Improving Efficiency in Parameter Estimation Using the Hamiltonian Monte Carlo Algorithm

Mohammed Alfaki

University of Bergen

Master Thesis Presentation, June 29, 2008

Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Well–posedness

Hadamard postulates: Given the operator $A : \mathcal{M} \to \mathcal{D}$. The problem of solving for $x \in \mathcal{M}$, given the data $d \in \mathcal{D}$,

$$A(x) = d + \delta$$

is well–posed if:

1. Existence: for each d, $\exists x$, s.t $A(x) = d$
2. uniqueness: if $A(x^{(1)}) = A(x^{(2)}) \Rightarrow x^{(1)} = x^{(2)}$.
3. stability: A^{-1} is continuous.

Otherwise it is ill–posed.

Deterministic Approach to Solve IP

1. Least Square Estimate Find the best model that minimize the missfit, i.e. norm of the residual $d - A(x)$.

$$x_{LS} = \arg\min_{x \in \mathcal{M}} \|d - A(x)\|^2.$$

If the problem is ill–posed, LS produces large and unreasonable models.

2. Regularization Methods impose stability on an ill–posed problem by incorporating prior information.
Tikhonov regularization:

$$x_\alpha = \arg\min_{m \in \mathcal{M}} \|d - A(x)\|^2 + \alpha \|P(x)\|^2, \quad \alpha > 0.$$

The penalty functional P express the prior information.

Deterministic Approach to Solve IP

1. Least Square Estimate Find the best model that minimize the missfit, i.e. norm of the residual $d - A(x)$.

$$x_{LS} = \arg\min_{x \in \mathcal{M}} \|d - A(x)\|^2.$$

   If the problem is ill–posed, LS produces large and unreasonable models.

2. Regularization Methods impose stability on an ill–posed problem by incorporating prior information.
   Tikhonov regularization:

$$x_\alpha = \arg\min_{m \in \mathcal{M}} \|d - A(x)\|^2 + \alpha \|P(x)\|^2, \quad \alpha > 0.$$

   The penalty functional P express the prior information.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
   - Limited number of data.
   - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as "what is $Pr(x_2 \leq 30)$" is not a problem.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
   - Limited number of data.
   - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as "what is $Pr(x_2 \leq 30)$" is not a problem.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
    - Limited number of data.
    - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as "what is $Pr(x_2 \leq 30)$" is not a problem.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
   - Limited number of data.
   - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as "what is $Pr(x_2 \leq 30)$" is not a problem.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
   - Limited number of data.
   - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as "what is $Pr(x_2 \leq 30)$" is not a problem.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
   - Limited number of data.
   - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as "what is $Pr(x_2 \leq 30)$" is not a problem.

Statistical Approach to Solve IP

Why statistical inversion?

1. The general theory obtained when using a probabilistic point view.
   - Limited number of data.
   - Experimental uncertainties.
2. We can add Prior information.

How statistical inversion?

1. We deal with x, d, and δ as random variables.
2. Express the solution as probability distribution.
3. Asking question such as ”what is $Pr(x_2 \leq 30)$” is not a problem.

Bayes' Theorem–Bayesian Inversion

1. The prior pdf $\pi(x)$ express the knowledge about x prior the data d.
2. $L(d|x)$ is the likelihood of the data d assuming the model is known.

Bayes' theorem update the prior belief by the posterior pdf

$$\pi(x|d) = \frac{L(d|x)\,\pi(x)}{\int_{\mathcal{M}} L(d|x)\,\pi(x)\,dx}.$$

Bayes' Theorem–Bayesian Inversion

1. The prior pdf $\pi(x)$ express the knowledge about x prior the data d.
2. $L(d|x)$ is the likelihood of the data d assuming the model is known.

   Bayes' theorem update the prior belief by the posterior pdf

$$\pi(x|d) = \frac{L(d|x)\,\pi(x)}{\int_{\mathcal{M}} L(d|x)\,\pi(x)\,dx}.$$

Derived Parameter

1. Expectation: posterior summary statistics

$$E(x) = \int_{\mathcal{M}} x \, \pi(x|d) \, dx$$

2. The integrals could be intractable, even for low dimensions.
3. An attractive methodology is Monte Carlo rendering

$$E(x) \approx \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

where $x^{(1)}, x^{(2)}, \ldots, x^{N}$ are models generated from $p(x) \approx \pi(x|d)$

Derived Parameter

1. Expectation: posterior summary statistics

$$E(x) = \int_{\mathcal{M}} x\, \pi(x|d)\, dx$$

2. The integrals could be intractable, even for low dimensions.
3. An attractive methodology is Monte Carlo rendering

$$E(x) \approx \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

where $x^{(1)}, x^{(2)}, \ldots, x^{N}$ are models generated from $p(x) \approx \pi(x|d)$

Derived Parameter

1. Expectation: posterior summary statistics

$$E(x) = \int_{\mathcal{M}} x\, \pi(x|d)\, dx$$

2. The integrals could be intractable, even for low dimensions.
3. An attractive methodology is Monte Carlo rendering

$$E(x) \approx \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

where $x^{(1)}, x^{(2)}, \ldots, x^{N}$ are models generated from $p(x) \approx \pi(x|d)$

Outline

1 Introduction
- Inverse Problem – IP
- Markov Chain Monte Carlo Methods
- Hamiltonian Monte Carlo Algorithm

2 Problem Definition
- The Random Sampling of Momentum
- Constant step–Size

3 Improving The HMC algorithm
- Suppress The random Walk
- Use Adaptive step–size in HD (SVHMC)

4 Simulation And Results
- How to Evalute the HMC algorithm
- Comparing The improved HMC Algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - reqiure huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

MCMC Algorithms

MCMC algorithms are based on Markov chains, which generate samples whose distribution approximate a given target distribution. such as $\pi(x|d)$.

1. For MCMC algorithm to be effective, the sample must be uncorrelated and independent
2. Classical MCMC algorithms
   - Gibbs Sampler algorithm
   - Metropolis–Hastings algorithm
3. Drawbacks of the classical MCMC algorithms
   - require huge number of samples
   - the samples usually highly correlated
   - Inefficient in high dimensions
   - Random walk algorithm

Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

The Hamiltonian

Suppose we wish to sample from the distribution $\pi(x)$.

1. Augment each parameter x_i by momentum variable p_i, then the Hamiltonian $\mathcal{H}(x,p)$ is given by

$$\begin{aligned} \mathcal{H}(x,p) &= V(x) + T(p) && \text{where} \\ V(x) &= -\log \pi(x) && \text{Potential energy} \\ T(p) &= \frac{1}{2}p^T p && \text{Kinetic energy} \end{aligned}$$

2. Define the extended target density by

$$\begin{aligned} \pi(x,p) &= \frac{1}{Z}\exp(-\mathcal{H}(x,p)) \\ &= \pi(x)\mathcal{N}(p;0,I) && \text{sparated density} \end{aligned}$$

The Hamiltonian

Suppose we wish to sample from the distribution $\pi(x)$.

1. Augment each parameter x_i by momentum variable p_i, then the Hamiltonian $\mathcal{H}(x,p)$ is given by

$$\begin{aligned} \mathcal{H}(x,p) &= V(x) + T(p) && \text{where} \\ V(x) &= -\log \pi(x) && \text{Potential energy} \\ T(p) &= \frac{1}{2} p^T p && \text{Kinetic energy} \end{aligned}$$

2. Define the extended target density by

$$\begin{aligned} \pi(x,p) &= \frac{1}{Z} \exp(-\mathcal{H}(x,p)) \\ &= \pi(x) \mathcal{N}(p; 0, I) && \text{sparated density} \end{aligned}$$

How HMC Work

- The dynamics of the system represented by (x, p) can be describe by Hamiltonian equations

$$\frac{dx}{d\tau} = +\frac{\partial \mathcal{H}}{\partial p} = p$$
$$\frac{dp}{d\tau} = -\frac{\partial \mathcal{H}}{\partial x} = -\nabla V(x)$$

- The gradient of $V(x)$ determines how the momentum p changes.
- The momentum variable determines where state x goes, this state changes through the time τ.
- The Hamiltonian Dynamics is time-reversible, volume and total energy preserving.

How HMC Work

- The dynamics of the system represented by (x, p) can be describe by Hamiltonian equations

$$\frac{dx}{d\tau} = +\frac{\partial \mathcal{H}}{\partial p} = p$$
$$\frac{dp}{d\tau} = -\frac{\partial \mathcal{H}}{\partial x} = -\nabla V(x)$$

- The gradient of $V(x)$ determines how the momentum p changes.
- The momentum variable determines where state x goes, this state changes through the time τ.
- The Hamiltonian Dynamics is time-reversible, volume and total energy preserving.

How HMC Work

- The dynamics of the system represented by (x, p) can be describe by Hamiltonian equations

$$\frac{dx}{d\tau} = +\frac{\partial \mathcal{H}}{\partial p} = p$$
$$\frac{dp}{d\tau} = -\frac{\partial \mathcal{H}}{\partial x} = -\nabla V(x)$$

- The gradient of $V(x)$ determines how the momentum p changes.
- The momentum variable determines where state x goes, this state changes through the time τ.
- The Hamiltonian Dynamics is time-reversible, volume and total energy preserving.

How HMC Work

- The dynamics of the system represented by (x, p) can be describe by Hamiltonian equations

$$\frac{dx}{d\tau} = +\frac{\partial \mathcal{H}}{\partial p} = p$$
$$\frac{dp}{d\tau} = -\frac{\partial \mathcal{H}}{\partial x} = -\nabla V(x)$$

- The gradient of $V(x)$ determines how the momentum p changes.
- The momentum variable determines where state x goes, this state changes through the time τ.
- The Hamiltonian Dynamics is time-reversible, volume and total energy preserving.

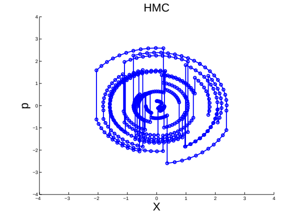
Implementing HMC Algorithm

- In practice the dynamics is simulated using leapfrog scheme.

$$p_{l+\frac{1}{2}} = p_l - \frac{\epsilon}{2}\nabla V(x_l),$$
$$x_{l+1} = x_l + \epsilon p_{l+\frac{1}{2}},$$
$$p_{l+1} = p_{l+\frac{1}{2}} - \frac{\epsilon}{2}\nabla V(x_{l+1}).$$

- In each iteration we make L leapfrog steps.
- Take (x_{L+1}, p_{L+1}) as proposal point.
- The proposal is accepted every time if the simulation almost exactly.

(x,p) Trajectory



A typical trajectories generated by HMC algorithm with 10 leapfrog steps and $\epsilon = 0.23$ from 128-D Gaussian

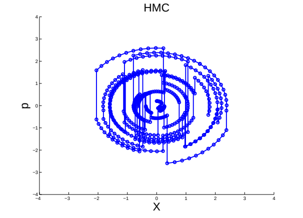
Implementing HMC Algorithm

- In practice the dynamics is simulated using leapfrog scheme.

$$p_{l+\frac{1}{2}} = p_l - \frac{\epsilon}{2}\nabla V(x_l),$$
$$x_{l+1} = x_l + \epsilon p_{l+\frac{1}{2}},$$
$$p_{l+1} = p_{l+\frac{1}{2}} - \frac{\epsilon}{2}\nabla V(x_{l+1}).$$

- In each iteration we make L leapfrog steps.
- Take (x_{L+1}, p_{L+1}) as proposal point.
- The proposal is accepted every time if the simulation almost exactly.

(x,p) Trajectory



A typical trajectories generated by HMC algorithm with 10 leapfrog steps and $\epsilon = 0.23$ from 128-D Gaussian

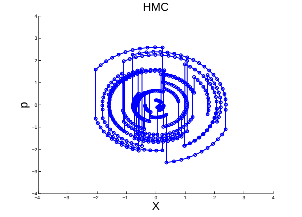
Implementing HMC Algorithm

- In practice the dynamics is simulated using leapfrog scheme.

$$p_{l+\frac{1}{2}} = p_l - \frac{\epsilon}{2}\nabla V(x_l),$$
$$x_{l+1} = x_l + \epsilon p_{l+\frac{1}{2}},$$
$$p_{l+1} = p_{l+\frac{1}{2}} - \frac{\epsilon}{2}\nabla V(x_{l+1}).$$

- In each iteration we make L leapfrog steps.
- Take (x_{L+1}, p_{L+1}) as proposal point.
- The proposal is accepted every time if the simulation almost exactly.

(x,p) Trajectory



A typical trajectories generated by HMC algorithm with 10 leapfrog steps and $\epsilon = 0.23$ from 128-D Gaussian

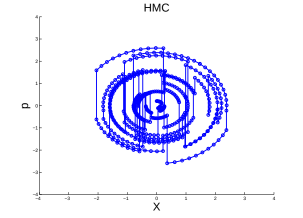
Implementing HMC Algorithm

- In practice the dynamics is simulated using leapfrog scheme.

$$p_{l+\frac{1}{2}} = p_l - \frac{\epsilon}{2}\nabla V(x_l),$$
$$x_{l+1} = x_l + \epsilon p_{l+\frac{1}{2}},$$
$$p_{l+1} = p_{l+\frac{1}{2}} - \frac{\epsilon}{2}\nabla V(x_{l+1}).$$

- In each iteration we make L leapfrog steps.
- Take (x_{L+1}, p_{L+1}) as proposal point.
- The proposal is accepted every time if the simulation almost exactly.

(x,p) Trajectory



A typical trajectories generated by HMC algorithm with 10 leapfrog steps and $\epsilon = 0.23$ from 128-D Gaussian

The Algorithm

Gibbs Sampling Stage

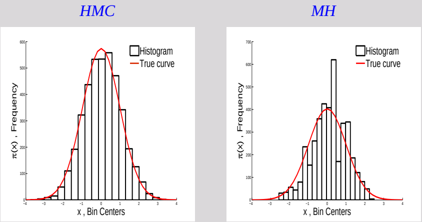
Algorithm

Initialize $x^{(0)}$
for $i = 1$ to $N-1$
    Sample $u \sim U_{[0,1]}$ and $p^* \sim N(0,I)$
    $x_0 = x^{(i)}$ and $p_0 = p^* + \frac{\varepsilon}{2}\nabla V(x_0)$
    For $l = 1$ to L
        $p_l = p_{l-1} - \frac{\varepsilon}{2}\nabla V(x_l)$
        $x_l = x_{l-1} + \varepsilon p_{l-1}$
        $p_l = p_{l-1} - \frac{\varepsilon}{2}\nabla V(x_l)$
    end For
    $dH = H(x_L, p_L) - H(x^{(i)}, p^*)$
    if $u < r = \min\{1, \exp(-dH)\}$
        $(x^{(i+1)}, p^{(i+1)}) = (x_L, p_L)$
    else
        $(x^{(i+1)}, p^{(i+1)}) = (x^{(i)}, p^{(i)})$
end for
return $x = [x^{(1)}, x^{(2)}, \ldots, x^{(N-1)}]$

Simulation of HD Stage

Accept Reject Stage

Example



The contour plots of the 2D correlated Gaussian distribution.

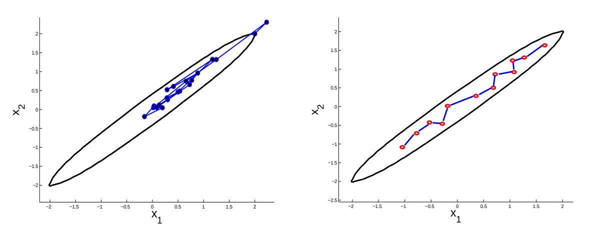
Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Random Walk on the Momentum

Remember: drawing the momentum variable p in HMC according to the Gibbs sampler, which is random walks algorithm.

Illustration



(a) HMC walker (b) desirable walker

Outline

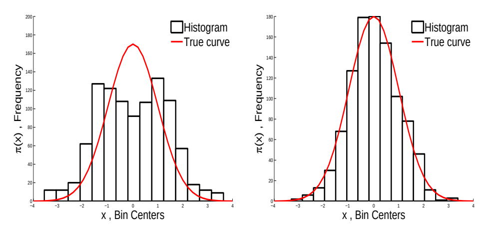
1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Constant step–Size

constant and Adaptive ϵ

- How to choose ϵ and L.
- Constant ϵ leads to extra computational costs.
- Adaptive step-size reduce the simulation error.

Compare



constant $\epsilon = 0.1$, $L = 30$ Adaptive ϵ, $L = 30$

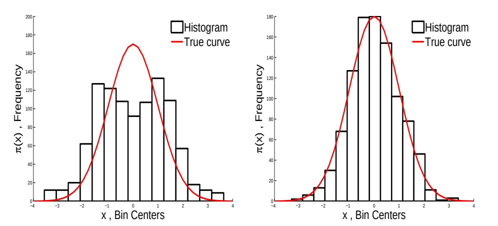
The effect of using the adaptive step-size on the 2-D uncorrelated Gaussian distribution

Constant step–Size

constant and Adaptive ϵ

- How to choose ϵ and L.
- Constant ϵ leads to extra computational costs.
- Adaptive step-size reduce the simulation error.

Compare



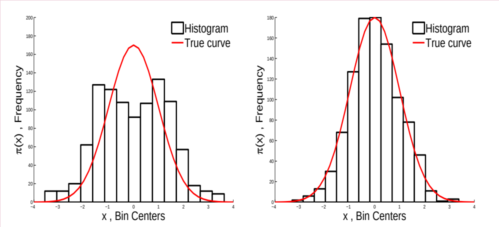
constant $\epsilon = 0.1$, $L = 30$ Adaptive ϵ, $L = 30$
The effect of using the adaptive step-size on the 2-D uncorrelated Gaussian distribution

Constant step–Size

constant and Adaptive ϵ

- How to choose ϵ and L.
- Constant ϵ leads to extra computational costs.
- Adaptive step-size reduce the simulation error.

Compare



constant $\epsilon = 0.1$, $L = 30$ Adaptive ϵ, $L = 30$

The effect of using the adaptive step-size on the 2-D uncorrelated Gaussian distribution

Outline

1 Introduction
- Inverse Problem – IP
- Markov Chain Monte Carlo Methods
- Hamiltonian Monte Carlo Algorithm

2 Problem Definition
- The Random Sampling of Momentum
- Constant step–Size

3 Improving The HMC algorithm
- Suppress The random Walk
- Use Adaptive step–size in HD (SVHMC)

4 Simulation And Results
- How to Evalute the HMC algorithm
- Comparing The improved HMC Algorithm

Applying over–relaxation on p (OHMC)

Orderd over-relaxation

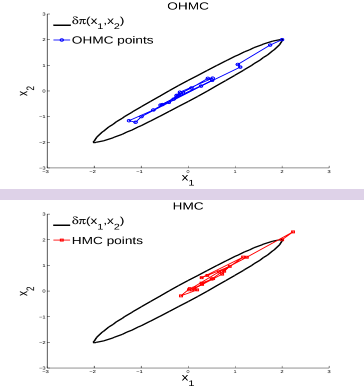
To over–relax $\mathbb{R}^n \ni p \sim \mathcal{N}(p; 0I)$

For $i = 1 : n$

- Generated K values from $\mathcal{N}(q_i | \{q_j\}_{i \neq j})$.
- Order these K values and the odd value p_i. $q_i^{(0)} \leq \cdots \leq q_i^{(r)} = p_i \leq \cdots \leq q_i^{(K)}$
- Set $p_i' = q_i^{(K-r)}$.

End for

Example



Applying over–relaxation on p (OHMC)

Orderd over-relaxation

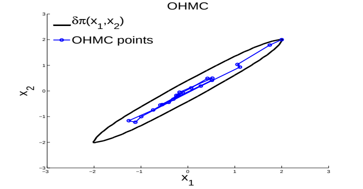
To over–relax $\mathbb{R}^n \ni p \sim \mathcal{N}(p; 0I)$

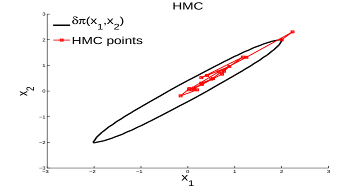
For $i = 1 : n$

- Generated K values from
  $$\mathcal{N}(q_i | \{q_j\}_{i \neq j}).$$
- Order these K values and the odd value p_i.
  $$q_i^{(0)} \leq \cdots \leq q_i^{(r)} = p_i \leq \cdots \leq q_i^{(K)}$$
- Set $p_i' = q_i^{(K-r)}$.

End for

Example





Applying over–relaxation on p (OHMC)

Orderd over-relaxation

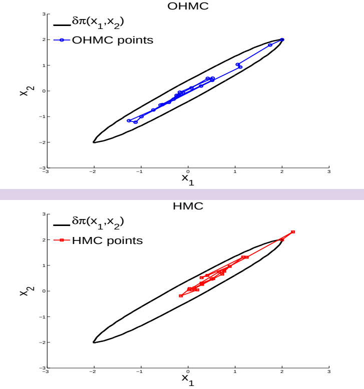
To over–relax $\mathbb{R}^n \ni p \sim \mathcal{N}(p; 0I)$

For $i = 1 : n$

- Generated K values from
$$\mathcal{N}(q_i | \{q_j\}_{i \neq j}).$$
- Order these K values and the odd value p_i.
$$q_i^{(0)} \leq \cdots \leq q_i^{(r)} = p_i \leq \cdots \leq q_i^{(K)}$$
- Set $p_i' = q_i^{(K-r)}$.

End for

Example



Outline

1. Introduction
   - Inverse Problem – IP
   - Markov Chain Monte Carlo Methods
   - Hamiltonian Monte Carlo Algorithm
2. Problem Definition
   - The Random Sampling of Momentum
   - Constant step–Size
3. Improving The HMC algorithm
   - Suppress The random Walk
   - Use Adaptive step–size in HD (SVHMC)
4. Simulation And Results
   - How to Evalute the HMC algorithm
   - Comparing The improved HMC Algorithm

Adaptive Step–size

Adaptive Störmer–Verlet

Simulate the dynamics with adaptive Störmer–Verlet scheme instead of the leapfrog scheme
For $l = 1 : L - steps$

$$\begin{aligned}
x_{l+\frac{1}{2}} &= x_l + \frac{\epsilon}{2\rho_l} p_{l+\frac{1}{2}}, \\
p_{l+\frac{1}{2}} &= p_l - \frac{\epsilon}{2\rho_l} \nabla V(x_l), \\
\rho_{l+1} + \rho_l &= 2U(x_{l+\frac{1}{2}}, p_{l+\frac{1}{2}}), \\
p_{l+1} &= p_{l+\frac{1}{2}} - \frac{\epsilon}{2\rho_{l+1}} \nabla V(x_{l+1}), \\
x_{l+1} &= x_{l+\frac{1}{2}} + \frac{\epsilon}{2\rho_{n+1}} p_{l+\frac{1}{2}}.
\end{aligned}$$

End For

Adaptive Step–size

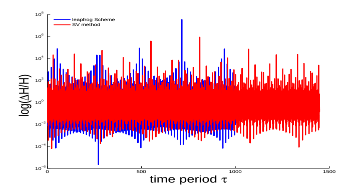
Adaptive Störmer–Verlet

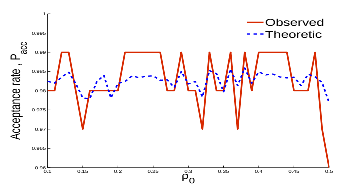
- Adaptive ϵ reduces ΔH.
- The ϵ depend on

$$U(x,p) = \sqrt{\|\nabla V(x)\|^2 + p^T[\nabla^2 V(x)]^2 p}$$

- The accepted rate agree with theory

Example





Summary

combine these to give (OSVHMC)

Algorithm–proposal change

Initialize $x^{(0)}$
for $i = 1$ to $N-1$
  Sample $u \sim U_{[0,1]}$ and $p^* \sim N(0, I)$
  $x_0 = x^{(i)}$ and $p_0 = p^* + \frac{\varepsilon}{2}\nabla V(x_0)$
  For $l = 1$ to L
    $p_l = p_{l-1} - \frac{\varepsilon}{2}\nabla V(x_l)$
    $x_l = x_{l-1} + \varepsilon p_{l-1}$
    $p_l = p_{l-1} - \frac{\varepsilon}{2}\nabla V(x_l)$
  end For
  $dH = H(x_L, p_L) - H(x^{(i)}, p^*)$
  if $u < r = \min\{1, \exp(-dH)\}$
    $(x^{(i+1)}, p^{(i+1)}) = (x_L, p_L)$
  else
    $(x^{(i+1)}, p^{(i+1)}) = (x^{(i)}, p^{(i)})$
end for
return $x = [x^{(1)}, x^{(2)}, \ldots, x^{(N-1)}]$

over-relax the Gibbs sampling (OHMC)

replace leapfrog by adaptive Stormer-Verlet (SVHMC)

Outline

1 Introduction
- Inverse Problem – IP
- Markov Chain Monte Carlo Methods
- Hamiltonian Monte Carlo Algorithm

2 Problem Definition
- The Random Sampling of Momentum
- Constant step–Size

3 Improving The HMC algorithm
- Suppress The random Walk
- Use Adaptive step–size in HD (SVHMC)

4 Simulation And Results
- How to Evalute the HMC algorithm
- Comparing The improved HMC Algorithm

The Evaluating Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\hat{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^{\alpha}}{(\kappa^*/\kappa)^{\alpha}+1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N\to\infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluating Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N \to \infty} \frac{\sigma_0^2 / N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluting Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N\to\infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluting Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N \to \infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluting Criteria

Suppose we have a chain $\{x_i\}_{i=1}^N$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N \to \infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluting Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N \to \infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluting Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N \to \infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

The Evaluating Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^\alpha}{(\kappa^*/\kappa)^\alpha + 1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N \to \infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

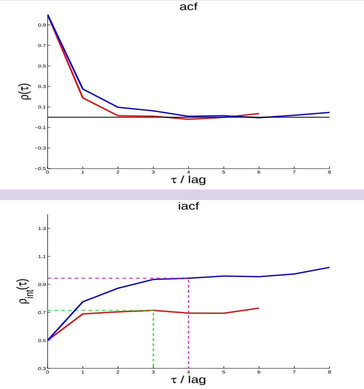
The Evaluting Criteria

Suppose we have a chain $\{x_i\}_{i=1}^{N}$ generated by any MCMC algorithm.

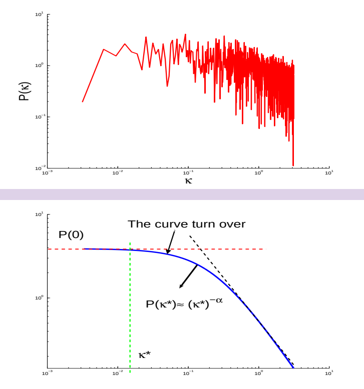
1. Degree of correlation criteria
   - Autocorrelation function $\rho(l) = \frac{Cov(x_i, x_{i+l})}{Var(x_i)}$.
   - Integrated autocorrelation time $\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \rho(t)$.
   - Number effective samples $N_{eff} = N/(2\tau_{int})$.
2. Spectral analysis criteria
   - compute $\tilde{P}_j = |\tilde{X}(\kappa)^* \tilde{X}(\kappa)|$, then estimate α, $P(0)$ and κ^* from the template $P(\kappa) = P_0 \frac{(\kappa^*/\kappa)^{\alpha}}{(\kappa^*/\kappa)^{\alpha}+1}$.
   - The sample mean variance $\sigma_{\bar{x}}^2 \approx \frac{1}{N} P(\kappa = 0)$.
   - Convergence ratio $r = \frac{\sigma_{\bar{x}}^2}{\sigma_0^2}$.
   - The Efficiency $E = \lim_{N\to\infty} \frac{\sigma_0^2/N}{\sigma_{\bar{x}}^2(N)}$.

Evaluting criteria/Geometric Illustration

Degree of correlation



Spectral analysis



Outline

1 Introduction
- Inverse Problem – IP
- Markov Chain Monte Carlo Methods
- Hamiltonian Monte Carlo Algorithm

2 Problem Definition
- The Random Sampling of Momentum
- Constant step–Size

3 Improving The HMC algorithm
- Suppress The random Walk
- Use Adaptive step–size in HD (SVHMC)

4 Simulation And Results
- How to Evalute the HMC algorithm
- Comparing The improved HMC Algorithm

Comparing OHMC vs HMC

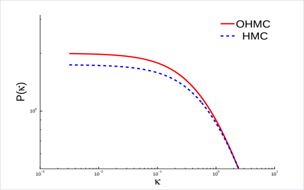
Gaussian Target

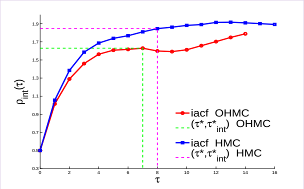
$$\pi(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}x^T\Sigma^{-1}x\right)$$
$$\Sigma = I$$

Results (n=64, N=2000)

	OHMC	HMC	Ideal
Accept. Rate	0.99	0.99	1
$P(0)$	3.10	3.56	1
κ^*	0.99	0.81	
cpu time/sec	561.22	557.38	
E	0.32	0.28	1
r	0.001	0.002	< 0.01
τ_{int}	1.63	1.85	0.5
N_{eff}	614	542	2000

Geometric Illustration



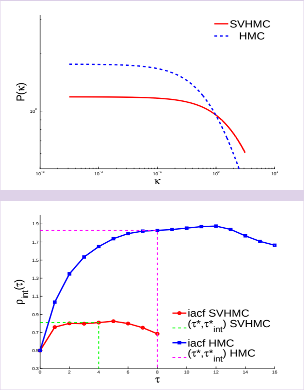


Comparing SVHMC vs HMC

Numerical Results (n=128 N=2000)

	SVHMC	HMC	Ideal
Accept. Rate	0.92	0.98	1
$P(0)$	2.11	3.13	1
κ^*	3.27	1.15	
cpu time/sec	1568.01	1117.78	
E	0.47	0.32	1
r	0.001	0.002	< 0.01
τ_{int}	0.86	1.80	0.5
N_{eff}	1167	554	2000

Geometric Illustration

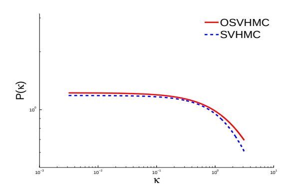


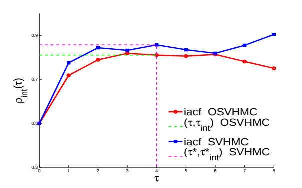
Comparing OSVHMC vs SVHMC

Numerical Results (n=64, N=2000)

	OSVHMC	SVHMC	Ideal
Accept. Rate	0.92	0.94	1
$P(0)$	1.90	2.26	1
κ^*	4.89	3.14	
cpu time/sec	639.58	669.40	
E	0.53	0.44	1
r	0.0009	0.0011	< 0.01
τ_{int}	0.71	0.94	0.5
N_{eff}	1400	1059	2000

Geometric Illustration





Summary and Conclusion

1. OHMC algorithm improves the number of the effective sample by factor of $\sim 12\%$.

$$\frac{E_{SVHMC}}{E_{HMC}} \approx 1.2$$

.

2. SVHMC algorithm outperform the classical HMC algorithm by having $\sim 50\%$ more effective sample

$$\frac{N_{eff}}{N_{eff}} \approx 0.5$$

3. Finally, the hybrid OSVHMC (relaxing the momentum and using Adaptive ϵ) outperform the SVHMC.

Summary and Conclusion

1. OHMC algorithm improves the number of the effective sample by factor of $\sim 12\%$.

$$\frac{E_{SVHMC}}{E_{HMC}} \approx 1.2$$

.

2. SVHMC algorithm outperform the classical HMC algorithm by having $\sim 50\%$ more effective sample

$$\frac{N_{eff}}{N_{eff}} \approx 0.5$$

3. Finally, the hybrid OSVHMC (relaxing the momentum and using Adaptive ϵ) outperform the SVHMC.

Summary and Conclusion

1. OHMC algorithm improves the number of the effective sample by factor of $\sim 12\%$.

$$\frac{E_{SVHMC}}{E_{HMC}} \approx 1.2$$

.

2. SVHMC algorithm outperform the classical HMC algorithm by having $\sim 50\%$ more effective sample

$$\frac{N_{eff}}{N_{eff}} \approx 0.5$$

3. Finally, the hybrid OSVHMC (relaxing the momentum and using Adaptive ϵ) outperform the SVHMC.

Finally

Thank You