Hamiltonian Monte Carlo and NUTS

Bayesian Methodology in Biostatistics (BST 249)

Jeffrey W. Miller

Department of Biostatistics
Harvard T.H. Chan School of Public Health

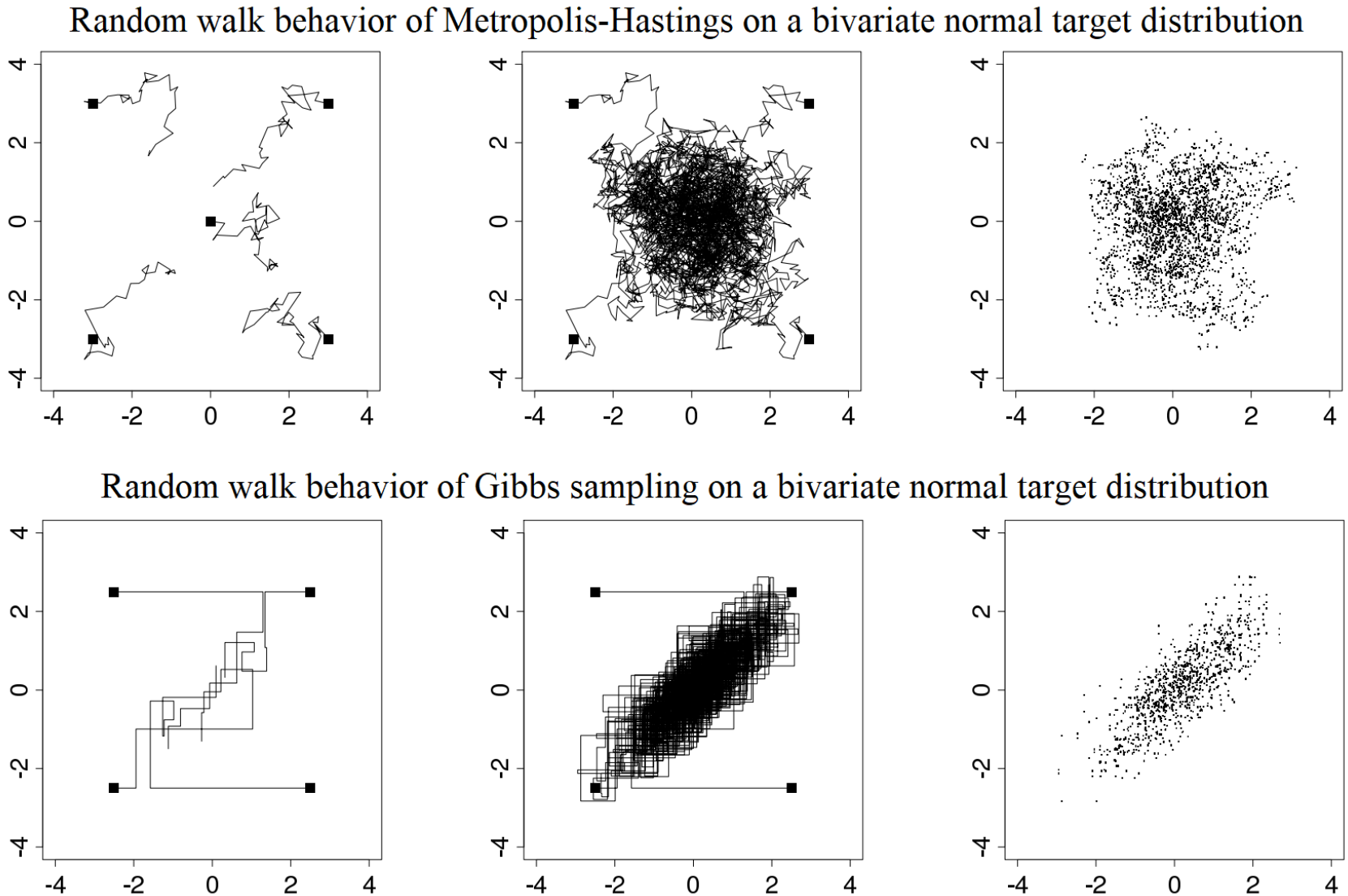
Outline

Outline

Hamiltonian Monte Carlo: Introduction

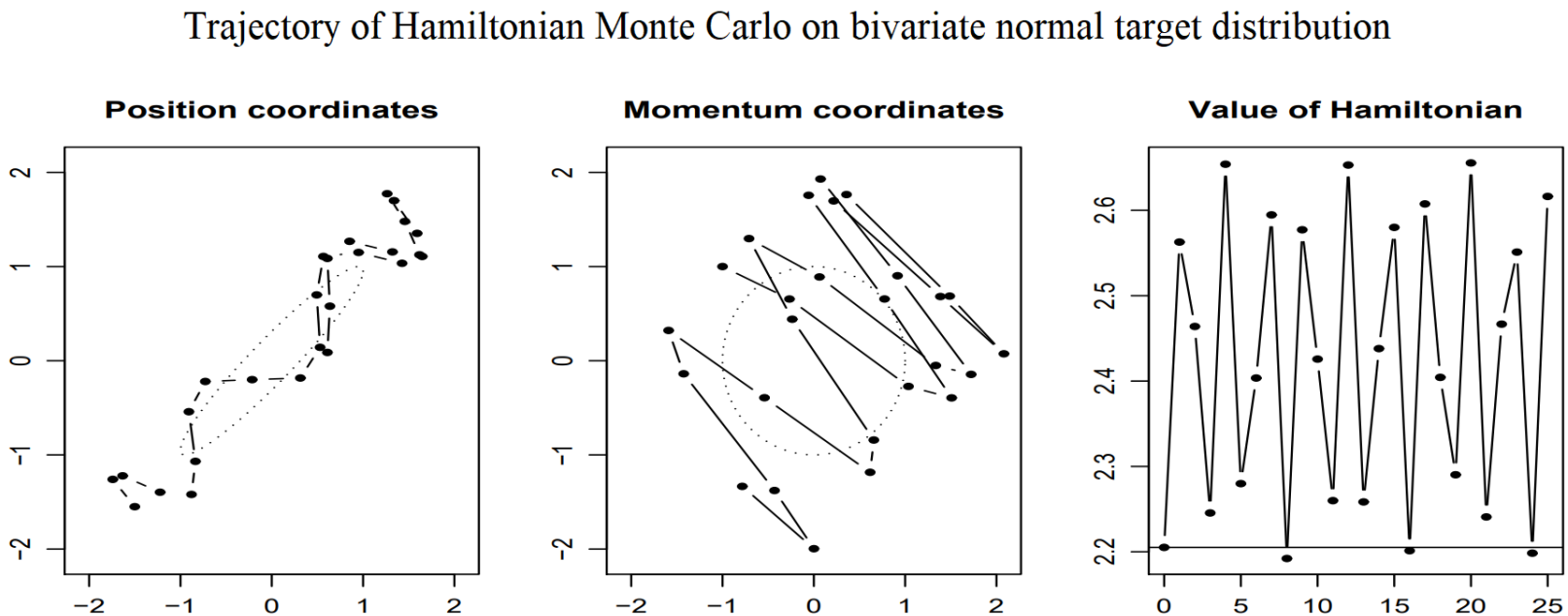
- Recall that reducing the correlation between successive states is key to improving the accuracy of MCMC approximations.
- Many MCMC samplers tend to exhibit so-called "random walk" behavior, roughly meaning that they meander to and fro as they sample from the target distribution.
- Using well-chosen transformations and large moves can improve mixing performance, but often they are hard to construct for complex distributions on high-dimensional spaces.
- Hamiltonian Monte Carlo (HMC), also referred to as Hybrid Monte Carlo, employs a dynamical systems approach to more quickly traverse the space and thus improve MCMC mixing.

Hamiltonian Monte Carlo: Introduction



(figure from Gelman et al. (2013), BDA3, Chapter 11)

Hamiltonian Monte Carlo: Introduction



(figure from Neal (2011))

Hamiltonian Monte Carlo: Introduction

HMC Harlem Shake:
`https://www.youtube.com/watch?v=Vv3f0QNWvWQ`

Hamiltonian Monte Carlo: Introduction

Tutorial with nice demos of Hamiltonian Monte Carlo:
`http://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html`

Hamiltonian Monte Carlo: Basic idea

- Goal: Sample from target density $\pi(x)$, where $x \in \mathbb{R}^d$ is a continuous variable.

- Assume we can compute the gradient of the log density, $\nabla \log \pi(x)$. Analogously to gradient-based optimization methods, HMC uses gradients to improve MCMC mixing.

- Basic idea:
  1. Sample an auxiliary variable $z \in \mathbb{R}^d$ where $z_i|x \sim \mathcal{N}(0, m_i)$ independently for $i = 1, \ldots, d$.
  2. Jointly transform (x, z) in a way that leaves $p(x, z)$ roughly constant by using Hamiltonian dynamics.
  3. Use a Metropolis–Hastings step to accept or reject the transformed (x, z).

Outline

Hamiltonian dynamics: Physical interpretation

- The transformation of (x, z) is done by running a dynamical system with Hamiltonian $H(x, z)$ forward in time, where

$$H(x, z) := -\log \pi(x) + \frac{1}{2} \sum_{i=1}^{d} z_i^2 / m_i.$$

- Note that $p(x, z) \propto \exp(-H(x, z))$.

- Later, we will see the step-by-step algorithm, but abstractly, the idea is to run the dynamical system

$$\frac{\partial x_i}{\partial t} := \frac{\partial H}{\partial z_i} = z_i / m_i$$
$$\frac{\partial z_i}{\partial t} := -\frac{\partial H}{\partial x_i} = \frac{\partial}{\partial x_i} \log \pi(x)$$

  where t is time.

Hamiltonian dynamics: Physical interpretation

- Intuition: x moves like a ball rolling on the surface $-\log \pi(x)$.

- Physical interpretation:

$$\begin{aligned} x_1, \ldots, x_d &= \text{ position coordinates} \\ z_1, \ldots, z_d &= \text{ momentum coordinates } \quad (z_i = m_i v_i) \\ -\log \pi(x) &= \text{ potential energy} \\ \tfrac{1}{2} \textstyle\sum_{i=1}^{d} z_i^2 / m_i &= \text{ kinetic energy } \quad = \textstyle\sum_i \tfrac{1}{2} m_i v_i^2. \end{aligned}$$

- The Hamiltonian represents the total energy of the system:

$$H = \text{Total energy} = \text{Potential energy} + \text{Kinetic energy}.$$

- By conservation of energy, H remains constant as the dynamical system evolves over time.

- Thus, $p(x, z) \propto \exp(-H(x, z))$ also remains constant as (x, z) evolves according to the dynamical system.

Hamiltonian dynamics: Intuition

- In vector form, the equations defining the dynamics are:

$$\text{velocity} = \frac{\partial}{\partial t}\text{position} = \frac{\partial x}{\partial t} = M^{-1}z,$$

$$\text{force} = \text{mass} \times \text{acceleration} = \frac{\partial}{\partial t}\text{momentum} = \frac{\partial z}{\partial t} = \nabla \log \pi(x)$$

where $M = \text{diag}(m_1, \ldots, m_d)$.

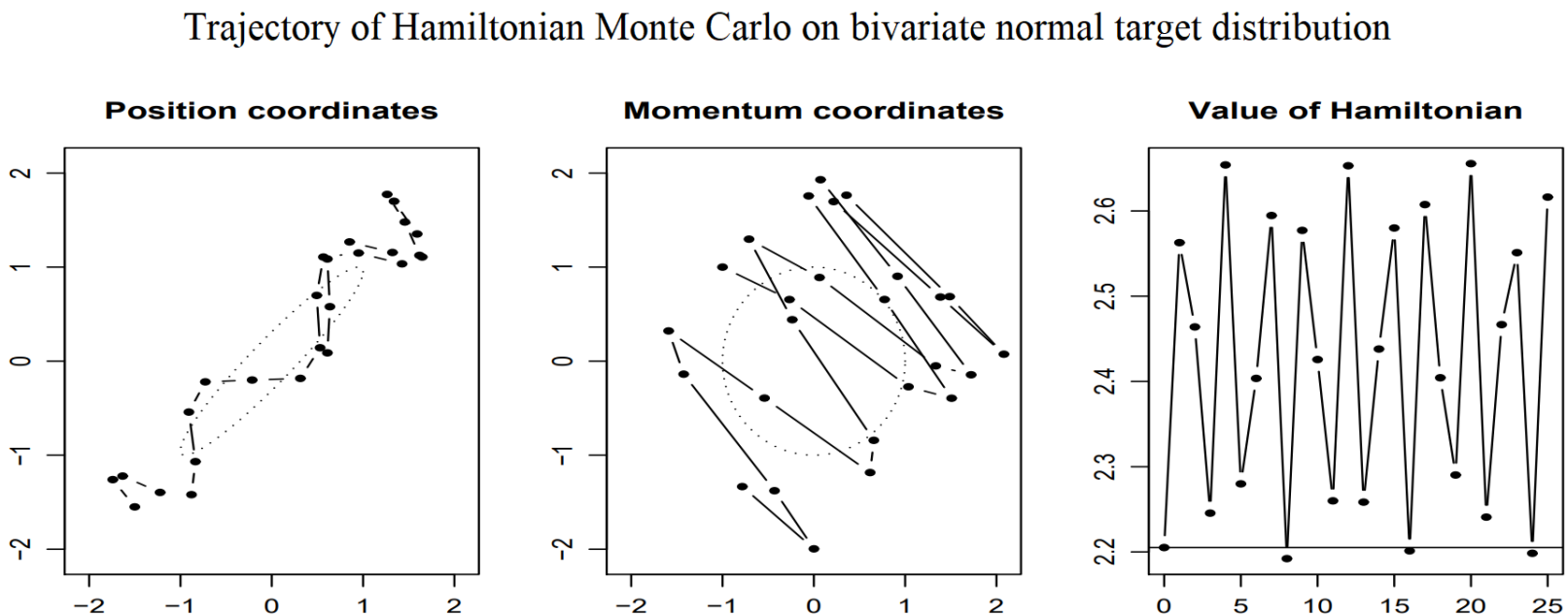
- To gain some intuition for how the system evolves, first suppose $\pi(x)$ is flat in some region. Then $\nabla \log \pi(x) = 0$, so there is zero acceleration and consequently, x will move at constant velocity through this region.

- Meanwhile, if $\pi(x)$ is not flat, then force $= \nabla \log \pi(x)$ means that x is accelerating in the direction of the gradient, i.e., it is accelerating towards a region of higher density.

Hamiltonian dynamics: Discretization

- In practice, it is necessary to discretize. Consequently, the Hamiltonian (total energy) is not exactly constant over time.

- HMC uses "leapfrog" discretization, which allows the Hamiltonian to oscillate somewhat but prevents it from drifting significantly over time.



(figure from Neal (2011))

Outline

Hamiltonian Monte Carlo: Algorithm

An HMC move on x consists of the following steps.

1. Sample $z \sim \mathcal{N}(0, M)$ where $M = \mathrm{diag}(m_1, \ldots, m_d)$.
2. $x_0 \leftarrow x$ and $z_0 \leftarrow z$.
3. For $\ell = 1, \ldots, L$,
   (a) $z \leftarrow z + \frac{1}{2}\varepsilon \nabla \log \pi(x)$

   (b) $x \leftarrow x + \varepsilon M^{-1} z$

   (c) $z \leftarrow z + \frac{1}{2}\varepsilon \nabla \log \pi(x)$

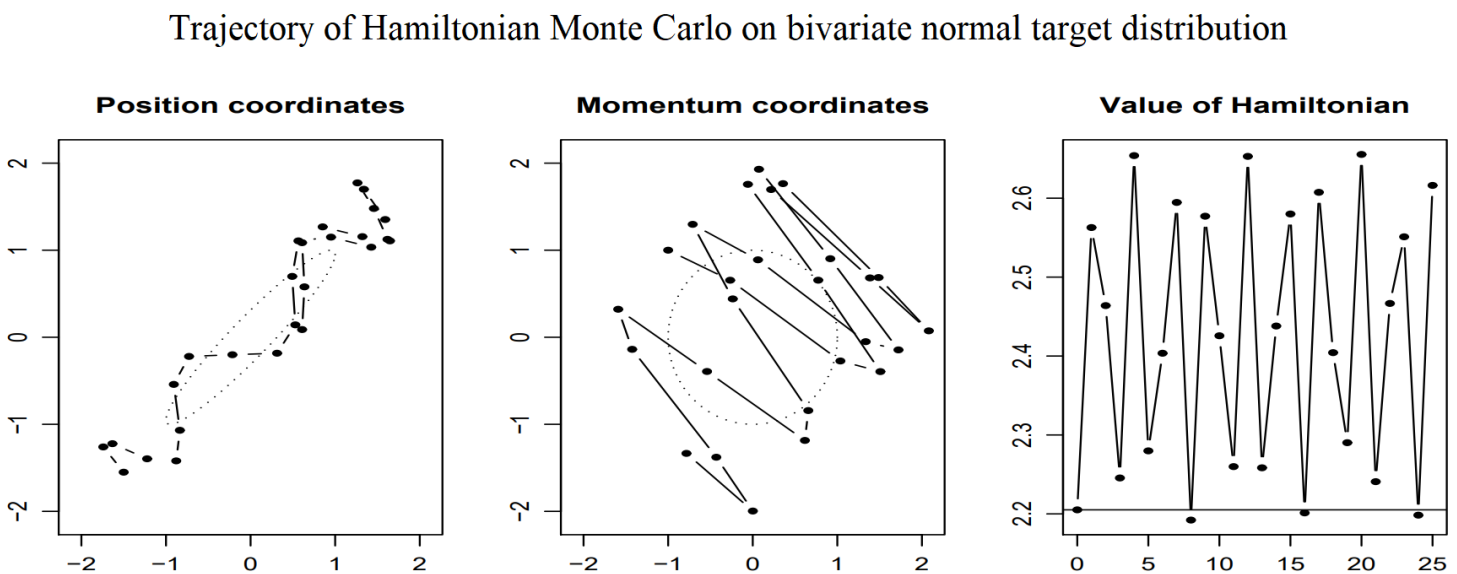
4. Metropolis accept/reject step:
   (a) $\alpha \leftarrow \min\left\{1, \dfrac{\pi(x)\,\mathcal{N}(z \mid 0, M)}{\pi(x_0)\,\mathcal{N}(z_0 \mid 0, M)}\right\}$

   (b) With probability α, accept x as the new state, otherwise reject and keep the state as x_0.

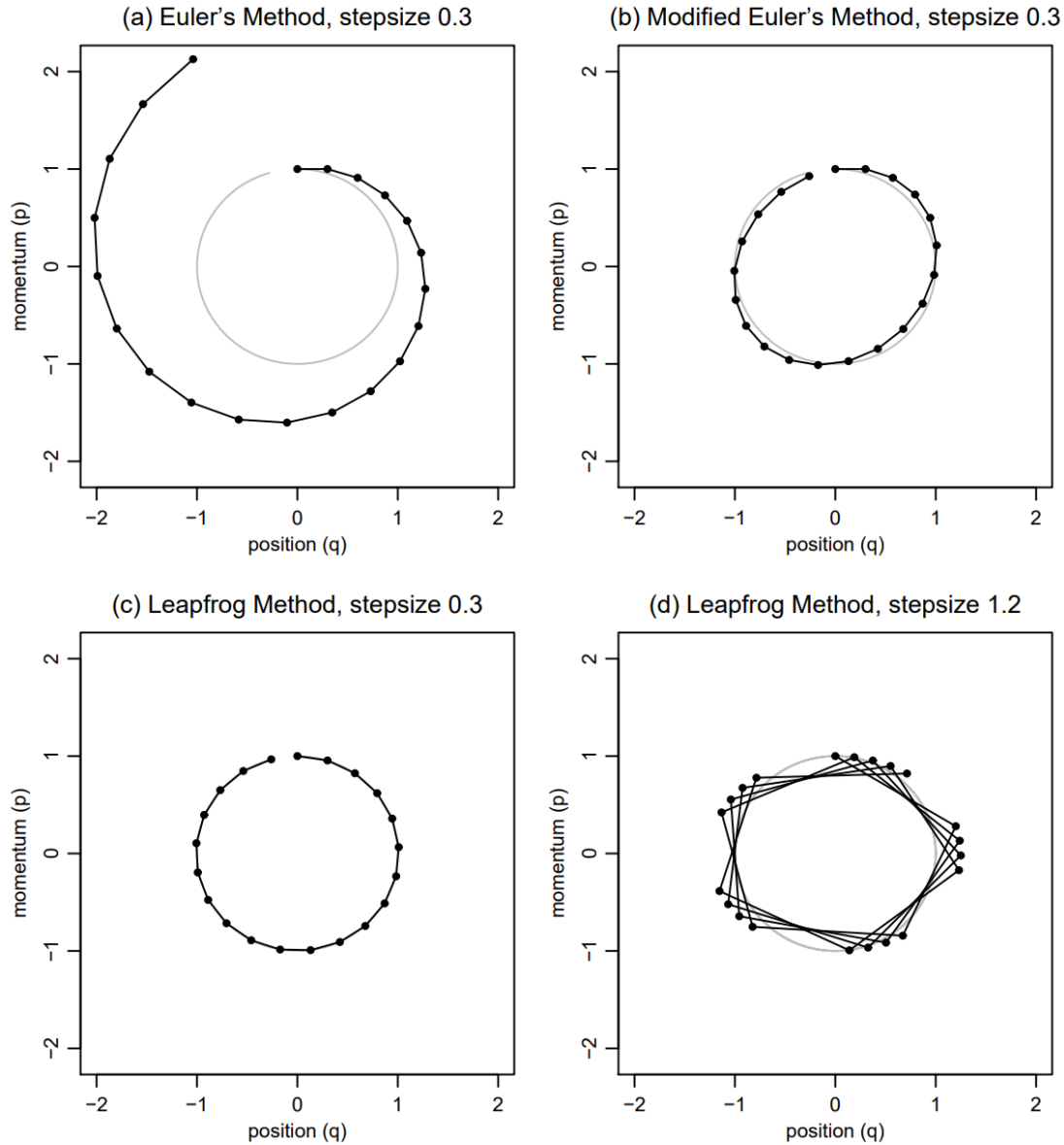
Note that $\pi(x)$ can be replaced by $\tilde{\pi}(x) \propto \pi(x)$.

Hamiltonian Monte Carlo: Leapfrog discretization

- Step 2 numerically propagates the dynamical system using a discrete approximation. Normally, the error of a discrete approximation would grow as the system is propagated.
- However, the "leapfrog" discretization in 2(a)-(c) has the special property that $p(x, z)$ does not significantly change as ℓ goes from 1 to L.
- This helps regulate the Metropolis acceptance probability.



Hamiltonian dynamics: Leapfrog discretization



(figure from Neal (2011))

Hamiltonian Monte Carlo: Leapfrog discretization

- Note that steps 2(a) and 2(c) are identical updates to z.
- 2(a) and 2(c) are "half-step" updates to z that sandwich the full-step update to x.
- Thus, steps 2(c) and 2(a) for each successive pair $(\ell, \ell + 1)$ can be combined into a single full step $z \leftarrow z + \varepsilon \nabla \log \pi(x)$.
- The accuracy of the discrete approximation improves as ε goes to 0, but the number of steps L needs to increase proportionally in order to move the same distance.
- For instance, it is natural to choose ε and L such that $\varepsilon L = 1$.

HMC: Handling regions of zero probability

- HMC is designed for strictly positive target densities $\pi(x) > 0$. If the algorithm lands in a region where $\pi(x) = 0$, then a few common strategies can be applied.

- One approach is to reject if $\pi(x) = 0$ during step 2, and set the state back to x_0 for another iteration.

- An alternative is to "bounce": If $\pi(x) = 0$ during step 2, change the sign of the momentum $z \leftarrow -z$ in order to go in the opposite direction.

- A third approach is to transform x so that the target density is positive on all of $\mathbb{R}^d$. This is attractive if the transformation and its Jacobian can be worked out analytically.

Outline

HMC: Setting the tuning parameters

- The algorithm has several tuning parameters:
  1. $M = \mathrm{diag}(m_1, \ldots, m_d)$, the covariance matrix of z,
  2. $\varepsilon > 0$, the discretization step size, and
  3. L, the number of leapfrog steps.
- These settings can be fixed at the start of the algorithm or chosen adaptively.

HMC: Setting the tuning parameters

- The following guidelines can be used to choose M, ε, and L.
  - Set M to approximate the inverse of the covariance of the target distribution. (Or, as a default, just use $M = I$.)
  - Find a value of ε that yields good mixing. (Or, a reasonable default is $\varepsilon = 0.1$.)
  - Set $L = 1/\varepsilon$ as a default.
- Roughly speaking, these choices calibrate the algorithm to the scale of the target distribution:
  - Suppose Σ is the covariance of the target and $M^{-1} \approx \Sigma$.
  - Since $z \sim \mathcal{N}(0, M)$ initially, we have $M^{-1}z \sim \mathcal{N}(0, M^{-1})$.
  - Recall that we take L steps of the form $x \leftarrow x + \varepsilon M^{-1}z$.
  - Thus, if z was the same throughout the trajectory, then the full trajectory would be $L\varepsilon M^{-1}z \sim \mathcal{N}(0, M^{-1}) \approx \mathcal{N}(0, \Sigma)$.

HMC: Setting the tuning parameters

- As a rough guide, theory suggests adjusting ε so that the acceptance rate (in step 3 of the algorithm) is around 65%.

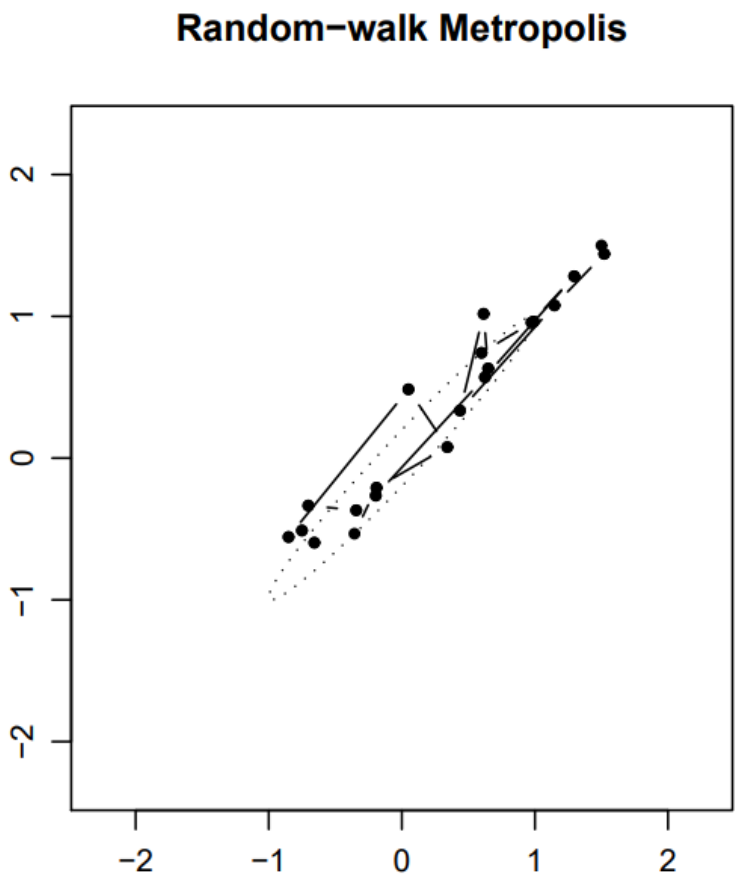
- As usual, MCMC tuning parameters can be adapted during the burn-in period, but adapting throughout the MCMC run can cause it to fail to converge to the target distribution.

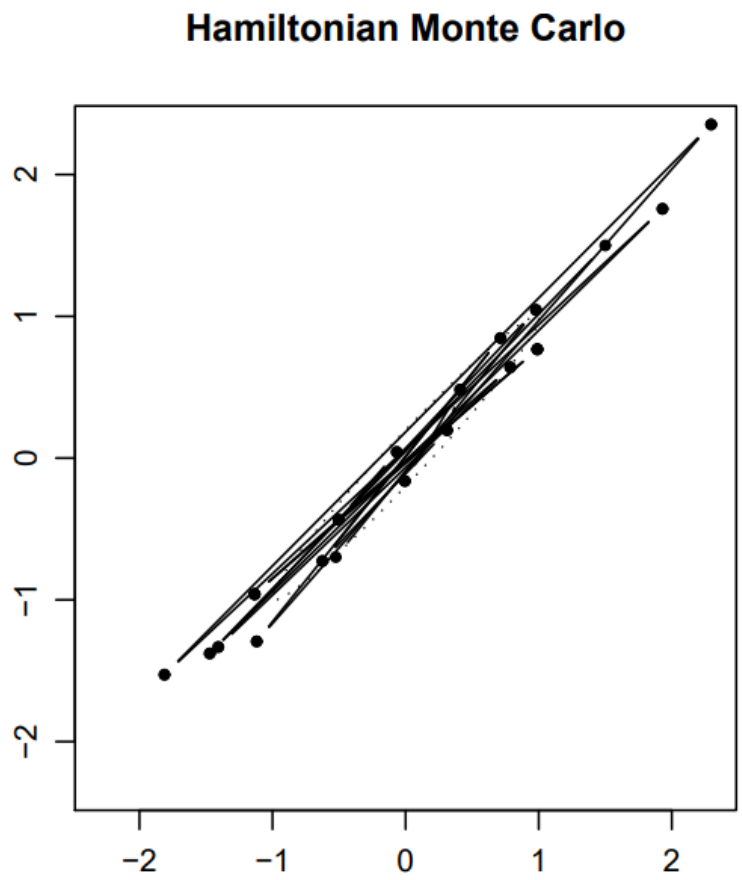
- The "no U-turn sampler" (NUTS) is a method for adapting HMC to the local properties of the target distribution throughout the MCMC run.

- Also, randomly perturbing the tuning parameters (within a reasonable range) can help prevent the algorithm from getting stuck. Random perturbations that do not depend on the MCMC state can be done throughout the MCMC run without compromising the guarantee of convergence.

Outline

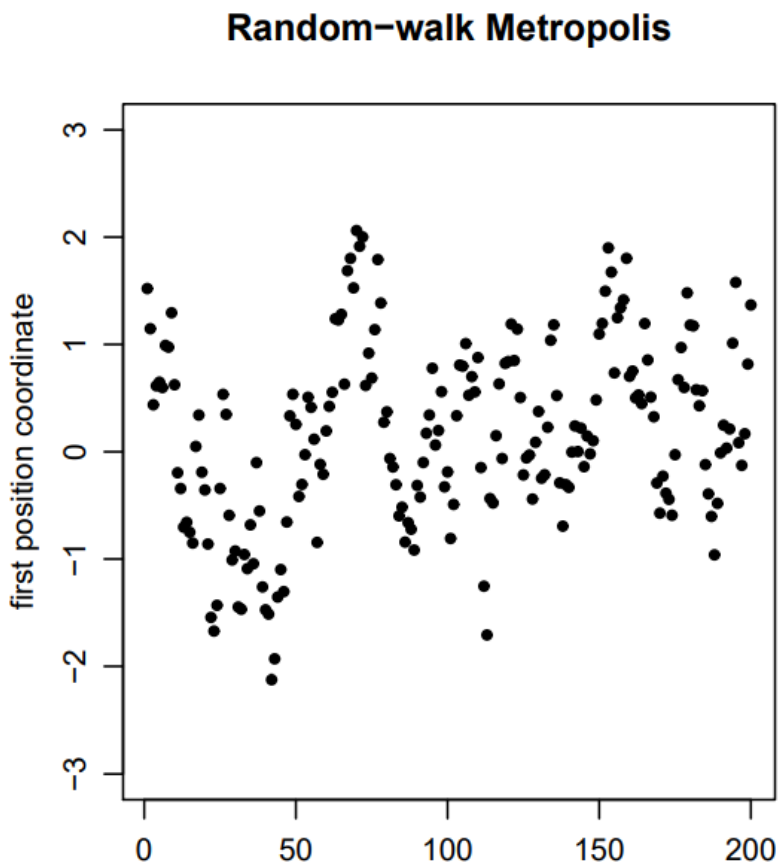
HMC: Bivariate Gaussian example

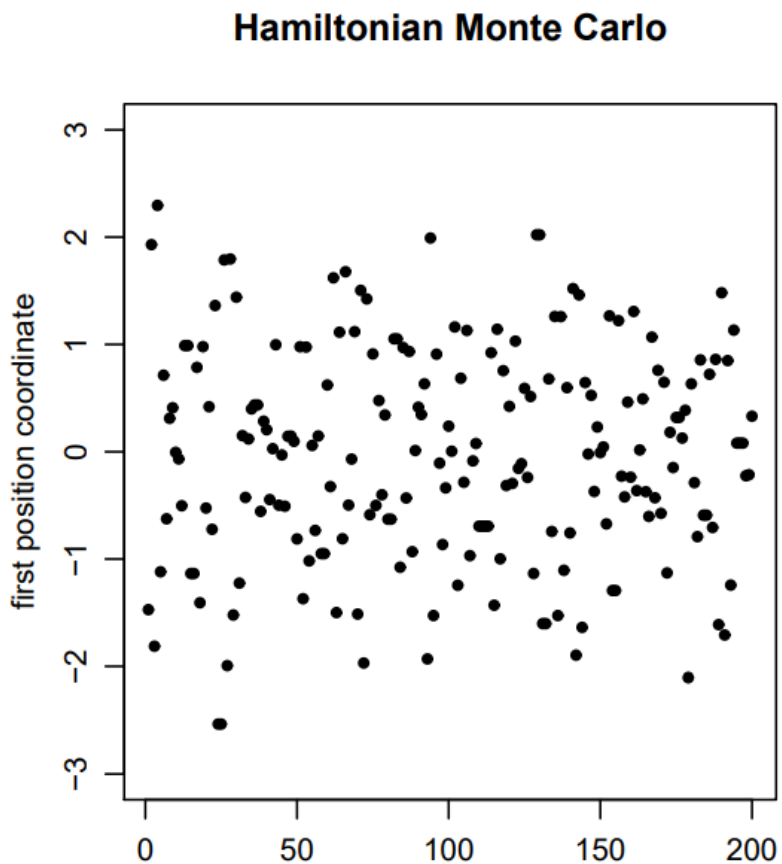




(figure from Neal (2011))

HMC: Bivariate Gaussian example





(figure from Neal (2011))

HMC: 100-dim Gaussian example

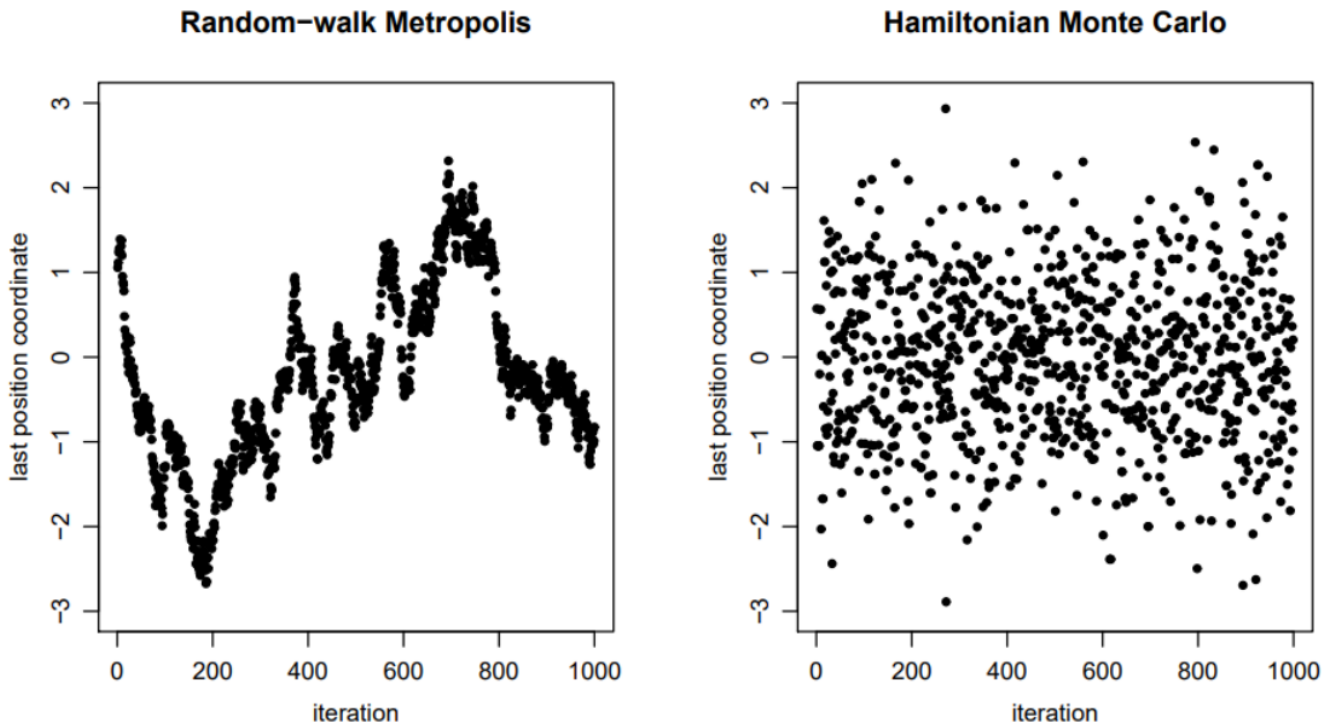


Figure 6: Values for the variable with largest standard deviation for the 100-dimensional example, from a random-walk Metropolis run and an HMC run with $L = 150$. To match computation time, 150 updates were counted as one iteration for random-walk Metropolis.

(figure from Neal (2011))

Outline

Adapting MCMC over time

- Mixing can often be improved by tuning the MCMC settings (such as the proposal distribution) based on lookng at the MCMC samples themselves.
- For instance, one could select the MH proposal distribution to match the estimated covariance of the target distribution.
- Or, one could adjust the size of proposals to attain a desired MH acceptance rate.
- It is always valid to adapt the MCMC settings based on a trial run or the burn-in period.
- However, if you want to continue adapting throughout the MCMC run, you need to be very careful, since otherwise the chain may fail to converge to the target distribution.

No U-turn sampler (NUTS): Introduction

- HMC's performance depends strongly on the tuning parameters M (momentum covariance), ε (step size), and L (number of steps per iteration).
- NUTS is an extension of HMC that adaptively tunes M and ε during burn-in, and adapts L throughout the MCMC run.
- NUTS eliminates the need to select the tuning paramters.
- Empirically, the mixing of NUTS is as good as hand-tuned HMC, and sometimes better.
- NUTS is the standard MCMC algorithm used in Stan.

No U-turn sampler (NUTS): Basic idea

- Basic idea: Roughly, NUTS propagates the Hamiltonian dynamics until the trajectory starts going back towards where it started.
- Mathematically, "going back" means the momentum vector z points in the direction of the starting position x_0.
- The intuition is that we want to make as big a move as possible. So going back toward where we started is undesirable.
- Unfortunately, stopping as soon as we start going back doesn't work since it doesn't yield a valid MCMC move. (Detailed balance is violated.)

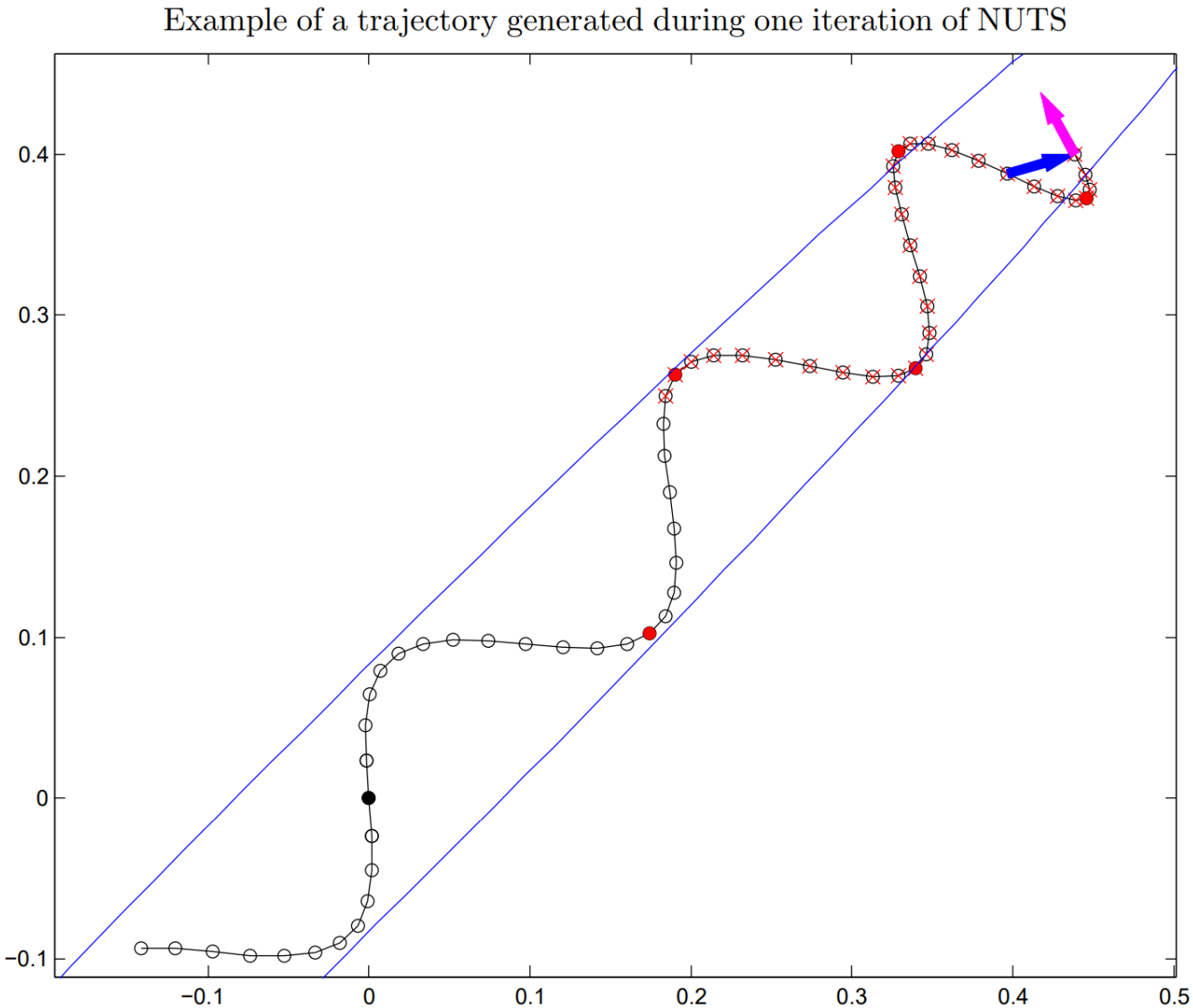
No U-turn sampler (NUTS): Basic idea

- NUTS is a way of constructing trajectories to avoid going back, while still satisfying detailed balance.
- This is implemented via a procedure for choosing L adaptively.
- NUTS also employs a method of adapting ε and M during the burn-in period.
- ε and M are held fixed after burn-in is over, but L is chosen adaptively at each iteration.

No U-turn sampler (NUTS): Method

- To ensure detailed balance is satisfied, NUTS runs the dynamics both forward and backward in time.
- First, it goes forward or backward 1 step, then forward or backward 2 steps, then forward or backward 4 steps, and so on, doubling each time.
- This doubling process stops when either the forward endpoint or the backward endpoint starts to go back.
- Finally, NUTS samples from the set of points generated during the doubling procedure, in a way that preserves detailed balance.

No U-turn sampler (NUTS): Method



(figure from Hoffman & Gelman (2014))

Illustration: NUTS versus Metropolis and Gibbs

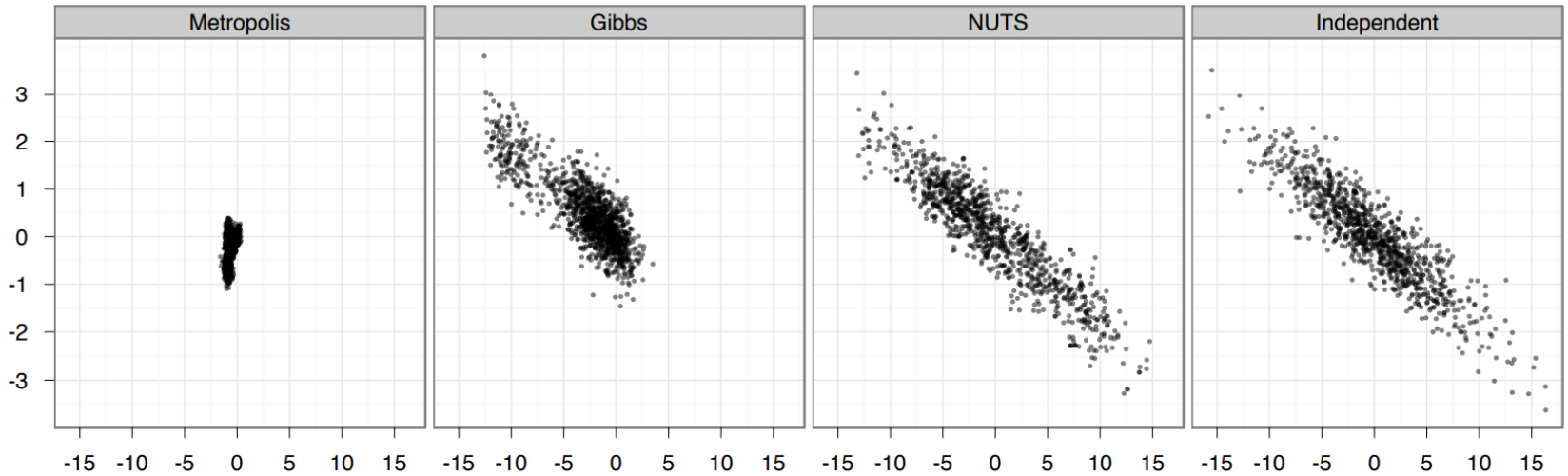
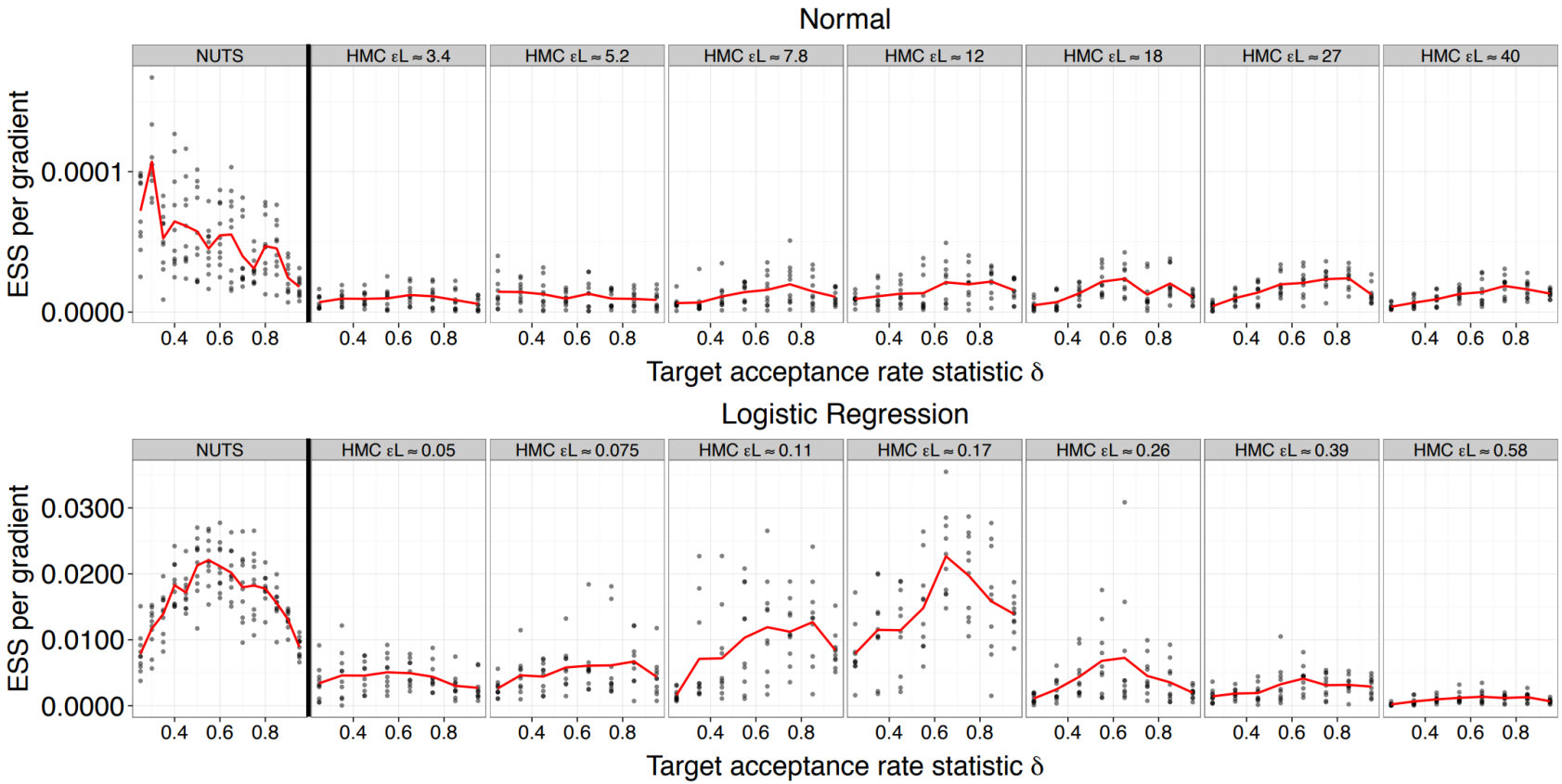


Figure 7: Samples generated by random-walk Metropolis, Gibbs sampling, and NUTS. The plots compare 1,000 independent draws from a highly correlated 250-dimensional distribution (right) with 1,000,000 samples (thinned to 1,000 samples for display) generated by random-walk Metropolis (left), 1,000,000 samples (thinned to 1,000 samples for display) generated by Gibbs sampling (second from left), and 1,000 samples generated by NUTS (second from right). Only the first two dimensions are shown here.

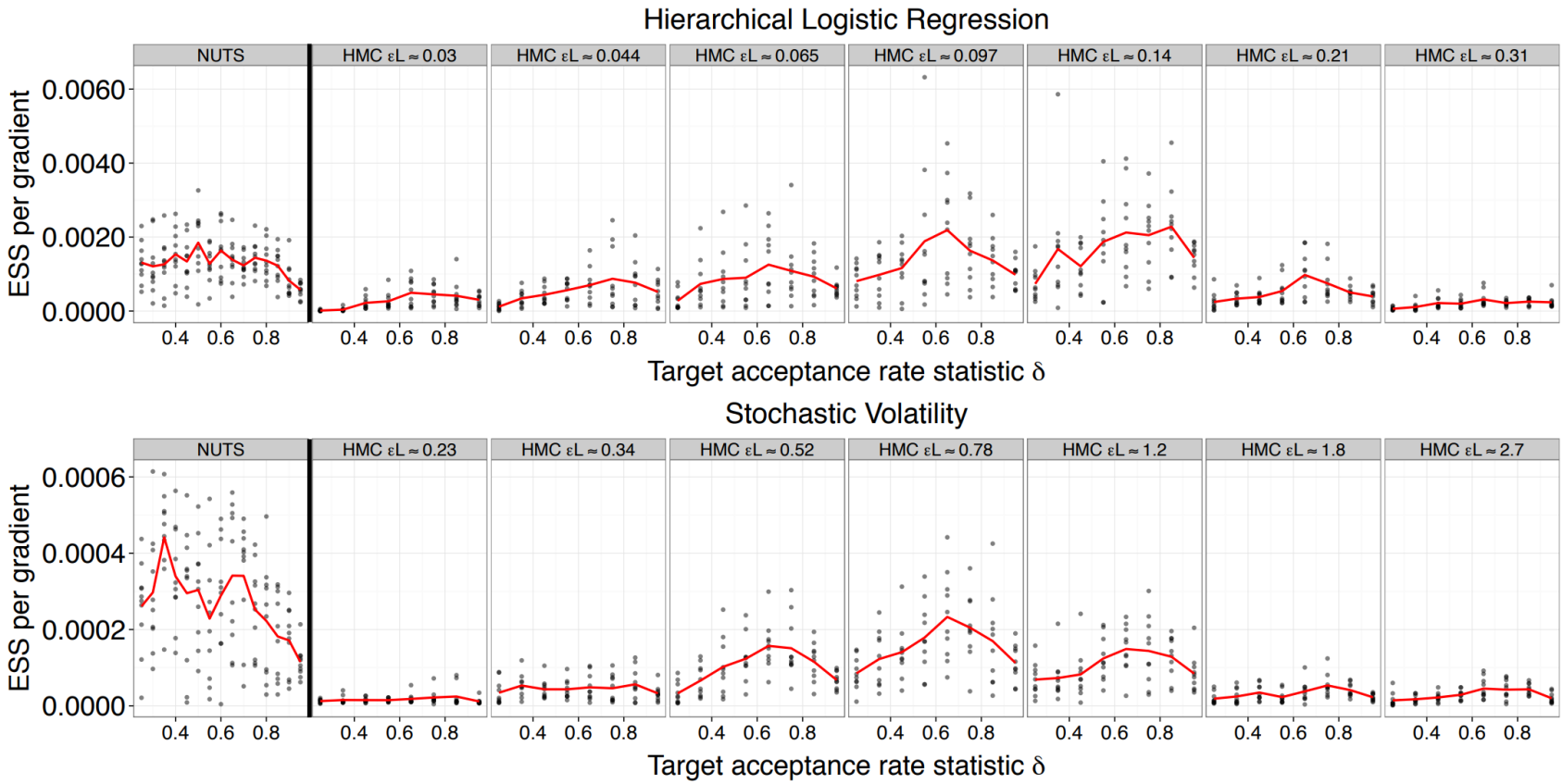
(figure from Hoffman & Gelman (2014))

Illustration: Performance of NUTS versus HMC



(figure from Hoffman & Gelman (2014))

Illustration: Performance of NUTS versus HMC



(figure from Hoffman & Gelman (2014))

References and supplements

- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian Data Analysis. CRC press.

- Neal, R. M. (2011). MCMC using Hamiltonian dynamics. Handbook of Markov chain Monte Carlo, 113.

- Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research, 15(1), 1593-1623.