizes. Since our goal is to fit negative log-posterior density function in (2.6) and $U(\theta) \to \infty$ as $\theta$ moves away from the high density domain, using softplus basis functions
in random networks are more capable to capture this far field feature by striking a better balance between flexibility and generalization while being less demanding on the datasize. Also, the number of hidden neurons (bases) can be used to regularize the approximation and mitigate overfitting issue.

Unlike typical (naive) Gaussian process models, our random network scales linearly with the number of training points. In fact, a random nonlinear network can be considered as a standard regression model with randomly mapped features. For such shallow random networks, the computational cost for inference is cubic in the number of hidden nodes. Those differences in scaling allow us to explicitly trade off computational efficiency and approximation accuracy and construct more efficient surrogate in certain applications. As our empirical results suggest, with appropriate training data good approximation of smooth functions in high dimensional space can be achieved using a moderate and scalable number of hidden units. Therefore, our proposed method has the potential to scale up to large data sets and provide effective and scalable surrogate Hamiltonians that balance accuracy and efficiency well.

4.3 Choice of Training Data and Potential Matching

To train the random network surrogate efficiently, we want to collect training data in such a way that adequate information of the target distribution can be gathered using a small number of training points. To this end, collecting training data from the history of the Markov chain is a natural choice since the Markov chain eventually follows the target distribution. Here, we analyze the asymptotical behavior of surrogate induced distribution and validate this choice of training data theoretically. Recall that we find our random network surrogate function by minimizing the mean square error (4.2). Similarly to Hyvärinen (2005), it turns out that minimizing (4.2) is asymptotically equivalent to minimizing a new
distance between the surrogate induced distribution and the underlying target distribution, independent of their corresponding normalizing constants.

Suppose we know the density of the underlying intractable target distribution up to a constant

$$\pi(\theta|Y) = \frac{1}{Z} \exp(-U(\theta))$$

where $Z$ is the unknown normalizing constant. Our goal is to approximate this distribution using a parametrized density model, also known up to a constant,

$$Q(\theta, \tau) = \frac{1}{Z(\tau)} \exp(-V(\theta, \tau))$$

Ignoring the multiplicative constant, the corresponding potential energy functions are $U(\theta)$ and $V(\theta, \tau)$ respectively. The straightforward square distance between the two potentials will not be a well-defined measure between the two distributions because of the unknown normalizing constants. Therefore, we use the following distance instead:

$$K(\tau) = \min_{\tau} \int \| V(\theta, \tau) - U(\theta) - d \|^2 \pi(\theta|Y) \, d\theta$$

$$= \int \| V(\theta, \tau) - U(\theta) \|^2 \pi(\theta|Y) \, d\theta - [E_\pi(V(\tau) - U)]^2 = \text{Var}_\pi(V(\tau) - U)$$  \hspace{1cm} (4.9)

Because of its similarity to score matching (Hyvärinen, 2005), we refer to the approximation method based on this new distance as potential matching; the corresponding potential matching estimator of $\tau$ is given by

$$\hat{\tau} = \arg \min_{\tau} K(\tau)$$

It is easy to verify that $K(\tau) = 0 \Rightarrow V(\tau) = U + \text{constant} \Rightarrow Q(\theta, \tau) = \pi(\theta|Y)$, so $K(\tau)$ is a well-defined squared distance. Exact evaluation of (4.9) is usually analytically intractable.
In practice, given $N$ samples from the target distribution $\theta_1, \theta_2, \ldots, \theta_N$, we minimize the empirical version of (4.9)

$$
\tilde{K}(\tau) = \min_d \frac{1}{N} \sum_{n=1}^{N} \|V(\theta_n, \tau) - U(\theta_n) - d\|^2 \\
= \frac{1}{N} \sum_{n=1}^{N} \|V(\theta_n, \tau) - U(\theta_n)\|^2 - \left( \frac{1}{N} \sum_{n=1}^{N} V(\theta_n, \tau) - U(\theta_n) \right)^2 \tag{4.10}
$$

which is asymptotically equivalent to $K$ by the law of large numbers. (4.10) could be more concise if we allow a shift term in the parametrized model ($V(\theta, \tau) = V(\theta, \tau_d) + \tau_d$). In that case, the empirical potential matching estimator is

$$
\hat{\tau} = \arg \min_\tau \tilde{K}(\tau) = \arg \min_\tau \frac{1}{N} \sum_{n=1}^{N} \|V(\theta_n, \tau) + (\tau_d - d) - U(\theta_n)\|^2 \\
= \arg \min_\tau \frac{1}{N} \sum_{n=1}^{N} \|V(\theta_n, \tau_d) + \tau_d - U(\theta_n)\|^2 \\
= \arg \min_\tau \frac{1}{N} \sum_{n=1}^{N} \|V(\theta_n, \tau) - U(\theta_n)\|^2
$$

Now suppose that we have collected the early evaluations from the history of Markov chain

$$
\mathcal{T}_N = \{(\theta^{(1)}, U(\theta^{(1)}), (\theta^{(2)}, U(\theta^{(2)}))), \ldots, (\theta^{(N)}, U(\theta^{(N)}))\}
$$

as training data and use the random network surrogate induced distribution as the parameterized density model, that is,

$$
V(\theta, \tau) = \sum_{i=1}^{s} v_i a(\theta; \gamma_i) + b, \quad \tau = (v, b) \tag{4.11}
$$

Then, the estimated parameter are asymptotically the potential matching estimates

$$
\lim_{N \to \infty} \hat{\tau}_{ELM, \mathcal{T}_N} = \arg \min_\tau \lim_{N \to \infty} C(\tau | \mathcal{T}_N) = \hat{\tau}
$$
Figure 4.2: Comparing HMC and NNS-HMC based on a 2-dimensional banana-shaped distribution. The left panel shows the gradient fields (force map) for the original Hamiltonian flow (red) and the surrogate induced Hamiltonian flow (blue). The middle and right panel show the trajectories for HMC and NNS-HMC samplers. Both samplers start from the same point (red square) with same initial momentums. Blue points at the end of the trajectories are the proposals. The overall acceptance probability drops from 0.97 using HMC to 0.88 using NNS-HMC.

since the Markov chain will eventually converge to the target distribution. When truncated at a finite $N$, the estimated parameters are almost the empirical potential matching estimates except that the samples from the history of the Markov chain are not exactly (but quite close) from the target distribution.

## 4.4 Surrogate Induced Hamiltonian Flow

As discussed in Rasmussen (2003), for computationally intensive models, one can improve computational efficiency of HMC by approximating the energy function and using the resulting approximation to device a surrogate transition mechanism while still converging to the correct target distribution. More specifically, Rasmussen proposed to use pre-convergence samples (which are discarded during the burn-in period) to approximate the energy function
using a Gaussian process model. However, since GPs typically scale cubically with the training data size, it has been challenging to generalize to high dimensional parameter spaces (Neal, 2011). Here, we define an alternative surrogate-induced Hamiltonian as follows:

\[ H^*(\theta, r) = z(\theta) + \frac{1}{2} r^T M^{-1} r \]

where \( z(\theta) \) is the neural network surrogate of the true potential energy \( U(\theta) \). \( H^*(\theta, r) \) now defines a surrogate-induced Hamiltonian flow, parametrized by a trajectory length \( t \), which is a map \( \phi^{H^*}_t : (\theta, r) \rightarrow (\theta^*, r^*) \). Here, \( (\theta^*, r^*) \) is the end-point of the trajectory governed by the following equations

\[
\frac{d\theta}{dt} = \nabla_r H^* = M^{-1} r, \quad \frac{dr}{dt} = -\nabla_{\theta} H^* = -\nabla_{\theta} z(\theta)
\] (4.12)

When the original potential \( U(\theta) \) is computationally costly, simulating the surrogate induced Hamiltonian system (4.12) provides a more efficient proposing mechanism for our HMC sampler. The introduced bias along with discretization error from the leap-frog integrator are all naturally corrected in the MH step where we use the original Hamiltonian in the computation of acceptance probability. As a result, the stationary distribution of the Markov chain will remain the correct target distribution (see Proposition 3.1). Note that by controlling the approximation quality of the surrogate function, we can maintain a relatively high acceptance probability. This is illustrated in Figure 4.2 for a two-dimensional banana-shaped distribution (Girolami and Calderhead, 2011).

As proposed and validated in section 4.3, the early evaluations of the target function during the early iterations of MCMC will be used as the training set based on which we can train a shallow random network using fast algebraic approaches, such as ELM (Algorithm 4.1). The gradient of the scalar output \( z \) (see 4.1) for a network with additive hidden nodes, for
Figure 4.3: Comparing the efficiency of our random network surrogates and Gaussian process surrogates on a challenging 32 dimensional Gaussian target whose covariance matrix has an eigenvector \((1, 1, \ldots, 1)^T\) with a corresponding eigenvalue of 1.0, and all other eigenvalues are 0.01. We set the step size to keep the acceptance probability around 70% for HMC and use the same step size in all surrogate methods. For FITC and random networks, the number of inducing points and hidden neurons are all set to be 1000 to allow reasonably accurate approximation. We ran each algorithm ten times and plot the medians and 80% error bars.

\[
\nabla_{\theta} z(\theta) = \sum_{i=1}^{s} v_i a'(w_i \cdot \theta + d_i)w_i
\]

which costs only \(O(s)\) computations. To balance the efficiency in computation and flexibility in approximation, and to reduce the possibility of overfitting, the number of hidden nodes \(s\) need to be small as long as a reasonable level of accuracy can be achieved. In practice, this can be done by monitoring the resulting acceptance rate using an initial chain.

Following Rasmussen (2003), we propose to run our method, henceforth called random network surrogate Hamiltonian Monte Carlo (RNS-HMC, see Algorithm 4.2), in two phases: exploration phase and exploitation phase. During the exploration phase, we initialize the training data set \(D\) with an empty set or geometrical information gathered from some samples from the prior distribution. We then run the standard HMC algorithm for some iterations and collect information from the new states (i.e., accepted proposals). When we have sufficiently explored the high density domain in parameter space and collected enough training data (during the burn-in period), a shallow random network is trained based on the collected...
Algorithm 4.2 Random Network Surrogate HMC

**Input:** Starting position $\theta^{(1)}$, step size $\varepsilon$ and number of hidden units $s$

Initialize the training data set: $D = \emptyset$ or several random samples from the prior

for $t = 1, 2, \cdots, B$ do

Resample momentum $r$

$r^{(t)} \sim \mathcal{N}(0, M), \ (\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$

Simulate discretization of Hamiltonian dynamics and propose $(\theta^*, r^*)$

Metropolis-Hasting correction:

$u \sim U[0, 1], \ \rho = \exp[H(\theta^{(t)}, r^{(t)}) - H(\theta^*, r^*)]$

if $u < \min(1, \rho)$ then

$\theta^{(t+1)} = \theta^*, \ D = D \cup \{(\theta^*, U(\theta^*))\}$

else

$\theta^{(t+1)} = \theta^{(t)}$

end if

end for

Train a neural network with $s$ hidden units via ELM on $D$ to form the surrogate function $z(\theta)$

for $t = B + 1, B + 2, \cdots T$ do

Resample momentum $r$

$r^{(t)} \sim \mathcal{N}(0, M), \ (\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$

Simulate discretization of a new Hamiltonian dynamics using $z(\theta)$:

for $l = 1$ to $L$ do

$r_{l-1} \leftarrow r_{l-1} - \varepsilon \nabla_{\theta} z(\theta_{l-1})$

$\theta_l \leftarrow \theta_{l-1} + \varepsilon M^{-1} r_{l-1}$

$r_l \leftarrow r_l - \frac{\varepsilon}{2} \nabla_{\theta} z(\theta_l)$

end for

$(\theta^*, r^*) = (\theta_L, -r_L)$

Metropolis-Hasting correction:

$u \sim U[0, 1], \ \rho = \exp[H(\theta^{(t)}, r^{(t)}) - H(\theta^*, r^*)]$

if $u < \min(1, \rho)$ then

$\theta^{(t+1)} = \theta^*$

else

$\theta^{(t+1)} = \theta^{(t)}$

end if

end for

training set $D$ to form a surrogate for the potential energy function. The surrogate function
will be used to approximate the gradient information needed for HMC simulations later in
the exploitation phase.

As an illustrative example, we compare the performance of different surrogate HMC methods.
on a challenging Gaussian target density in 32 dimensions (A lower dimensional case was used in Rasmussen (2003)). The target density has 31 confined directions and a main direction that is 10 times wider, and all variables are correlated. Both the full GPs and FITC methods are implemented using GPML package (Rasmussen and Nickisch, 2010). The results are presented in Figure 4.3. Compared to the full GPs, FITC and random networks (with additive and RBF nodes) all scales linearly with the number of observations. Both random network surrogates can start with fewer training data. We also compare the efficiency of the surrogate induced Hamiltonian flows in terms of time normalized mean effective sample sizes (ESS). The efficiency of FITC and random networks all increases as the number of observations increase until no more approximation gain can be obtained (see the acceptance probability in the middle panel). However, the efficiency of full GP begins to drop before the model reaches its full capacity. That is because its predictive complexity also grows with the number of observations, which in turn diminishes the overall efficiency. Overall, the random network with additive nodes outperform other methods based on this example.

Our proposed method provides a natural framework to incorporate surrogate functions in HMC. Moreover, it can be easily extended to RMHMC. To this end, the Hessian matrix of the surrogate function can be used to construct a metric in parameter space and the third order derivatives can be used to simulate the corresponding modified Hamiltonian flow. We refer to this extended version of our method as RNS-RMHMC.

Note that the approximation quality of the neural network surrogate function depends on several factors including the dimension of parameter space, $d$, the number of hidden neurons, $s$ and the training size, $N$. Here, we assume that $N$ is sufficiently large enough, and investigate the efficiency of RNS-HMC in terms of its acceptance probability for different values of $d$ and $s$ based on a standard logistic regression model with simulated data. Similar to the results presented in Strathmann et al. (2015), Figure 4.4 shows the acceptance rate (over 10 MCMC runs) as a function of $d$ and $s$. For dimensions up to $d \approx 50$, RNS-HMC
4.5 Adaptive RNS-HMC

So far, we have assumed that the neural network model in our method is trained using a sufficiently large enough number of training points after waiting for an adequate number of iterations to allow the sampler explore the parameter space. This, however, could be very time consuming in practice: waiting for a long time to collect a large number of training points could undermine the benefits of using the surrogate Hamiltonian function.

Figure 4.5 shows the average acceptance probabilities (over 10 MCMC chains) as a function of the number of training points, \( N \), and the number of hidden neurons, \( s \), on a simulated logistic regression model for a fixed number of parameters, \( d = 32 \). While it takes around 2000 training data points to fulfill the network’s capability and reach a high acceptance probability...
Figure 4.5: Acceptance probability of the surrogate induced Hamiltonian flow based on a simulated logistic regression models with dimension $d = 32$. **Left:** Acceptance probability as a function of the number of hidden neurons $s$ (x-axis) and the number of training points $N$ (y-axis). **Middle:** Acceptance probability as a function of $N$ for a fixed $s = 1000$. **Right:** Acceptance probability as a function of $s$ for a fixed $N = 1600$.

comparable to HMC, only 500 training points are enough to provide an acceptable surrogate Hamiltonian flow (around 0.1 acceptance probability). Therefore, we can start training the neural network surrogate earlier and adapting it as more training points become available. Although adapting a Markov chain based on its history may undermine its ergodicity and consequently its convergence to the target distribution (Andrieu and Thoms, 2008), we can enforce a vanishing adaption rate $a_t$ such that $a_t \to 0$ and $\sum_{t=1}^{\infty} a_t = \infty$ and update the surrogate function with probability $a_t$ at iteration $t$. By Theorem 5 of Roberts and Rosenthal (2007), the resulting algorithm is ergodic and converges to the right target distribution.

It is straightforward to adapt the neural network surrogate from the history of Markov chain. However, the estimator in Algorithm 4.1 costs $O(Nds + Ns^2 + s^3)$ computation and $O(Ns)$ storage, where $N$ is the number of training data (e.g., the number of rows in the output matrix $H$). As $N$ increases, finding the right weight and bias for the output neuron becomes increasingly difficult. In Greville (2008), Greville shows that $H^\dagger$ can be learned incrementally in real time as new data points become available. Based on Greville’s method, online and adaptive pseudoinverse solutions for updating ELM weights has been proposed.
in van Schaik and Tapson (2015) which can be readily employed here to develop an adaptive version of RNS-HMC. To be more efficient, only the estimator is updated.

**Proposition 4.2.** Suppose the current output matrix is $H_k$ and the target vector is $t_k$. At time $k + 1$, a new sample $\theta_{k+1}$ and the target (potential) $t_{k+1}$ are collected. Denote the output vector from the hidden layer at $\theta_{k+1}$ as $h_{k+1}$. The adaptive updating formula for the empirical potential matching estimator is given by

$$v_{k+1} = v_k + (t_{k+1} - h_{k+1}^\top v_k)b_{k+1}$$

where $b_{k+1}$ and auxiliary matrices $\Phi_{k+1}$, $\Xi_{k+1}$ are updated according to $c_{k+1} = \Phi_k h_{k+1}$.

(i) $c_{k+1} = 0$

$$b_{k+1} = \frac{\Xi_k h_{k+1}}{1 + h_{k+1}^\top \Xi_k h_{k+1}}, \quad \Phi_{k+1} = \Phi_k, \quad \Xi_{k+1} = \Xi_k - \Xi_k h_{k+1} b_{k+1}^\top$$

(ii) $c_{k+1} \neq 0$

$$b_{k+1} = c_{k+1} / \|c_{k+1}\|^2, \quad \Phi_{k+1} = \Phi_k - \Phi_k h_{k+1} b_{k+1}^\top$$

$$\Xi_{k+1} = \Xi_k - \Xi_k h_{k+1} b_{k+1}^\top + (1 + h_{k+1}^\top \Xi_k h_{k+1})b_{k+1} b_{k+1}^\top - b_{k+1} h_{k+1}^\top \Xi_k$$

**Proof.** See the Appendix C.1. \qed

At time $k$, the estimator takes a one-off $O(kds + ks^2 + s^3)$ computation and $O(s^2)$ storage (only need to store $\Phi_k$ and $\Xi_k$, not $H_k^\dagger$). Starting at a previously computed solution $v_K = H_K^\dagger t_K$, and two auxiliary matrices $\Phi_K = I - H_K^\dagger (H_K^\dagger)^\top$, $\Xi_K = H_K^\dagger (H_K^\dagger)^\top$, this adaptive updating costs $O(ds + s^2)$ computation and $O(s^2)$ storage, independent of the training data size $k$.

We refer to this extended version of our method as Adaptive RNS-HMC (ARNS-HMC).
Algorithm 4.3 Adaptive RNS-HMC

**Input:** Initial estimator $v_0$ and auxiliary matrices $\Phi_0, \Xi_0$, adaption schedule $a_t$, step size $\varepsilon$, number of hidden units $s$

Initialize the surrogate $z_0 = z(\theta, v_0)$

for $t = 0, 1, \cdots T$ do

Resample momentum $p$

$r^{(t)} \sim N(0, M), (\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$

Propose $(\theta^*, r^*)$ by simulating the surrogate induced Hamiltonian dynamics with

$\nabla_\theta U^*(\theta) = \nabla_\theta z_t(\theta)$

Metropolis-Hastings correction:

$u \sim U[0, 1], \rho = \exp[H(\theta^{(t)}, r^{(t)}) - H(\theta^*, r^*)]$

if $u < \min(1, \rho)$ then

$\theta^{(t+1)} = \theta^*$

else

$\theta^{(t+1)} = \theta^{(t)}$

end if

update the estimator and auxiliary matrices to $v_{t+1}, \Phi_{t+1}, \Xi_{t+1}$ using $(\theta^{(t+1)}, U(\theta^{(t+1)}))$

$u \sim U[0, 1]$, if $u < a_t$ then

$z_{t+1} = z(\theta, v_{t+1})$

else

$z_{t+1} = z_t$

end if

end for

4.6 Experiments

In this section, we use several experiments based on logistic regression models and inverse problem for elliptic partial differential equation (PDE) to compare our proposed methods to standard HMC and RMHMC in terms of sampling efficiency defined as time-normalized effective sample size (ESS). As before, we use the minimum ESS over all parameters normalized by the CPU time, $s$ (in seconds), as the overall measure of efficiency: $\min(\text{ESS})/s$.

The corresponding step sizes $\varepsilon$ and number of leapfrog steps $L$ for HMC and RMHMC are chosen to make them stable and efficient (e.g., reasonably high acceptance probability and fast mixing). The same settings are used for our methods. Note that while the acceptance rates are similar in the first two examples, they drop a little bit for the last two examples, which is mainly due to the constraints we imposed on our surrogate functions. To prevent
non-ergodicity and ensure high ESS for both HMC and RNS-HMC, we follow the suggestion by Neal (2011) to uniformly sample $L$ from $\{1, \ldots, L\}$ in each iteration. The number of hidden nodes $s$ in random network surrogates are not tuned too much and better results could be obtained by more careful tunings.

In what follows, we first compare different methods in terms of their time-normalized ESS after the burn-in-period. To this end, we collect 5000 samples after a reasonably large number of iterations (5000) of burn-in to make sure the chains have reached their stationary states. For our methods, the accepted proposals during the burn-in period after a short warming-up session (the first 1000 iterations) are used as a training set for a shallow random network. Later, we show the advantages of our adaptive algorithm.

4.6.1 Logistic Regression Model

As our first example, we compare HMC, RMHMC, RNS-HMC, and RNS-RMHMC using a simulation study based on a logistic regression model with 50 parameters and $N = 10^5$ observations. The design matrix is $X = (\frac{1}{10} \mathbf{1}, X_1)$ and true parameters $\beta$ are uniformly sampled from $[0, 1]^{50}$, where $X_1 \sim \mathcal{N}_{49}(0, \frac{1}{100} \mathbf{I}_{49})$. The binary responses $Y = (y_1, y_2, \ldots, y_N)^T$ are sampled independently from Bernoulli distributions with probabilities $p_i = 1/(1 + \exp(-x_i^T \beta))$.

We assume $\beta \sim \mathcal{N}_{50}(\mathbf{0}, 100 \mathbf{I}_{50})$, and sample from the corresponding posterior distribution.

Notice that the potential energy function $U$ is now a convex function, the Hessian matrix is positive semi-definite everywhere. Therefore, we use the Hessian matrix of the surrogate as a local metric in RNS-RMHMC. For HMC and RNS-HMC, we set the step size and leapfrog steps $\epsilon = 0.045$, $L = 6$. For RMHMC and RNS-RMHMC, we set the step size and leapfrog steps $\epsilon = 0.54$, $L = 2$.

To illustrate that our method indeed converges to the right target distribution, Figure 4.6
provides the one- and two-dimensional posterior marginals of some selected parameters obtained by HMC and RNS-HMC. Table 4.1 compares the performance of the four algorithms. As we can see, RNS-HMC has substantially improved the sampling efficiency in terms of time-normalized min(ESS).

Next, we apply our method to two real datasets: Bank Marketing and the a9a dataset (Lin et al., 2008). The Bank Marketing dataset (40197 observations and 24 features) is collected based on direct marketing campaigns of a Portuguese banking institution aiming at predicting if a client will subscribe to a term deposit Moro et al. (2014). We set the step size and number of leapfrog steps $\varepsilon = 0.012$, $L = 45$ for HMC and RNS-HMC; $\varepsilon = 0.4$, $L = 6$ for RMHMC and RNS-RMHMC. The a9a dataset (32561 features and 123 features) is compiled from the UCI adult dataset Bache and Lichman (2013) which has been used to determine whether a person makes over 50K a year. We reduce the number of features to 60 by random projection (increasing the dimension to 100 results in a substantial drop in the acceptance probability). We set the step size and number of leapfrog steps $\varepsilon = 0.012$, $L = 10$ for HMC and RNS-HMC; $\varepsilon = 0.5$, $L = 4$ for RMHMC and RNS-RMHMC. All datasets are normalized to have zero mean and unit standard deviation. The priors are the same as before.
Table 4.1: Comparing the algorithms using logistic regression models and an elliptic PDE inverse problem. For each method, we provide the acceptance probability (AP), the CPU time (s) for each iteration and the time-normalized ESS.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Method</th>
<th>AP</th>
<th>ESS</th>
<th>s/Iter</th>
<th>min(ESS)/s</th>
<th>spdup</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR (Simulation) s = 2000</td>
<td>HMC</td>
<td>0.76</td>
<td>(4351,5000,5000)</td>
<td>0.061</td>
<td>14.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMHMC</td>
<td>0.80</td>
<td>(1182,1496,1655)</td>
<td>3.794</td>
<td>0.06</td>
<td>0.004</td>
</tr>
<tr>
<td></td>
<td>RNS-HMC</td>
<td>0.76</td>
<td>(4449,4999,5000)</td>
<td>0.007</td>
<td>123.56</td>
<td>8.72</td>
</tr>
<tr>
<td></td>
<td>RNS-RMHMC</td>
<td>0.82</td>
<td>(1116,1471,1662)</td>
<td>0.103</td>
<td>2.17</td>
<td>0.15</td>
</tr>
<tr>
<td>LR (Bank Marketing) s = 1000</td>
<td>HMC</td>
<td>0.70</td>
<td>(2005,2454,3368)</td>
<td>0.061</td>
<td>6.52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMHMC</td>
<td>0.92</td>
<td>(1769,2128,2428)</td>
<td>0.631</td>
<td>0.56</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>RNS-HMC</td>
<td>0.70</td>
<td>(1761,2358,3378)</td>
<td>0.007</td>
<td>52.22</td>
<td>8.01</td>
</tr>
<tr>
<td></td>
<td>RNS-RMHMC</td>
<td>0.90</td>
<td>(1974,2254,2457)</td>
<td>0.027</td>
<td>14.41</td>
<td>2.21</td>
</tr>
<tr>
<td>LR (a9a 60 dimension) s = 2500</td>
<td>HMC</td>
<td>0.72</td>
<td>(1996,2959,3564)</td>
<td>0.033</td>
<td>11.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMHMC</td>
<td>0.82</td>
<td>(5000,5000,5000)</td>
<td>3.492</td>
<td>0.29</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>RNS-HMC</td>
<td>0.68</td>
<td>(1835,2650,3203)</td>
<td>0.005</td>
<td>81.80</td>
<td>6.84</td>
</tr>
<tr>
<td></td>
<td>RNS-RMHMC</td>
<td>0.79</td>
<td>(4957,5000,5000)</td>
<td>0.370</td>
<td>2.68</td>
<td>0.22</td>
</tr>
<tr>
<td>Elliptic PDE s = 1000</td>
<td>HMC</td>
<td>0.91</td>
<td>(4533,5000,5000)</td>
<td>0.775</td>
<td>1.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMHMC</td>
<td>0.80</td>
<td>(5000,5000,5000)</td>
<td>4.388</td>
<td>0.23</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>RNS-HMC</td>
<td>0.75</td>
<td>(2306,3034,3516)</td>
<td>0.066</td>
<td>7.10</td>
<td>6.07</td>
</tr>
<tr>
<td></td>
<td>RNS-RMHMC</td>
<td>0.66</td>
<td>(2126,4052,5000)</td>
<td>0.097</td>
<td>4.38</td>
<td>3.74</td>
</tr>
</tbody>
</table>

results for the two data sets are summarized in Table 1. As before, both RNS-HMC and RNS-RMHMC significantly outperform their counterpart algorithms.

### 4.6.2 Elliptic PDE Inverse Problem

Another computationally intensive model is the elliptic PDE inverse problem discussed in Dashti and Stuart (2011); Conard et al. (2014). This classical inverse problem involves inference of the diffusion coefficient in an elliptic PDE which is usually used to model isothermal steady flow in porous media. Let $c$ be the unknown diffusion coefficient and $u$ be the pressure field, the forward model is governed by the elliptic PDE

$$\nabla_x \cdot (c(x, \theta) \nabla_x u(x, \theta)) = 0,$$  \hspace{1cm} (4.14)
where \( \mathbf{x} = (x_1, x_2) \in [0, 1]^2 \) is the spatial coordinate. The boundary conditions are

\[
u(\mathbf{x}, \theta) |_{x_2=0} = x_1, \quad u(\mathbf{x}, \theta) |_{x_2=1} = 1 - x_1, \quad \frac{\partial u(\mathbf{x}, \theta)}{\partial x_1} |_{x_1=0} = 0, \quad \frac{\partial u(\mathbf{x}, \theta)}{\partial x_1} |_{x_1=1} = 0
\]

In our numerical simulation, (4.14) is solved using standard continuous GFEM with bilinear basis functions on a uniform 30 \( \times \) 30 quadrilateral mesh.

A log-Gaussian process prior is used for \( c(\mathbf{x}) \) with mean zero and an isotropic squared-exponential covariance kernel:

\[
C(x_1, x_2) = \sigma^2 \exp \left( - \frac{\|x_1 - x_2\|^2}{2\ell^2} \right)
\]

for which we set the variance \( \sigma^2 = 1 \) and the length scale \( \ell = 0.2 \). Now, the diffusivity field can be easily parameterized with a Karhunen-Loeve (K-L) expansion:

\[
c(\mathbf{x}, \theta) \approx \exp \left( \sum_{i=1}^{d} \theta_i \sqrt{\lambda_i} v_i(\mathbf{x}) \right)
\]

where \( \lambda_i \) and \( v_i(\mathbf{x}) \) are the eigenvalues and eigenfunctions of the integral operator defined by the kernel \( C \), and the parameter \( \theta_i \) are endowed with independent standard normal priors, \( \theta_i \sim \mathcal{N}(0, 0.5^2) \), which are the targets of inference. In particular, we truncate the K-L expansion at \( d = 20 \) modes and condition the corresponding mode weights on data. Data are generated by adding independent Gaussian noise to observations of the solution field on a uniform 11 \( \times \) 11 grid covering the unit square.

\[
y_j = u(x_j; \theta) + \varepsilon_j, \quad \varepsilon_j \sim \mathcal{N}(0, 0.1^2), \quad j = 1, 2, \ldots, N
\]

The number of leap frog steps and step sizes are set to be \( L = 10, \varepsilon = 0.16 \) for both HMC and NNS-HMC. Note that the potential energy function is no longer convex; therefore, we
can not construct a local metric from the Hessian matrix directly. However, the diagonal elements

\[
\frac{\partial^2 U}{\partial \theta_i^2} = \frac{1}{\sigma_\theta^2} + \sum_{j=1}^{N} \frac{1}{\sigma_y^2} \left( \frac{\partial u_j}{\partial \theta_i} \right)^2 - \sum_{j=1}^{N} \frac{\varepsilon_j \partial^2 u_j}{\sigma_y^2 \partial \theta_i^2}, \quad \sigma_\theta = 0.5, \sigma_y = 0.1, \quad i = 1, 2, \ldots, d
\]

are highly likely to be positive in that the deterministic part (first two terms) is always positive and the noise part (last term) tends to cancel out. The diagonals of the Hessian matrix of surrogate therefore induce an effective local metric which can be used in RNS-RMHMC. A comparison of the results of all algorithms are presented in Table 4.1. As before, RNS-HMC provides a substantial improvement in the sampling efficiency. For the RMHMC methods, we set \( L = 3, \varepsilon = 0.8 \). As seen in the table, RMHMC is less efficient than HMC mainly due to the slow computation speed. However, RNS-RMHMC improves RMHMC substantially and outperforms HMC. Although the metric induced by the diagonals of the Hessian matrix of surrogate may not be as effective as Fisher information, it is much cheaper to compute and provide a good approximation.

**Remark.** In addition to the usual computational bottleneck as in previous examples, e.g., large amount of data, there is another challenge on top of that for this example due to the complicated forward model. Instead of a simple explicit probabilistic model that prescribes the likelihood of data given the parameter of interest, a PDE (4.14) is involved in the probabilistic model. The evaluation of geometrical and statistical quantities, therefore, involves solving a PDE similar to (4.14) in each iteration of HMC and RHMHC. This is a preventive factor in practice. Using our methods based on neural network surrogates provide a huge advantage. Numerical experiments show a gain of efficiency by more than 6 times. More improvement is expected as the amount of data increases.
4.6.3 Adaptive Learning

Next, using the above four examples we show that ARNS-HMC can start with far fewer training points and quickly reach the same level of performance as that of RNS-HMC. Figure 4.7 shows that as the number of training points (from initial MCMC iterations) increases, ARNS-HMC fully achieves the network’s capability and reaches a comparable acceptance rate to that of HMC.

We also compare ARNS-HMC to HMC and RNS-HMC in terms of the relative error of mean
Figure 4.8: Relative error of mean as a function of running time.

(REM) which is defined as $\|\bar{\theta}(t) - E(\theta)\|_2 / \|E(\theta)\|_2$, where $\bar{\theta}(t)$ means sample mean up to time $t$. Figure 4.8 shows the results using the four examples discussed above. Note that before training the neural network models, both RNS-HMC and ARNS-HMC are simply standard HMC so the three algorithms have similar performance. As we can see, ARNS-HMC has the best overall performance: it tends to provide lower REM at early iterations. This could be useful if we have limited time budget to fit a model.
4.7 Discussion

In this chapter, we propose an efficient and scalable computational method for Bayesian inference by exploring and exploiting regularity of probability models in parameter space. Our method is based on training surrogate function of the potential energy after exploring the parameter space sufficiently well. For situations where it is not practical to wait for a thorough exploration of parameter space, we have proposed an adaptive version of our method that can start with fewer training points and can quickly reach its full potential.

As an example, we used random networks and efficient learning algorithms to construct effective surrogate functions. These random bases surrogate functions provide good approximations of collective information of the full data set while striking a good balance between accuracy and computational cost for efficient computation. Random networks combined with the optimized learning process can provide flexibility, accuracy, and scalability. Note that in general the overall performance could be sensitive to the architecture of the random network. Our proposed random network surrogate method scales differently than GP emulators because of the specific constraints we imposed on its architecture. As our experimental results show, this approach could improve the performance of HMC in some applications.

In its current form, our method is more effective in problems with costly likelihood and a moderate number of parameters. In spite of improvements we have made to standard HMC, dealing with high dimensional and complex distributions still remains quite challenging. For multimodal distributions, for example, our method’s effectiveness largely depends on the quality of training samples. If these samples are collected from one mode only, the surrogate function will miss the remaining modes and the sampler might not be able to explore them (especially if they are isolated modes). A surrogate function based on Gaussian processes might have a better chance at finding these modes in the tails of the approximate distribution since it tends to go to zero gradually. To address this issue, we can utilize mode searching
and mode exploring ideas such as those proposed by Ahn et al. (2013); Lan et al. (2014).

For HMC, gradient of the potential function is an important driving force in the Hamiltonian dynamics. Although accurate approximation of a well sampled smooth function automatically leads to accurate approximation of its gradient, this is not the case when the sampling is not well distributed. For example, when dense and well sampled training data sets are difficult to obtain in very high dimensions, one can incorporate the gradient information in the training process. In future, we will study more effective way to utilize this information in the training process.
Chapter 5

Variational Hamiltonian Monte Carlo

As the main alternative to MCMC algorithms, variational Bayesian (VB) (Jordan et al., 1999; Wainwright and Jordan, 2008) transforms Bayesian inference problems into optimization problems where a parametrized tractable distribution is introduced to fit the target intractable posterior distribution, by minimizing the Kullback-Leibler divergence with respect to the variational parameters. Compared to MCMC methods, VB introduces bias but is usually faster. A natural question would be: can we combine both methods to mitigate the drawbacks and get the best of both worlds? In recent years, several methods have been proposed based on combining variational Bayesian inference and MCMC simulation in order to improve their overall accuracy and computational efficiency. The first attempt in this direction dates back to de Freitas et al. (2001) where a variational approximation was used as proposal probability in a block Metropolis-Hastings (MH) MCMC kernel to locate the high probability regions quickly, thus facilitating convergence. Recently, a new synthesis of variational inference and MCMC methods has been explored in Salimans et al. (2015) where one or more steps of MCMC are integrated into variational approximation. The extra flexibility from MCMC steps provides a rich class of distributions to provide a closer fit to the exact posterior, which allows for further improvement on the approximation quality.
In this chapter, we explore the possibility of utilizing variational approximation to speed up HMC for computationally intensive models with large scale data sets or complex likelihoods. The key idea is to integrate fast variational approximation into the sampling procedure so that the overall computational complexity can be reduced. To this end, we exploit the regularity in parameter space to construct a free-form approximation of the target distribution by a fast and flexible surrogate function using an optimized additive model of proper random basis, which can be viewed as a single-hidden layer feedforward neural network as well. The surrogate provides sufficiently accurate approximation while allowing for fast computation in the sampling procedure, resulting in an efficient approximate inference algorithm.

In particular, we extend the random network surrogate method discussed in chapter 4 and introduce a general variational framework for it, which further reduces the computational cost by allowing to use the fast surrogate in the M-H correction step. As a result, the modified HMC sampler converges to the best approximation from an exponential family with pre-specified random sufficient statistics rather than the target distribution. This variational perspective distinguishes our approach from the existing surrogate methods on acceleration HMC. We also derive a new training procedure for the random network surrogate. The new procedure resembles score matching (Hyvärinen, 2005) but mitigates the restriction on the training data, making it suitable for learning within the variational framework. Our numerical experiments show that our method successfully combines the advantages of both variational Bayes and MCMC algorithms, and outperforms its VB and MCMC counterparts.

This chapter is organized as follows. Section 5.1 reviews one ingredient related to our method, fixed-form variational Bayes, and discusses its properties, limitations, and some recent advancements. We then introduce and detail our method, termed Variational Hamiltonian Monte Carlo (VHMC), in section 5.2. The efficiency of VHMC is demonstrated on a variety of numerical experiments in section 5.3. Finally, the conclusion and future directions are discussed in section 5.4.
5.1 Fixed-form Variational Bayes

Instead of running a Markov chain, we can approximate the intractable posterior distribution with a more convenient and tractable distribution. A popular approach of obtaining such an approximation is fixed-form variational Bayes (Honkela et al., 2010; Saul and Jordan, 1996; Salimans and Knowles, 2013), where a parametrized distribution $q_\eta(\theta)$ is proposed to approximate the target posterior $\pi(\theta|Y)$ by minimizing the KL divergence

$$D_{KL}(q_\eta(\theta)||\pi(\theta|Y)) = \int q_\eta(\theta) \log \left( \frac{q_\eta(\theta)}{\pi(\theta|Y)} \right) d\theta$$

(5.1)

where $\log(\pi(Y))$ is known as log evidence, a constant used extensively in model selections. Define

$$L(q_\eta(\theta)) = \int q_\eta(\theta) \log \left( \frac{\pi(\theta,Y)}{q_\eta(\theta)} \right) d\theta$$

then, we have

$$D_{KL}(q_\eta(\theta)||\pi(\theta|Y)) = -L(q_\eta(\theta)) + \log \pi(Y)$$

(5.2)

Since KL divergence is always non-negative, $L(q_\eta(\theta))$ are lower bounds on the log evidence. To minimize the KL divergence, it suffices to maximize the lower bounds $L(q_\eta(\theta))$. Usually, $q_\eta(\theta)$ is chosen from the exponential family of distributions with the following canonical form:

$$q_\eta(\theta) = \exp[T(\theta)\eta - A(\eta)]\nu(\theta)$$

(5.3)

where $T(\theta)$ is a row vector of sufficient statistics, $A(\eta)$ is for normalization and $\nu(\theta)$ is a base measure. The column vector $\eta$ is often called the natural parameters of the exponential family distribution $q_\eta(\theta)$. Taking this approach and substituting into (5.2), we now have a
parametric optimization problem in $\eta$:

$$\hat{\eta} = \arg \max_{\eta} \mathbb{E}_{q_{\eta}(\theta)}[\log \pi(\theta, Y) - \log q_{\eta}(\theta)]$$  \hspace{1cm} (5.4)$$

The above optimization problem can be solved using gradient-based optimization or fix-point algorithms if $\mathbb{E}_{q_{\eta}(\theta)}[\log q_{\eta}(\theta)]$, $\mathbb{E}_{q_{\eta}(\theta)}[\log \pi(\theta, Y)]$ and its derivatives with respect to $\eta$ can be evaluated analytically. Without assuming posterior independence and requiring conjugate exponential models, posterior approximations of this type are usually much more accurate than a factorized approximation following the mean-field assumptions. However, the requirement of being able to analytically evaluate those quantities mentioned above is also very restrictive.

### 5.1.1 Variational Bayes as Linear Regression

To mitigate these limitations, Salimans and Knowles (2013) proposed a new optimization algorithm which relates (5.4) to stochastic linear regression. To reveal the connection, the posterior approximate (5.3) is relaxed and rewritten in the unnormalized form

$$\tilde{q}_{\eta}(\theta) = \exp[\tilde{T}(\theta)\tilde{\eta}]\nu(\theta)$$ \hspace{1cm} (5.5)$$

where the nonlinear normalizer $A(\eta)$ is removed and the vectors of sufficient statistics and natural parameters are augmented, i.e. $\tilde{T}(\theta) = (1, T(\theta))$, $\tilde{\eta} = (\eta_0, \eta^T)^T$. Note that if $\eta_0 = -A(\eta)$, (5.5) reduces to the same family of normalized distribution functions as (5.3). To deal with $\tilde{q}_{\eta}(\theta)$, the unnormalized version of KL divergence is utilized, which is given by

$$\tilde{D}_{KL}(\tilde{q}_{\eta}||\pi(\theta, Y)) = \int \tilde{q}_{\eta}(\theta) \log \frac{\tilde{q}_{\eta}(\theta)}{\pi(\theta, Y)} d\nu(\theta) - \int \tilde{q}_{\eta}(\theta) d\nu(\theta)$$

$$= \int \exp[\tilde{T}(\theta)\tilde{\eta}][\tilde{T}(\theta)\tilde{\eta} - \log \pi(\theta, Y)] d\nu(\theta) - \int \exp[\tilde{T}(\theta)\tilde{\eta}] d\nu(\theta)$$ \hspace{1cm} (5.6)$$
Differentiating (5.6) with respect to the natural parameters \( \tilde{\eta} \), we have

\[
\nabla_{\tilde{\eta}} \tilde{D}_{KL}(\tilde{q}_{\tilde{\eta}} || \pi(\theta, Y)) = \int \tilde{q}_{\tilde{\eta}}(\theta)[\tilde{T}(\theta)\top \tilde{T}(\theta)\tilde{\eta} - \tilde{T}(\theta)\top \log \pi(\theta, Y)]d\nu(\theta) \tag{5.7}
\]

Solving \( \nabla_{\tilde{\eta}} \tilde{D}_{KL}(\tilde{q}_{\tilde{\eta}} || \pi(\theta, Y)) = 0 \), we find the minimum

\[
\tilde{\eta} = \mathbb{E}_{q}[\tilde{T}(\theta)\top \tilde{T}(\theta)]^{-1}\mathbb{E}_{q}[\tilde{T}(\theta)\top \log \pi(\theta, Y)] \tag{5.8}
\]

where \( q \) is the normalized density of \( \tilde{q} \). (5.8) is a valid estimate since the Fisher information matrix, \( \mathbb{E}_{q}[\tilde{T}(\theta)\top \tilde{T}(\theta)] \) is virtually non-singular for most identifiable approximating exponential family distribution \( q \).

One interesting observation is that (5.8) resembles the maximum likelihood estimator for a linear regression model where the design matrix is associated with the sufficient statistics \( \tilde{T} \) and the dependent variable is associated with the unnormalized log posterior \( \log \pi(\theta, Y) \). Based on this observation, Salimans and Knowles (2013) derived a stochastic approximation algorithm by viewing (5.8) as a fixed point update. Let \( C = \mathbb{E}_{q}[\tilde{T}(\theta)\top \tilde{T}(\theta)] \) and \( g = \mathbb{E}_{q}[\tilde{T}(\theta)\top \log \pi(\theta, Y)] \). The minimum in (5.8) can be rewritten as \( \tilde{\eta} = C^{-1}g \). Note that \( C \) and \( g \) both depend on \( \eta \), so this does not yet give a solution. However, \( C \) and \( g \) can be iteratively approximated by weighted Monte Carlo as follows. Draw a single sample \( \theta^* \) from the current posterior approximation \( q_{\eta_t}(\theta) \) at each iteration \( t \) and use the update equations

\[
\begin{align*}
g_{t+1} &= (1 - w)g_t + w\hat{g}_t \\
C_{t+1} &= (1 - w)C_t + w\hat{C}_t \tag{5.9}
\end{align*}
\]

for some \( w \in [0, 1] \) where \( \hat{g}_t = \tilde{T}(\theta^*_t)\top \log \pi(\theta^*_t, Y) \) and \( \hat{C}_t = \tilde{T}(\theta^*_t)\top \tilde{T}(\theta^*_t) \). \( w \) is used to downweight earlier iterations when \( q \) was less accurate. The parameters then are updated as \( \tilde{\eta}_{t+1} = C_{t+1}^{-1}g_{t+1} \). In practice, \( w \) is chosen to be small enough to guarantee convergence of the algorithm.
5.2 Variational HMC

Besides subsampling, an alternative approach that can save computational cost is to construct fast and accurate surrogate functions for the expensive potential energy functions (Liu, 2001; Neal, 2011). As one of the commonly used models for emulating expensive-to-evaluate functions, Gaussian process (GP) is used in Rasmussen (2003) to approximate the potential energy and its derivatives based on true values of these quantities (training set) collected during an initial exploratory phase. However, a major drawback of GP-based surrogate methods is that inference time typically grows cubically in the size of training set due to the necessity of inverting a dense covariance matrix. This is especially crucial in high dimensional spaces, where large training sets are often needed before a reasonable level of approximation accuracy is achieved. Our goal, therefore, is to develop a method that can scale to large training set while still maintaining a desired level of flexibility. For this purpose, in chapter 4 we propose RNS-HMC which uses random networks along with efficient training algorithms to construct surrogate functions. A typical single-hidden layer feedforward neural network (SLFN) with scalar output is defined as

\[ z(\theta) = \sum_{i=1}^{s} v_i a(\theta; \gamma_i) \]  

(5.10)

where \( \gamma_i \) is the \( i \)th hidden node parameter, \( v_i \) is the output weight for the \( i \)th hidden node, \( a \) is a nonlinear activation function. Given a training dataset, the estimates of the parameters can be obtained by minimizing the mean square error (MSE) cost function. To save training time, randomly assigned node parameters \( \{\gamma_i\}_{i=1}^{s} \) are suggested in Ferrari and Stengel (2005); Huang et al. (2006b); Rahimi and Recht (2007, 2008) where the optimization is reduced to a linear regression problem with randomly mapped features which can be solved efficiently using algebraic approaches. This can also be viewed as using an additive model based on random (adaptive) basis to approximate the target distribution. Unlike a standard Gaussian
process, the above random network based surrogate scales linearly in the size of training
data, and cubically in the number of hidden neurons. This allows us to explicitly balance
evaluation time and model capacity. However, RNS-HMC uses the true Hamiltonian in the
M-H correction step to ensure convergence to the target distribution, which usually requires
a costly computation. In what follows, we introduce a general variational framework for
(random network) surrogate methods and extend RNS-HMC by allowing to use the fast
surrogate in the M-H correction steps. By doing so, we obtain a new approximation inference
method that combines the advantages of both variational Bayes and MCMC algorithms.

5.2.1 Free-form Variational Bayes

The correspondence between distributions and their potential energy functions builds a
bridge between distribution approximation and function approximation. Viewing this way,
each random neural network in (5.10) corresponds to a distribution in the exponential family

\[ q_v(\theta) \propto \exp(-z(\theta)) = \exp\left[-\sum_{i=1}^{s} v_i a(\theta; \gamma_i) - \Phi(v)\right] \]  

(5.11)

where \( v = (v_1, v_2, \ldots, v_s)^T \) is called the vector of canonical parameters, and the collection of
randomly-mapped features \( \Psi = [\Psi_1, \Psi_2, \ldots, \Psi_s], \Psi_i = -a(\theta; \gamma_i), i = 1, 2, \ldots, s \) is known as
sufficient statistics. The quantity \( \Phi \), known as the log partition function, is defined by the
following integral

\[ \Phi(v) = \log \int \exp(v \cdot \Psi(\theta)) \, d\theta \]  

(5.12)

Note that \( q_v(\theta) \) is properly normalized if and only if the above integral (5.12) is finite.
Therefore, the canonical parameters \( v \) of interest belong to the set

\[ \Omega := \{ v \in \mathbb{R}^s | \Phi(v) < +\infty \} \]  

(5.13)
We call $q_v(\theta)$ the induced distribution of random network surrogate $z(\theta)$ if $v \in \Omega$.

As our random network surrogate approximates the true potential energy function $U(\theta)$, the underlying distribution $q_v(\theta)$ then approximates the target posterior distribution $\pi(\theta|Y)$ (see Theorem 3.2). Because both the surrogate induced distribution $q_v(\theta)$ and the target posterior distribution $\pi(\theta|Y)$ are known up to a constant, we introduce the following expected squared distance between the two unnormalized densities

$$\tilde{D}_{SM}(q_v(\theta)||\pi(\theta,Y)) = \frac{1}{2} \int q_v(\theta)\|\nabla_\theta z(\theta) - \nabla_\theta U(\theta)\|^2 d\theta$$

which is similar to the well known score matching squared distance (Hyvärinen, 2005) (see section 5.2.3 for a more detailed explanation). By minimizing the expected squared distance $\tilde{D}_{SM}$, we arrive at a variational inference algorithm

$$\hat{v} = \arg \min_{v \in \Omega} \tilde{D}_{SM}(q_v(\theta)||\pi(\theta,Y))$$

The surrogate induced approximation (5.11) enriches our choices for variational approximation from fixed-form tractable distributions (e.g., Gaussian or mixture of Gaussians) to fast and flexible intractable distributions. The integral in (5.14) is usually hard to evaluate in practice but can be approximated using Monte Carlo estimates. Unlike the fixed-form approximation, the surrogate induced distribution generally does not allow for drawing samples directly. However, we can use MCMC methods to draw samples from it.

Due to the arbitrary form and random nature in approximation (5.11), we call variational algorithm (5.15) free-form variational Bayes. By choosing a proper number of hidden neurons $s$, the free-form variational Bayes provides an implicit subsampling procedure that can effectively remove redundancy and noise in the data while striking a good balance between computational cost and approximation accuracy of the underlying distribution.
Remark. From an approximation point of view, each \( a(\theta; \gamma) \) is a basis with a random configuration of \( \gamma \), which specifies the orientation and scaling properties, within a profile of the activation function \( a \). In particular we choose the softplus function \( \sigma(x) = \log(1 + \exp(x)) \) to approximate the potential function, e.g., the Hamiltonian corresponding to the posterior distribution, for our free-form variational Bayes. As shown in Zhang et al. (2015a), different basis can be used in our free-form variational Bayes formulation. In particular, if exponential square kernel is used as radial basis functions (RBF) centered at given data, GP method is recovered. However, using kernel functions centered at very data point, GP method does not effectively exploit redundancy in data or regularity in parameter space and hence may result in expensive computational cost as well as instability for large data set. If exponential square kernel is used at a smaller set of properly induced points, a more computationally tractable GP model that tries to exploit redundancy in data, sparse GP (Snelson and Ghahramani, 2006; Quinonero-Candela and Rasmussen, 2005), is recovered. Since kernel function is usually local, to approximate potential functions in HMC which go to infinity at far field, it is shown in (Zhang et al., 2015a) that the use of random basis based on softplus function provides a more compact and better approximation that has a good balance between local features and global behaviors. Also the choice of \( s \), the number of basis, can be used to balance flexibility/accuracy and overfitting.

5.2.2 Surrogate Induced Hamiltonian Dynamics

To sample from the surrogate induced distribution \( q_v(\theta) \), we generate proposals by simulating the corresponding surrogate induced Hamiltonian dynamics governed by the following equations

\[
\begin{align*}
\frac{d\theta}{dt} &= \nabla_r \tilde{H}(r) = M^{-1}r \\
\frac{dr}{dt} &= -\nabla_\theta \tilde{H}(\theta) = -\nabla_\theta z(\theta)
\end{align*}
\] (5.16)
where the modified Hamiltonian is \( \tilde{H}(\theta, r) = z(\theta) + K(r) \). Practitioners can use \textit{leapfrog} scheme to solve (5.16) numerically. The end-point \((\theta^*, r^*)\) of the trajectory starting at \((\theta_0, r_0)\) is accepted with probability

\[
\alpha_{\text{vhmc}} = \min[1, \exp(\tilde{H}(\theta_0, r_0) - \tilde{H}(\theta^*, r^*))]
\]

(5.17)

Note that here (5.17), we use the surrogate induced Hamiltonian \( \tilde{H} \), instead of the true Hamiltonian \( H \). Similar to the true Hamiltonian flow, we have the following theorem

\textbf{Theorem 5.1.} \textit{If we construct a Markov chain by simulating surrogate induced Hamiltonian dynamics (5.16) using leapfrog steps and Metropolis-Hastings correction with the acceptance probability according to (5.17), the equilibrium distribution of the chain is}

\[
\tilde{\pi}(\theta, r) \propto \exp(-z(\theta) - K(r))
\]

\textit{Proof.} See Appendix D.1 for a detailed proof.

Theorem 5.1 implies that the marginal distribution of \( \theta \) is exactly the surrogate induced distribution \( q_v(\theta) \). Therefore, we can run the Markov chain and collect the values of interest, such as the potential energy function and its derivatives, as additional training data to improve the surrogate approximation (solving (5.15)) on the fly. This way, our algorithm can be viewed as a generalized version of the stochastic approximation algorithm proposed in Salimans and Knowles (2013) for \textit{fix-form} variational Bayes.

\subsection*{5.2.3 Score Matching}

A well-known strategy to estimate unnormalized models is score matching (Hyvärinen, 2005). Assuming that data \( D \) come from a distribution with unknown density \( \pi_D(.) \), we want to
find an approximation with a parameterized unnormalized density model \( q_v(.) \), where \( v \) is an \( s \)-dimensional vector of parameters. Score matching estimates the model by minimizing the expected squared distance between the model score function \( \psi_v(\theta) = \nabla_\theta \log q_v(\theta) \) and the data score function \( \psi_D(\theta) = \nabla_\theta \log \pi_D(\theta) \)

\[
J(v) = \frac{1}{2} \int \pi_D(\theta) \| \psi_v(\theta) - \psi_D(\theta) \|^2 \, d\theta \tag{5.18}
\]

A simple trick of partial integration was suggested in Hyvärinen (2005) to avoid the expensive estimation of the data score function \( \psi_D(\theta) \) from \( D \).

**Theorem 5.2** (Hyvärinen 2005). Assume that the model score function \( \psi_v(\theta) \) is differentiable, as well as some weak regularity conditions. Then, the objective function \( J \) in (5.18) can be expressed as

\[
J(v) = \int \pi_D(\theta) \sum_{i=1}^d \left[ \partial_i \psi_v(\theta) + \frac{1}{2} \psi_v(\theta)^2 \right] \, d\theta + C
\]

where \( C \) is a constant that does not depend on \( v \). \( \psi_v(\theta) = \partial_{\theta_i} \log q_v(\theta) \) is the \( i \)-th element of the model score function, and \( \partial_i \psi_v(\theta) = \partial_{\theta_i} \psi_v(\theta) \) is the partial derivative of the \( i \)-th element of the model score function with respect to the \( i \)-th variable.

Similar ideas can be applied here to train our random network surrogate. Notice that the posterior density is known up to a constant, the data score function then can be evaluated exactly \( \psi_D(\theta) = -\nabla_\theta U(\theta) \). Therefore, no partial integration trick is needed. Moreover, it allows us to estimate our density model \( q_v(\theta) \) without requiring samples from the posterior distribution. Since sampling from the posterior distribution is computationally costly, we sample from the simpler and cheaper surrogate induced distribution (our variational model) instead. The corresponding expected squared distance is

\[
\tilde{J}(v) = \frac{1}{2} \int q_v(\theta) \| \psi_v(\theta) - \psi_D(\theta) \|^2 \, d\theta \tag{5.19}
\]
Note that the model score function \( \psi_v(\theta) = -\nabla_\theta z(\theta) \), \( \tilde{J}(v) \) is exactly the expected squared distance \( \tilde{D}_{SM} \) we introduced in section 5.2.1. In the case that our density model is not degenerate, we have a similar local consistency result to Hyvärinen (2005) as shown by the following theorem.

**Theorem 5.3.** Assume that the data density \( \pi_D(.) \) follows the model: \( \pi_D(.) = q_v(.) \) for some \( v^* \). Further, assume that no other parameter value gives a probability density that is equal to \( q_v(.) \), and that \( q_v(\theta) > 0 \) for all \( v, \theta \). Then

\[ \tilde{J}(v) = 0 \iff v = v^* \]

**Proof.** See Appendix D.2.

As such, minimizing the expected squared distance \( \tilde{D}_{SM} \) would be sufficient to estimate the model.

**Remark.** Note that the data score function is

\[ \psi_D(\theta) = \sum_{n=1}^{N} \nabla_\theta \log p(y_n|\theta) + \nabla_\theta \log p(\theta) \]

we may choose our surrogate function as

\[ z(\theta) = \sum_{n=1}^{N} \log p(y_n|\theta) + \log p(\theta) \quad (5.20) \]

then the data density \( \pi_D \) follows the model exactly. In order to reduce computational cost, our model is usually much simpler than (5.20) (\( s \ll N \)). This allows us to explore and exploit the redundancy in the data from a function approximation perspective.

In practice, samples from the surrogate induced distribution are collected as observations and our surrogate can be trained by minimizing the empirical version of \( \tilde{J}(v) \). A regularization
term could be included to improve numerical stability.

Now suppose that we have collected training data of size $t$ from the Markov chain history

$$\mathcal{T}_s^{(t)} := \{(\theta_n, \nabla_\theta U(\theta_n))\}_{n=1}^t \in \mathbb{R}^d \times \mathbb{R}^d$$

where $\theta_n$ is the $n$-th sample. The estimator of the output weight vector (variational parameter) can be obtained by minimizing the empirical square distance between the gradient of the surrogate and the gradient of the potential (i.e., score function) plus an additional regularization term:

$$\hat{v} = \arg\min_v \frac{1}{2} \sum_{n=1}^t \|\nabla_\theta z(\theta_n) - \nabla_\theta U(\theta_n)\|^2 + \frac{\lambda}{2} \|v\|^2$$

which has an online updating formula summarized in the following Proposition 5.1. For ease of presentation, we use the additive nodes $a(\theta; \gamma) = a(w_i \cdot \theta + d_i)$.

**Proposition 5.1.** Suppose our current estimator of the output weight vector is $v^{(t)}$ based on the current training dataset $\mathcal{T}_s^{(t)} := \{(\theta_n, \nabla_\theta U(\theta_n))\}_{n=1}^t \in \mathbb{R}^d \times \mathbb{R}^d$ using $s$ hidden neurons. Given a new training data point $(\theta_{t+1}, \nabla_\theta U(\theta_{t+1}))$, the updating formula for the estimator is given by

$$v^{(t+1)} = v^{(t)} + W^{(t+1)} (\nabla_\theta U(\theta_{t+1}) - A_{t+1} v^{(t)})$$

where

$$W^{(t+1)} = C^{(t)} A_{t+1}^T \left[I_d + A_{t+1} C^{(t)} A_{t+1}^T\right]^{-1}$$

$$A_{t+1} = (A_1(\theta_{t+1}), \ldots, A_s(\theta_{t+1}))$$

with $A_i(\theta_{t+1}) := a'(w_i \cdot \theta_{t+1} + d_i) w_i$, and $C^{(t)}$ can be updated by Sherman-Morrison-Woodbury formula:

$$C^{(t+1)} = C^{(t)} - W^{(t+1)} A_{t+1} C^{(t)}$$

105
Proof. See Appendix D.3.

The estimator and inverse matrix can be initialized as \( \mathbf{v}^{(0)} = 0 \), \( C^{(0)} = \frac{1}{\lambda} I_d \). The online learning can be achieved by storing the inverse matrix \( C \) and performing the above updating formulas, which cost \( O(d^3 + ds^2) \) computation and \( O(s^2) \) storage independent of \( t \).

5.2.4 Variational HMC in Practice

The random network surrogate is capable of approximating the potential energy function well when there is enough training data. However, the approximation could be poor when only few training data are available which is true in the early stage of the Markov chain simulations. To alleviate this issue, we propose to add an auxiliary regularizer which provides enough information for the sampler at the beginning and gradually diminishes as the surrogate becomes increasingly accurate. Here, we use the Laplace’s approximation to the potential energy function but any other fast approximation could be used. The regularized surrogate approximation then takes the form

\[
V_t(\theta) = \mu_t z_t(\theta) + \frac{1}{2}(1 - \mu_t)(\theta - \theta^L)^\top \nabla^2 \theta U(\theta^L)(\theta - \theta^L)
\]

where \( \theta^L \) is the maximum a posteriori (MAP) estimate and \( \mu_t \in [0, 1] \) is a smooth monotone function monitoring the transition from the Laplace’s approximation to the surrogate approximation. Refining the surrogate approximation by acquiring training data from simulating the regularized surrogate induced Hamiltonian flow, we arrive at an efficient approximate inference method: **Variational Hamiltonian Monte Carlo (VHMC)** (Algorithm 5.1).

In practice, the surrogate approximation may achieve sufficient quality and an early stopping could save us from inefficient updating of the output weight vector. In fact, the stopping time \( t_0 \) serves as a knob to control the desired approximation quality. Before stopping, VHMC
Algorithm 5.1 Variational Hamiltonian Monte Carlo

**Input:** Regularization coefficient \( \lambda \), transition function \( \mu_t \), number of hidden neurons \( s \), starting position \( \theta^{(1)} \) and HMC parameters

Find the Maximum A Posterior \( \theta^L \) and compute the Hessian matrix \( \nabla^2_{\theta} U(\theta^L) \)

Randomly assign the input weights and bias: \( \{w_i\}_{i=1}^s \) and \( \{d_i\}_{i=1}^s \)

For \( t = 1, 2, \ldots, T \) do

Propose \( (\theta^*, r^*) \) with regularized surrogate induced Hamiltonian flow, using \( \nabla_{\theta} V_t(\theta) \)

Perform Metropolis-Hasting step according to the underlying distribution \( \pi_t \sim \exp(-V_t(\theta) - K(r)) \)

If New state is accepted & \( t < t_0 \) then

Acquire new training data point \( (\theta_{t+1}, \nabla_{\theta} U(\theta_{t+1})) \)

Update the output weight estimate \( v^{(t+1)} \leftarrow (5.23) \) and the inverse matrix \( C^{(t+1)} \leftarrow (5.24) \)

Else

\( v^{(t+1)} = v^{(t)}, C^{(t+1)} = C^{(t)} \)

End if

End for

acts as a free-form variational Bayes method that keeps improving itself by collecting training data from the history of the Markov chain. After stopping, VHMC performs as a standard HMC algorithm which samples from the surrogate induced distribution. VHMC successfully combines the advantages of both variational Bayes and Hamiltonian Monte Carlo, resulting in higher computational efficiency compared to HMC and better approximation compared to VB.

5.3 Experiments

In this section, four experiments are conducted to verify the effectiveness and efficiency of the proposed Variational HMC framework. In the first two examples, we demonstrate the performance of VHMC from a pure approximation perspective and compare it to state-of-the-art fix-form variational Bayes methods. We then test the efficiency of VHMC on two machine learning problems with large datasets and compare our methods to SGLD (Welling and Teh, 2011), which is one of the state-of-the-art stochastic gradient MCMC methods.
5.3.1 A Beta-binomial Model for Overdispersion

We first demonstrate the performance of our variational Hamiltonian Monte Carlo method on a toy example from Albert (2009), which considers the problem of estimating the rates of death from stomach cancer for the largest cities in Missouri. The data is available from the R package LearnBayes which consists of 20 pairs \((n_j, y_j)\) where \(n_j\) records the number of individuals that were at risk for cancer in city \(j\), and \(y_j\) is the number of cancer deaths that occurred in that city. The counts \(y_j\) are overdispersed compared to what would be expected under a binomial model with a constant probability; therefore, Albert (2009) assumes a beta-binomial model with mean \(m\) and precision \(K\):

\[
p(y_j|m, K) = \binom{n_j}{y_j} \frac{B(Km + y_j, K(1 - m) + n_j - y_j)}{B(Km, K(1 - m))}
\]

and assigns the parameters the following improper prior:

\[
p(m, K) \propto \frac{1}{m(1 - m)} \frac{1}{(1 + K)^2}
\]

The resulting posterior is extremely skewed (as shown in the bottom right corner in Figure 5.1) and a reparameterization

\[
x_1 = \text{logit}(m), \ x_2 = \text{logit}(K)
\]

is proposed to ameliorate this issue.

We choose \(\mu_t = 1 - \exp(-t/200)\) as our transition schedule and set up the HMC parameter to achieve around 85% acceptance. We run the variational Hamiltonian Monte Carlo long enough so that we can estimate the full approximation quality of our surrogate. We then train the neural network based surrogate using different numbers of hidden neurons and examine the resulting KL-divergence and score matching squared distance to the true pos-
Figure 5.1: Approximate posteriors for a varying number of hidden neurons. Exact posterior at bottom right.

Figure 5.2: KL-divergence and score matching squared distance between the surrogate approximation and the exact posterior density using an increasing number of hidden neurons.

As we can see from Figures 5.1 and 5.2, the neural network based surrogate indeed offers a high quality approximation and becomes more accurate as the number of hidden neurons increases. The surrogate induced Hamiltonian flow effectively explores the parameter space and transfers information from the posterior to the surrogate.
5.3.2 Bayesian Probit Regression

Next, we demonstrate the approximation performance of our Variational HMC algorithm relative to existing variational approaches on a simple Bayesian classification problem, binary probit regression. Given $N$ observed data pairs $\{(y_n, x_n) | y_n \in \{0, 1\}, x_n \in \mathbb{R}^d\}_{n=1}^N$, the model comprised a probit likelihood function $P(y_n = 1 | \theta) = \Phi(\theta^\top x_n)$ and a Gaussian prior over the parameter $p(\theta) = \mathcal{N}(0, 100)$, where $\Phi$ is the standard Gaussian cdf. A full covariance multivariate normal approximation is used for all variational approaches. The synthetic data we use are simulated from the model, with $N = 10000$ and $d = 5$. We show the performance averaged over 50 runs for all methods. We compare our algorithm to Variational Bayesian Expectation Maximization (VBEM) (Beal and Ghahramani, 2002; Ormerod and Wand, 2010), and the fixed-form variational approximation of Salimans and Knowles (2013).

For all variational approaches, we initialize the posterior approximation to the prior. For our Variational HMC algorithm, we choose $s = 100$ random hidden units for the surrogate and set the starting point to be the origin. The number of hidden units is chosen in such a way that the surrogate is flexible enough to fit the target well and remain fast in computation. The HMC parameters are set to make the acceptance probability around 70%. The target density is almost Gaussian, and a fast transition $\mu_t = 1 - \exp(-t/5)$ is enough to stabilize our algorithm. The approximation performance is assessed in terms of the root mean squared error (RMSE) between the estimate (variational mean for VB and sample mean for VHMC) and the true parameter that is used to generate the dataset.

Figure 5.3 shows the performance of our Variational HMC algorithm, as well as the performance of the other two variational Bayes methods. As we can see from the graph, VHMC and the subsampling based fixed-form variational approach (FF-minibatch) achieve lower RMSE than the VBEM algorithm. That is because of the extra factorization assumptions made by VBEM when introducing the auxiliary variables (Ormerod and Wand, 2010). Even though Gaussian approximation is already sufficiently accurate on this simple example, VHMC can
Figure 5.3: RMSE approximate posterior mean as a function of the number of likelihood evaluations for different variational Bayesian approaches and our Variational HMC algorithm.

still arrive at a lower RMSE due to the extra flexibility provided by the free-form random network surrogate function.

5.3.3 Bayesian Logistic Regression

Next, we apply our Variational HMC method to a Bayesian logistic regression model. Given the $i$-th input vector $x_i$, the corresponding output (label) $y_i = \{0, 1\}$ is assumed to follow the probability

$$p(y_i = 1 | x_i, \beta) = 1/(1 + \exp(-x_i^\top \beta))$$

and a Gaussian prior $p(\beta) = N(0, 100)$ is used for the model parameter $\beta$. We test our proposed algorithm on the a9a dataset (Lin et al., 2008). The original dataset, which is compiled from the UCI adult dataset, has 32561 observations and 123 features. We use a 50 dimension random projection of the original features. We choose $s = 2000$ hidden units for the surrogate and set a transition schedule $\mu_t = 1 - \exp(-t/500)$ for our VHMC algorithm. We then compare the algorithm to HMC (Duane et al., 1987; Neal, 2011) and to stochastic
gradient Langevin dynamics (SGLD) (Welling and Teh, 2011). For HMC and VHMC, we set the leap-frog stepsize such that the acceptance rate is around 70%. For SGLD we choose batch size of 500 and use a range of fixed stepsizes.

Following Ahn et al. (2012), we investigate the time normalized effective sample size (ESS)\(^1\) averaged over the 51 parameters and compare this with the relative error after a fixed amount of computation time. The relative error of mean (REM) and relative error of covariance (REC) is defined as

\[
REM_t = \frac{\sum_i |\beta^t_i - \beta^o_i|}{\sum_i |\beta^o_i|}, \quad REC_t = \frac{\sum_{ij} |C^t_{ij} - C^o_{ij}|}{\sum_{ij} |C^o_{ij}|}
\]

where \( \overline{\beta}^t = \frac{1}{t} \sum_{t'=1}^t \beta_{t'} \), \( C^t = \frac{1}{t} \sum_{t'=1}^t (\beta_{t'} - \overline{\beta}_t)(\beta_{t'} - \overline{\beta}_t)^t \) are the sample mean and sample covariance up to time \( t \) and the ground truth \( \beta^o \), \( C^o \) are obtained using a long run (\( T = 500K \) samples) of HMC algorithm.

Figure 5.4 shows the relative error at time \( T = 300, T = 3000 \) as a function of the time normalized mean ESS, which is a measure of the mixing rate. The results for the mean are shown on the top, and those for the covariance are on the bottom. We run each algorithm with a different setting of parameters that control the mixing rate: number of leap-frog steps \( L = [50, 40, 30, 20, 10, 5, 1] \) for HMC and \( L = [50, 40, 30, 20, 10, 5] \) for VHMC, and stepsizes \( \epsilon = [2e^{-3}, 1e^{-3}, 5e^{-4}, 1e^{-4}, 5e^{-5}, 1e^{-5}] \) for SGLD.

As we decrease the stepsize, SGLD becomes less biased in the gradient approximation, resulting in smaller relative error. However, the exploration efficiency drops at the same time and sampling variance gradually dominates the relative error. In contrast, HMC uses a fixed leap-frog stepsize and therefore maintains high exploration efficiency in parameter space. The down side is the expensive computation of the full gradient and the possible

---

\(^1\)Given \( B \) samples, \( \text{ESS} = B[1 + 2 \sum_{k=1}^K \gamma(k)]^{-1} \), where \( \gamma(k) \) is the sample autocorrelation at lag \( k \) (Geyer, 1992)
Figure 5.4: Final error of logistic regression at time $T$ versus mixing rate for the mean (top) and covariance (bottom) estimates after 300 (left) and 3000 (right) seconds of computation. Each algorithm is run using different setting of parameters.

turning back of the trajectories when the number of leap-frog steps is unnecessarily large. Adopting a flexible neural network surrogate, VHMC balances the computational cost and approximation quality much better than subsampling and achieves lower relative error with high mixing rates.

5.3.4 Independent Component Analysis

Finally, we apply our method to sample from the posterior distribution of the unmixing matrix in Independent Component Analysis (ICA) (Hyvärinen and Oja, 2000). Given $N$
$d$-dimensional observations $X = \{x_n \in \mathbb{R}^d\}_{n=1}^N$, we model the data as

$$p(x|W) = |\text{det}(W)| \prod_{i=1}^{d} p_i(w_i^T x)$$

, where $w_i$ is the $i$-th row of $W$ and $p_i$ is supposed to capture the true density of the $i$-th independent component. Following Welling and Teh (2011), we use a Gaussian prior over the unmixing matrix $p(w_{ij}) = \mathcal{N}(0, \sigma)$ and choose $p_i(y_i) = [4 \cosh(\frac{1}{2} y_i)]^{-1}$ with $y_i = w_j^T x$. We evaluate our method using the MEG dataset (Vigário et al., 1997), which has 122 channels and 17730 observations. We extract the first 5 channels for our experiment which leads to samples with 25 dimensions. We then compare our algorithm to standard HMC and SGLD (Welling and Teh, 2011). For SGLD, we use a natural gradient (Amari et al., 1996) which has been found to improve the efficiency of gradient descent significantly. We set $\sigma = 100$ for the Gaussian prior. For HMC and Variational HMC, we set the leap-frog stepsize to keep the acceptance ratio around 70% and set $L = 40$ to allow an efficient exploration in parameter space. For SGLD, we choose batch size of 500 and use stepsizes from a polynomial annealing schedule $a(b + t)^{-\delta}$, with $a = 5 \times 10^{-3}, b = 10^{-4}$ and $\delta = 0.5$. (This setting reduces the stepsize from $5 \times 10^{-5}$ to $1 \times 10^{-6}$ during $1e+7$ iterations). We choose $s = 1000$ hidden units and set the transition schedule $\mu_t = 1 - \exp(-t/2000)$ for our Variational HMC algorithm. 

To measure the convergence of the samplers, we use the Amari distance (Amari et al., 1996) $d_A(W, W_0)$, where $W$ is the sample average and $W_0$ is the true unmixing matrix estimated using a long run ($T = 100K$ samples) of standard HMC algorithm.

The Amari distance as a function of runtime is reported for each of these methods in Figure 5.5. From the graph, we can see that SGLD converges faster than standard HMC. The bias introduced by subsampling is compensated by the fast exploration in parameter space which reduce the sample variance. However, the exploration efficiency of SGLD decreases as the stepsize is annealed. By maintaining efficient exploration in parameter space (same stepsize as HMC) while reducing the computation in simulating the Hamiltonian flow, VHMC
outperforms SGLD, arriving at a lower Amari distance much more rapidly.

5.4 Discussion

In this chapter, we introduce a general variational framework for (random network) surrogate methods which further reduces the computational cost by allowing to use the fast surrogate in the M-H correction step. Based on the framework, we present a novel approach, Variational Hamiltonian Monte Carlo, for approximate Bayesian inference. The flexible and efficient random network surrogates induce a large family of distributions which enrich our choice for variational approximation from fix-form tractable distributions to flexible free-form intractable distributions. The surrogate keeps refining its approximation by collecting training data while the sampler is exploring the parameter space. This way, our algorithm can be viewed as a generalized version of the stochastic approximation algorithm proposed in Salimans and Knowles (2013) for fix-form variational Bayes. Unlike subsampling-based MCMC methods, VHMC maintains the relatively high exploration efficiency of its MCMC
counterparts while reducing the computational cost. Compared to fixed-form variational approximation, VHMC is more flexible and thus can approximate more general target distribution better.

As mentioned in chapter 4, the random network surrogate is more effective in problems with costly likelihood and a moderate number of parameters. In really high dimensions, one can adopt more sophisticated structure (e.g., deep networks) to approximate the target potential function. However, the associated training could be costly and the benefit from using the efficient surrogate might be undermined. In its current form, the training data are collected from the exact gradient of the potential evaluated at pre-convergent samples. It would be interesting to collect noisy unbiased estimate of the true gradient based on subsampling as our training data. The introduced noise can be automatically filtered when training the random network surrogate in (5.22), a procedure similar to linear regression. This way, VHMC can be viewed as stochastic gradient MCMC methods plus an automatic noise filter which reduces the amount of the induced noise and allows for larger step sizes (higher exploration efficiency).
Chapter 6

Conclusion

Modern Bayesian statistics has witnessed the success of Markov chain Monte Carlo algorithms which allow for intractable integration and proper uncertainty quantification. However, due to the random walk nature of movements, simple MCMC methods often suffer from slow mixing and take a long time to converge. While modern MCMC methods suppress random walk behaviors by leveraging continuous dynamics to define a transition kernel that efficiently explores the target distribution, they often need expensive geometrical quantities from the model. To accelerate MCMC algorithms, recent focuses have been mainly on stochastic gradient MCMC where stochastic gradients are used in place of full-data gradients. However, the noise introduced by subsampling could lead to non-ignorable loss of accuracy, which in turn exerts a substantially negative impact on exploration efficiency. As an alternative, surrogate methods reduce the computation cost by finding computationally cheaper surrogate functions to substitute the expensive log-posterior. By exploiting smoothness or regularity in parameter space, surrogate methods reduce the computation cost while maintaining comparable exploration efficiency. Nevertheless, current surrogate methods are often limited to moderate dimensional problems by the cost of acquiring such surrogates with desired approximation accuracy.
In this thesis, we derived a series of surrogate methods that can be potentially generalized to higher dimensional problems, including two grid based surrogate Hamiltonian Monte Carlo, random network surrogate HMC and variational Hamiltonian Monte Carlo to scale MCMC algorithms to large data sets or complex models. We summarize the main contributions as follows:

- We provide a general framework where efficient numerical approximation can be used to design scalable MCMC algorithms.
- We introduce two grid based precomputing methods to construct efficient surrogate functions that work for problems of different dimensionalities and provide an upper bound for the effectiveness of the surrogate.
- We provide a faster alternative to GPs based surrogate methods which is based on random nonlinear bases and derive an efficient potential matching training procedure for the random bases surrogate.
- We derive a general variational framework for (random network) surrogate methods, based on which we developed an efficient approximation inference algorithm that combines the advantages of both variational Bayes and MCMC algorithms.

Although we discussed these surrogate methods in the context of HMC, the proposed framework can be easily extended to other MCMC samplers.

Unlike stochastic gradient MCMC algorithms that sacrifice the efficient exploration of Hamiltonian flow to improve the scalability of the algorithm, surrogate methods stand a chance to reduce the computation cost while maintaining the coherent exploration of their MCMC counterparts, given that the redundancy of the model can be properly exploited to construct efficient surrogate transition kernels with good approximation quality. The work presented here is by no means a comprehensive collections of surrogate methods for scalable MCMC
algorithms. Instead, it opens many potential directions for future research, both in terms of
designing efficient surrogate methods by using other flexible structures or problem-specific
approximation methods, and applying our results and techniques to solve other computa-
tionally challenging Bayesian inference problems such as learning hyper-parameters and
Bayesian uncertainty quantification for differential equations.

6.1 Future Directions

While our proposed surrogate methods proved efficient in problems with costly likelihood and
a moderate number of parameters, dealing with high dimensional and complex distributions
still remains quite challenging. In addition, the idea of combining variational Bayes and
MCMC algorithms requires a deeper investigation, especially in designing scalable MCMC
algorithms. In the following, I list some of the possible future research directions.

High Dimensional Problems In general, approximating functions in high dimensional
space is a challenging problem. As the dimension increases, simple grid based methods will
fail due to the curse of dimensionality. As more sophisticated structures such as deep neural
networks prove useful in learning high dimensional objects in machine learning society, it
would be interesting to train such deep architectures as our surrogate in high dimensional
Bayesian inference problems. Another issue in high dimensional problems is that the target
distribution could have multimodes. In that case, our method’s effectiveness largely depends
on the quality of training data collected in the history of the Markov chain and we may need
to resort to mode searching and mode exploring techniques proposed by Ahn et al. (2013);
Lan et al. (2014)

Variational Bayes plus MCMC The gap between variational Bayes and MCMC have
been bridged recently to derive efficient approximate Bayesian inference algorithms. Viewing
the stochastic Markov chain as a variational approximation with intermediate states being a set of auxiliary random variable, Salimans et al. (2015) propose a set of efficient inference methods that incorporates flexible MCMC steps into variational approximations. Mandt et al. (2016) introduces a variational perspective for stochastic gradient descent (SGD) with a constant learning rate. The constant SGD is interpreted as a stochastic process with a stationary distribution and approximation inference therefore can be performed by running constant SGD with hyper-parameters tuned to minimize the distance between its stationary distribution and the exact posterior. Moreover, this variational perspective can be used to analyze stochastic gradient MCMC algorithms with constant step sizes and derive criterion for the involved hyper-parameters such as to enhance numerical stability and exploration efficiency. For our variational Hamiltonian Monte Carlo method (Zhang et al., 2016), stochastic gradients can also be used as training data and the surrogate training process serves as a variance reduction procedure which allows for larger step sizes. Overall, the marriage of variational Bayes and (scalable) MCMC algorithms provides a promising means of designing efficient scalable Bayesian inference methods.

**Uncertainty Quantification in Bayesian Inversion** Many inverse uncertainty quantification problems arising in the sciences and engineering require the incorporation of data into model. Bayesian approach provide a natural, rigorous and well-posed framework for doing this (Dashti and Stuart, 2011, 2015). The Bayesian formulation enables direct application of the classical theory of regularization and leads to new algorithmic approaches which build on the full power of analysis and numerical analysis to leverage the structure of the infinite dimensional inference problem. However, the model likelihoods usually involve numerical solutions to PDEs and thus are expensive to evaluate. Our proposed random network surrogate HMC and variational HMC algorithms, therefore, provide a promising way to design scalable MCMC algorithms for these “big model” problems.
Bibliography


Appendix A

Hamiltonian Monte Carlo

A.1 Proof of Theorem 2.1

Proof. Let $z_0 = (\theta(t_0), r(t_0))$ denotes the initial state. the flow $\phi_s^H$ satisfies Hamilton equations

$$\frac{d\phi_s^H(z_0)}{ds} = J\nabla z H(\phi_s^H(z_0))$$

The derivative $\partial\phi_s^H/\partial z_0$, therefore, is a solution of the following equations

$$\frac{d\Phi}{ds} = J\nabla z^2 H(\phi_s^H(z_0))\Phi$$

where $\nabla z^2 H$ is the Hessian matrix of $H$ (therefore is symmetric). We therefore obtain

$$\frac{d}{ds} \left( \left( \frac{\partial\phi_s^H}{\partial z_0} \right)^T J^{-1} \left( \frac{\partial\phi_s^H}{\partial z_0} \right) \right) = \left( \frac{d}{ds} \frac{\partial\phi_s^H}{\partial z_0} \right)^T J^{-1} \left( \frac{\partial\phi_s^H}{\partial z_0} \right) + \left( \frac{\partial\phi_s^H}{\partial z_0} \right)^T J^{-1} \left( \frac{d}{ds} \frac{\partial\phi_s^H}{\partial z_0} \right)$$

$$= \left( \frac{\partial\phi_s^H}{\partial z_0} \right)^T \nabla z^2 H(\phi_s^H(z_0)) J^T J^{-1} \left( \frac{\partial\phi_s^H}{\partial z_0} \right) + \left( \frac{\partial\phi_s^H}{\partial z_0} \right)^T \nabla z^2 H(\phi_s^H(z_0)) \left( \frac{\partial\phi_s^H}{\partial z_0} \right) = 0$$
since $J^t = -J$ and $J^tJ^{-1} = -I$. Note that the relation

$$(\frac{\partial \phi^H_s}{\partial z_0})^\top J^{-1} \left( \frac{\partial \phi^H_s}{\partial z_0} \right) = J^{-1}$$

is satisfied for $s = 0$ because $\phi^H_0$ is the identity map, it is satisfied for all $s$ and all $z_0$, as long as the solution remains in the domain of definition of $H$. \qed
Appendix B

Sparse Grid

B.1 Proof of Lemma 3.1

Proof. We follow Bungartz and Griebel (2004). Note that the number of points employed by the hierarchical increment \( W_i = \Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d} \) is \( 2^{i_1 - 1} \) and the employed points are exclusive between different \( W_i \)'s, we get

\[
|\tilde{H}_{n+d-1,d}| = \sum_{|i| \leq n+d-1} 2^{i_1 - 1} = \sum_{l=d}^{n+d-1} 2^{l-d} \sum_{|i|=l} 1
\]

\[
= \sum_{l=d}^{n+d-1} 2^{l-d} \cdot \binom{l-1}{d-1} = \sum_{l=0}^{n-1} 2^l \cdot \binom{l + d - 1}{d - 1}
\]

Using the \((d - 1)\)-derivative of power functions, we can express the summand as follows

\[
\sum_{l=0}^{n-1} 2^l \cdot \binom{l + d - 1}{d - 1} = \frac{1}{(d - 1)!} \cdot \sum_{i=0}^{n-1} (x_{i+d-1})^{(d-1)}|_{x=2}
\]
\[
= \frac{1}{(d-1)!} \cdot \left( x^{d-1} \cdot \frac{1 - x^n}{1 - x} \right)^{(d-1)} \bigg|_{x=2} \\
= \frac{1}{(d-1)!} \cdot \sum_{l=0}^{d-1} \binom{d-1}{l} \cdot (x^{d-1} - x^{n+d-1})^l \cdot \left( \frac{1}{1 - x} \right)^{(d-1)-l} \bigg|_{x=2} \\
= (-1)^d + 2^n \cdot \sum_{l=0}^{d-1} \binom{n + d - 1}{l} \cdot (-2)^{d-1-l}
\]

The summand for \( l = d - 1 \) is the largest one and it holds

\[
2^n \cdot \frac{(n + d - 1)!}{(d-1)!n!} = 2^n \cdot \left( \frac{n^{d-1}}{(d-1)!} + O(n^{d-2}) \right)
\]
Appendix C

Random Network Surrogate HMC

Theorem C.1 (Greville). If a matrix $A$, with $k$ columns, is denoted by $A_k$ and partitioned as $A_k = [A_{k-1} \ a_k]$, with $A_{k-1}$ a matrix having $k - 1$ columns, then the Moore-Penrose generalized inverse of $A_k$ is

$$A_k^\dagger = \begin{bmatrix} A_{k-1}^\dagger (I - a_k b_k^\top) \\ b_k^\top \end{bmatrix}$$

where

$$c_k = (I - A_{k-1} A_{k-1}^\dagger) a_k, \quad b_k = \begin{cases} \frac{(A_{k-1}^\dagger)^r A_{k-1}^\dagger a_k}{1 + \|A_{k-1}^\dagger a_k\|^2}, & c_k = 0 \\ \frac{c_k}{\|c_k\|^2}, & c_k \neq 0 \end{cases}$$

C.1 Proof of Proposition 4.2

To save computational cost, we only update the estimator. Suppose the current output matrix is $H_{k+1} = \begin{bmatrix} H_k \\ h_{k+1}^\top \end{bmatrix}$ and the target vector is $t_{k+1} = \begin{bmatrix} t_k \\ t_{k+1} \end{bmatrix}$, then

$$v_{k+1}^\top H_{k+1}^\top = t_{k+1}^\top \Rightarrow v_{k+1}^\top = t_{k+1}^\top (H_{k+1}^\top)^\dagger$$
\[
\begin{align*}
&= \left[ t_k^T t_{k+1} \right] \left( [H_k^T h_{k+1}] \right)^\dagger \\
&= \left[ t_k^T t_{k+1} \right] \begin{bmatrix} (H_k^T)^\dagger (I - h_{k+1} b_{k+1}^T) \\ b_{k+1}^T \end{bmatrix} \\
&= t_k^T (H_k^T)^\dagger (I - h_{k+1} b_{k+1}^T) + t_{k+1} b_{k+1}^T \\
&= v_k^T + (t_{k+1} - v_k^T h_{k+1}) b_{k+1}^T
\end{align*}
\]

where \( b_{k+1} \) is obtained according to Theorem C.1. Note that the computation of \( b_{k+1} \) and \( c_{k+1} \) still involves \( H_k \) and \( H_k^\dagger \) whose sizes increase as data size grows. Following Kohonen (1988); Kovanic (1979); van Schaik and Tapson (2015), we introduce two auxiliary matrices here

\[
\Phi_k = I - H_k^T (H_k^T)^\dagger \in \mathbb{R}^{s \times s}, \quad \Xi_k = (H_k^T)^\dagger (H_k^T)^\dagger = H_k^T (H_k^T)^\dagger \in \mathbb{R}^{s \times s}
\]

and rewrite \( b_{k+1} \) and \( c_{k+1} \) as

\[
c_{k+1} = \Phi_k h_{k+1}, \quad b_{k+1} = \begin{cases} \\
\frac{\Xi_k h_{k+1}}{1 + h_{k+1}^T h_{k+1}}, & c_{k+1} = 0 \\
\frac{c_{k+1}}{\|c_{k+1}\|}, & c_{k+1} \neq 0
\end{cases}
\]

Updating of the two auxiliary matrices can also be done adaptively

\[
\Phi_{k+1} = I - H_{k+1}^T (H_{k+1}^T)^\dagger = I - [H_k^T h_{k+1}] \begin{bmatrix} (H_k^T)^\dagger (I - h_{k+1} b_{k+1}^T) \\ b_{k+1}^T \end{bmatrix} = \Phi_k - \Phi_k h_{k+1} b_{k+1}^T
\]

\[
\Xi_{k+1} = H_{k+1}^T (H_{k+1}^T)^\dagger = (I - b_{k+1} h_{k+1}^T) H_k^T b_{k+1}^T \begin{bmatrix} (H_k^T)^\dagger (I - h_{k+1} b_{k+1}^T) \\ b_{k+1}^T \end{bmatrix} = (I - b_{k+1} h_{k+1}^T) \Xi_k (I - h_{k+1} b_{k+1}^T) + b_{k+1} b_{k+1}^T
\]
and if $c_{k+1} = 0$, these formulas can be simplified as below

$$
\Phi_{k+1} = \Phi_k, \quad \Xi_{k+1} = \Xi_k - \Xi_k h_{k+1} b_{k+1}^T
$$
Appendix D

Variational Hamiltonian Monte Carlo

D.1 Proof of Theorem 5.1

Proof. In order to prove that the equilibrium distribution is the surrogate induced distribution, it suffices to show that the detailed balance condition still holds. Denote \( x = (\theta_0, r_0) \), \( x' = (\theta^*, r^*) \). In the Metropolis-Hasting step, we use the original Hamiltonian to compute the acceptance probability

\[
\alpha(x, x') = \min(1, \exp[-\tilde{H}(x') + \tilde{H}(x)])
\]

therefore,

\[
\alpha(x, x') \mathbb{P}(dx) = \alpha(x, x') \exp[-\tilde{H}(x)] dx
\]

(reversibility) = \min(\exp[-\tilde{H}(x)], \exp[-\tilde{H}(x')]) \left| \frac{dx}{dx'} \right| dx'

= \alpha(x', x) \exp[-\tilde{H}(x')] dx' (volume conservation)

= \alpha(x', x) \mathbb{P}(dx')
Now that we showed the detailed balance condition is satisfied, along with the reversibility 
of the surrogate induced Hamiltonian flow, the modified Markov chain will converge to the 
surrogate induced distribution.

\[ \square \]

D.2 Proof of Theorem 5.3

Proof. Assume \( \bar{J}(v) = 0 \). Then, the assumption \( q_v(\theta) > 0 \) implies that \( \psi_v(.) = \psi_D(.) \). This 
implies \( \log \pi_D(.) = \log q_v(.) + c \) for some constant \( c \). However, \( c \) is necessarily 0 since both 
\( \pi_D \) and \( q_v \) are probability density functions. Thus, \( \pi_D = q_v \). By assumption, only \( v = v^* \) 
satisfies this equality and that indicates \( v = v^* \). So the implication from left to right has 
been proven. The converse is trivial.

\[ \square \]

D.3 Proof of Proposition 5.1

Proof.

\[ v^{(t)} = \arg \min_v \frac{1}{2} \sum_{n=1}^{t} \| \nabla_\theta z(\theta_n) - \nabla_\theta U(\theta_n) \|^2 + \frac{\lambda}{2} \| v \|^2 \]

\[ = \arg \min_v \sum_{n=1}^{t} \| \nabla_\theta z(\theta_n) - \nabla_\theta U(\theta_n) \|^2 + \lambda \| v \|^2 \] \hspace{1cm} (D.1)

As assumed the neural network surrogate takes the form

\[ z(\theta) = \sum_{i=1}^{s} v_i a(w_i \cdot \theta + d_i) + b \]

its derivative is

\[ \nabla_\theta z(\theta) = \sum_{i=1}^{s} v_i a'(w_i \cdot \theta + d_i)w_i = A(\theta)v \]
where $A(\theta) = (A_1(\theta), A_2(\theta), \ldots, A_s(\theta))$ and

$$A_i(\theta) = a'(w_i \cdot \theta + d_i)w_i, \quad i = 1, \ldots, s$$

Denote

$$A^{(t)} = \begin{bmatrix} A(\theta_1) \\ A(\theta_2) \\ \vdots \\ A(\theta_t) \end{bmatrix}, \quad B^{(t)} = \begin{bmatrix} \nabla_{\theta} U(\theta_1) \\ \nabla_{\theta} U(\theta_2) \\ \vdots \\ \nabla_{\theta} U(\theta_t) \end{bmatrix}$$

(D.1) can be simplified and solved as below

$$v^{(t)} = \arg \min_v \| A^{(t)}v - B^{(t)} \|^2 + \lambda \| v \|^2$$

$$= \arg \min_v v^\top (A^{(t)} + \lambda I) v - 2 (B^{(t)})^\top A^{(t)} v$$

$$= (A^{(t)})^\top (A^{(t)} + \lambda I)^{-1} (A^{(t)})^\top B^{(t)}$$

Similarly, given a new data point $(\theta_{t+1}, \nabla_{\theta} U(\theta_{t+1}))$, the new estimator is

$$v^{(t+1)} = ((A^{(t+1)})^\top A^{(t+1)} + \lambda I)^{-1} (A^{(t+1)})^\top B^{(t+1)}$$

where

$$A^{(t+1)} = \begin{bmatrix} A^{(t)} \\ A_{t+1} \end{bmatrix}, \quad B^{(t+1)} = \begin{bmatrix} B^{(t)} \\ B_{t+1} \end{bmatrix}$$

$A_{t+1} = A(\theta_{t+1})$, $B_{t+1} = \nabla_{\theta} U(\theta_{t+1})$. Therefore

$$v^{(t+1)} = ((A^{(t+1)})^\top A^{(t+1)} + \lambda I)^{-1} (A^{(t+1)})^\top B^{(t+1)}$$

$$= ((A^{(t)})^\top A^{(t)} + A_{t+1}^\top A_{t+1} + \lambda I)^{-1}$$

$$\left[ (A^{(t)})^\top B^{(t)} + A_{t+1}^\top B_{t+1} \right]$$

(D.2)
Denote \( C^{(t)} = [(A^{(t)})^T A^{(t)} + \lambda I]^{-1} \), by Sherman-Morrison-Woodbury formula,

\[
C^{(t+1)} = [(A^{(t)})^T A^{(t)} + A_{t+1}^T A_{t+1} + \lambda I]^{-1} \\
= C^{(t)} - C^{(t)} A_{t+1}^T [I + A_{t+1} C^{(t)} A_{t+1}^T]^{-1} A_{t+1} C^{(t)}
\]

substitute into (D.2)

\[
v^{(t+1)} = C^{(t)} \left[ (A^{(t)})^T B^{(t)} + A_{t+1}^T B_{t+1} \right] - \\
C^{(t)} A_{t+1}^T [I + A_{t+1} C^{(t)} A_{t+1}^T]^{-1} A_{t+1} \\
C^{(t)} \left[ (A^{(t)})^T B^{(t)} + A_{t+1}^T B_{t+1} \right] \\
= v^{(t)} + C^{(t)} A_{t+1}^T \left( B_{t+1} - \\
[I + A_{t+1} C^{(t)} A_{t+1}^T]^{-1} A_{t+1} C^{(t)} A_{t+1}^T B_{t+1} \right) \\
- C^{(t)} A_{t+1}^T [I + A_{t+1} C^{(t)} A_{t+1}^T]^{-1} A_{t+1} v^{(t)} \\
= v^{(t)} + W^{(t+1)} (B_{t+1} - A_{t+1} v^{(t)})
\]

where

\[
W^{(t+1)} = C^{(t)} A_{t+1}^T [I + A_{t+1} C^{(t)} A_{t+1}^T]^{-1}
\]

and the updating formula for \( C^{(t+1)} \) can be simplified as

\[
C^{(t+1)} = C^{(t)} - W^{(t+1)} A_{t+1} C^{(t)}
\]

\[\Box\]