Scalable Bayesian Inference: Monte Carlo Methods for Modern Data

Chris Nemeth

Greek Stochastics, July 2026

MARS

This course

Question: How do we do Bayesian inference when models are high-dimensional and datasets are massive?

This course

Question: How do we do Bayesian inference when models are high-dimensional and datasets are massive?

1. **Foundations** – Monte Carlo and MCMC.

This course

Question: How do we do Bayesian inference when models are high-dimensional and datasets are massive?

1. **Foundations** – Monte Carlo and MCMC.
2. **Scaling with dimension** d – proposal choice determines efficiency: RWM, MALA and HMC.

This course

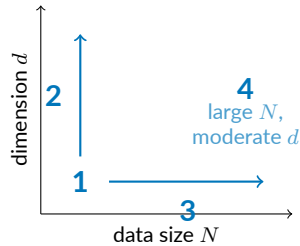
Question: How do we do Bayesian inference when models are high-dimensional and datasets are massive?

1. **Foundations** – Monte Carlo and MCMC.
2. **Scaling with dimension** d – proposal choice determines efficiency: RWM, MALA and HMC.
3. **Scaling with data size** N – stochastic gradient MCMC: give up exactness, gain speed.

This course

Question: How do we do Bayesian inference when models are high-dimensional and datasets are massive?

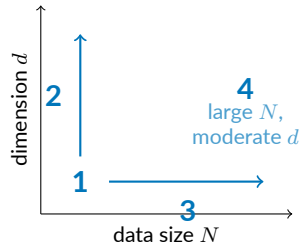
1. **Foundations** – Monte Carlo and MCMC.
2. **Scaling with dimension** d – proposal choice determines efficiency: RWM, MALA and HMC.
3. **Scaling with data size** N – stochastic gradient MCMC: give up exactness, gain speed.
4. **Having it both ways** – exact subsampling MCMC: Metropolis–Hastings with Scalable Subsampling.



This course

Question: How do we do Bayesian inference when models are high-dimensional and datasets are massive?

1. **Foundations** – Monte Carlo and MCMC.
2. **Scaling with dimension** d – proposal choice determines efficiency: RWM, MALA and HMC.
3. **Scaling with data size** N – stochastic gradient MCMC: give up exactness, gain speed.
4. **Having it both ways** – exact subsampling MCMC: Metropolis–Hastings with Scalable Subsampling.



Based on: Fearnhead, Nemeth, Oates and Sherlock (2025). *Scalable Monte Carlo for Bayesian Learning*. Cambridge University Press.

Part 1

Foundations: Monte Carlo and MCMC

Bayesian Statistics

The posterior density for $\theta \in \mathbb{R}^d$ given the data $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$, up to a constant of proportionality, is

$$\pi(\theta) := p(\theta|\mathbf{y}) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\theta),$$

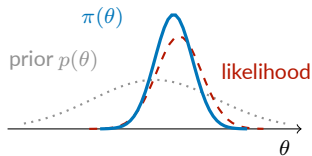
where $p(\theta)$ is a prior density for the parameters θ.

Bayesian Statistics

The posterior density for $\theta \in \mathbb{R}^d$ given the data $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$, up to a constant of proportionality, is

$$\pi(\theta) := p(\theta|\mathbf{y}) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\theta),$$

where $p(\theta)$ is a prior density for the parameters θ.



Motivating example: image classification

Bayesian Neural Networks

Assume that $y_1, y_2, \ldots, y_N$ are observed data, where each y_j represents one of G classes, i.e. $y_j \in \{1, 2, \ldots, G\}$, with d-dimensional covariate vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$.

Bayesian Neural Networks

Assume that $y_1, y_2, \ldots, y_N$ are observed data, where each y_j represents one of G classes, i.e. $y_j \in \{1, 2, \ldots, G\}$, with d-dimensional covariate vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$. Under a simple two-layer neural network model, the probability of a particular class y_j is

$$\mathbb{P}(Y = y_j | \mathbf{z}_j, \theta) \propto \exp(\mathbf{A}_{y_j}^\top \sigma(\mathbf{B}^\top \mathbf{z}_j + \mathbf{b}) + a_{y_j}),$$

where σ is a (vector) softmax function applied to the hidden layer.

Bayesian Neural Networks

Assume that $y_1, y_2, \ldots, y_N$ are observed data, where each y_j represents one of G classes, i.e. $y_j \in \{1, 2, \ldots, G\}$, with d-dimensional covariate vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$. Under a simple two-layer neural network model, the probability of a particular class y_j is

$$\mathbb{P}(Y = y_j | \mathbf{z}_j, \theta) \propto \exp(\mathbf{A}_{y_j}^\top \sigma(\mathbf{B}^\top \mathbf{z}_j + \mathbf{b}) + a_{y_j}),$$

where σ is a (vector) softmax function applied to the hidden layer.

The parameters $\theta = \text{vec}(\mathbf{a}, \mathbf{A}, \mathbf{b}, \mathbf{B})$: the vectors $\mathbf{a}, \mathbf{b}$ are *biases* and the matrices $\mathbf{A}, \mathbf{B}$ are *weights*.

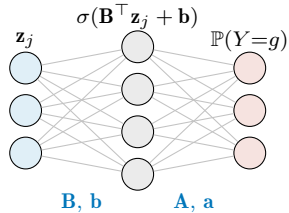
Bayesian Neural Networks

Assume that $y_1, y_2, \ldots, y_N$ are observed data, where each y_j represents one of G classes, i.e. $y_j \in \{1, 2, \ldots, G\}$, with d-dimensional covariate vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$. Under a simple two-layer neural network model, the probability of a particular class y_j is

$$\mathbb{P}(Y = y_j | \mathbf{z}_j, \theta) \propto \exp(\mathbf{A}_{y_j}^\top \sigma(\mathbf{B}^\top \mathbf{z}_j + \mathbf{b}) + a_{y_j}),$$

where σ is a (vector) softmax function applied to the hidden layer.

The parameters $\theta = \text{vec}(\mathbf{a}, \mathbf{A}, \mathbf{b}, \mathbf{B})$: the vectors $\mathbf{a}, \mathbf{b}$ are *biases* and the matrices $\mathbf{A}, \mathbf{B}$ are *weights*.



Assume the prior $p(\theta)$ is Gaussian.

Question: What is $\pi(\theta)$? Both d and N are large!

Monte Carlo Methods

Monte Carlo methods basically means simulation.

Monte Carlo Methods

Monte Carlo methods basically means simulation.

They are used to estimate $\mathbb{E}[\theta] = \int \theta\pi(\theta)\,\mathrm{d}\theta$ for a random variable θ where $\mathbb{E}[\theta]$ has no analytic form.

Monte Carlo Methods

Monte Carlo methods basically means simulation.

They are used to estimate $\mathbb{E}[\theta] = \int \theta\pi(\theta)\,\mathrm{d}\theta$ for a random variable θ where $\mathbb{E}[\theta]$ has no analytic form.

This is done as follows:-

1. Take a sample of size m, i.e., $\theta_1, \theta_2, \ldots, \theta_m$ from the distribution π. We assume that the observations are independent.
2. Then $\mathbb{E}[\theta] \approx \frac{1}{m}\sum_{i=1}^{m}\theta_i$.

Monte Carlo Methods

This generalises, for some function $\phi(\cdot)$

$$\mu = \mathbb{E}\left[\phi(\theta)\right] = \int \phi(\theta)\pi(\theta)\,\mathrm{d}\theta.$$

Then if we take a sample $\theta_1, \theta_2, \ldots, \theta_m$ from the distribution π, we have that

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} \phi(\theta_i)$$

is an unbiased estimate of μ.

Simple Monte Carlo

How do we get our samples $\theta_1, \theta_2, \ldots,$ anyway?

Simple Monte Carlo

How do we get our samples $\theta_1, \theta_2, \ldots,$ anyway?

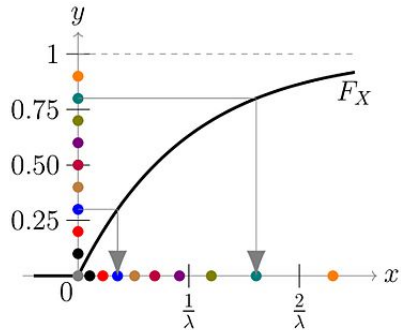
Assume that our density of interest, $\pi(\theta)$ has a distribution function F. Then we can use inversion sampling as follows:

Simple Monte Carlo

How do we get our samples $\theta_1, \theta_2, \ldots,$ anyway?

Assume that our density of interest, $\pi(\theta)$ has a distribution function F. Then we can use inversion sampling as follows:

- Generate a random u from Unif $[0, 1]$
- Find the inverse of F, i.e. F^{-1}
- Compute $\theta = F^{-1}(u)$. Then $\theta \sim F$.



Monte Carlo Inversion

Let's assume that we want to sample from an Exponential distribution with rate parameter λ, i.e. $\theta \sim \text{Exp}(\lambda)$.

Monte Carlo Inversion

Let's assume that we want to sample from an Exponential distribution with rate parameter λ, i.e. $\theta \sim \text{Exp}(\lambda)$.

Probability density function:

$\pi(\theta) = \lambda e^{-\lambda\theta}, \theta \geq 0$

Monte Carlo Inversion

Let's assume that we want to sample from an Exponential distribution with rate parameter λ, i.e. $\theta \sim \text{Exp}(\lambda)$.

Probability density function:

$\pi(\theta) = \lambda e^{-\lambda\theta}, \theta \geq 0$

Cumulative distribution function:

$F(\theta) = \int_0^\theta \lambda e^{-\lambda t} \mathrm{dt} = 1 - \mathrm{e}^{-\lambda\theta}$

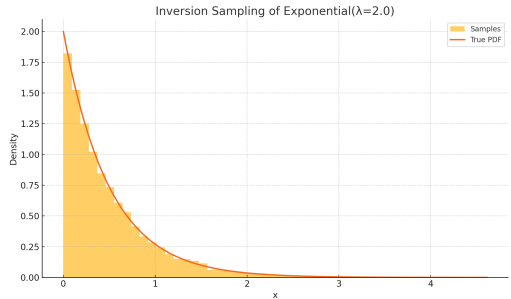
Monte Carlo Inversion

Let's assume that we want to sample from an Exponential distribution with rate parameter λ, i.e. $\theta \sim \text{Exp}(\lambda)$.

Probability density function:
$\pi(\theta) = \lambda e^{-\lambda\theta}, \theta \geq 0$

Cumulative distribution function:
$F(\theta) = \int_0^\theta \lambda e^{-\lambda t} \mathrm{dt} = 1 - \mathrm{e}^{-\lambda\theta}$

Inverse CDF: Since $u = F(\theta)$ and $u \sim \mathcal{U}[0, 1]$



$$u = 1 - e^{-\lambda\theta} \quad \Longrightarrow \quad \theta = -\frac{1}{\lambda}\ln(1-u)$$

Properties of Monte Carlo integration

- Unbiased.

Properties of Monte Carlo integration

- Unbiased.
- Improved convergence rates $\mathcal{O}(m^{-1/2})$ over numerical integration $\mathcal{O}(m^{-1/d})$ in high dimensions.

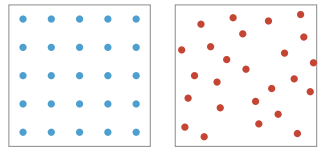
Properties of Monte Carlo integration

- Unbiased.
- Improved convergence rates $\mathcal{O}(m^{-1/2})$ over numerical integration $\mathcal{O}(m^{-1/d})$ in high dimensions.
- If the variance of $\phi(\theta)$ exists, then $\text{Var}\left[\hat{\mu}\right] = \frac{1}{m}\text{Var}\left[\phi(\theta)\right]$.

Properties of Monte Carlo integration

- Unbiased.
- Improved convergence rates $\mathcal{O}(m^{-1/2})$ over numerical integration $\mathcal{O}(m^{-1/d})$ in high dimensions.
- If the variance of $\phi(\theta)$ exists, then $\text{Var}\left[\hat{\mu}\right] = \frac{1}{m}\text{Var}\left[\phi(\theta)\right]$.
- Central limit theorem

$$\frac{\sqrt{m}(\hat{\mu} - \mu)}{\sqrt{\text{Var}\left[\phi(\theta)\right]}} \xrightarrow{D} \mathcal{N}(0, 1)$$



grid: $\mathcal{O}(m^{-1/d})$ MC: $\mathcal{O}(m^{-1/2})$

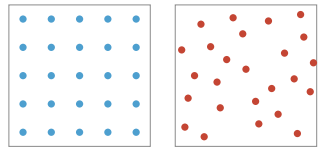
same m points – but only the Monte Carlo rate is dimension-free

Properties of Monte Carlo integration

- Unbiased.
- Improved convergence rates $\mathcal{O}(m^{-1/2})$ over numerical integration $\mathcal{O}(m^{-1/d})$ in high dimensions.
- If the variance of $\phi(\theta)$ exists, then $\text{Var}[\hat{\mu}] = \frac{1}{m}\text{Var}[\phi(\theta)]$.
- Central limit theorem

$$\frac{\sqrt{m}(\hat{\mu} - \mu)}{\sqrt{\text{Var}[\phi(\theta)]}} \xrightarrow{D} \mathcal{N}(0, 1)$$

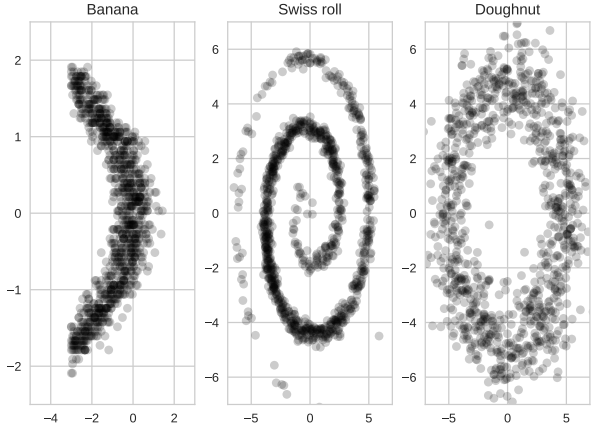
All the above hold provided $\{\theta_i\}$ are independent.



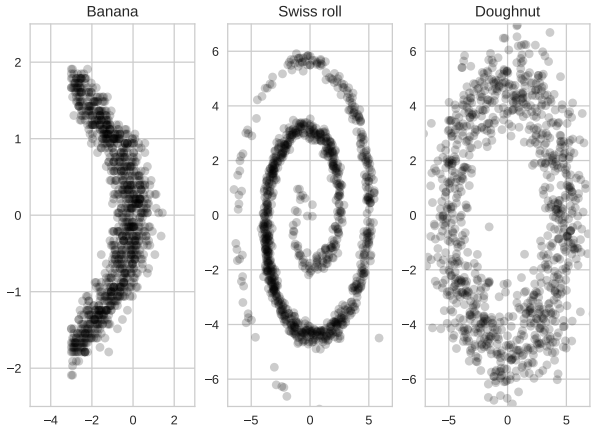
grid: $\mathcal{O}(m^{-1/d})$ MC: $\mathcal{O}(m^{-1/2})$

same m points – but only the Monte Carlo rate is dimension-free

What if $\pi(\theta)$ looks like this?



What if $\pi(\theta)$ looks like this?



No closed-form F^{-1} here – and these are only two-dimensional targets.

Tales from Los Alamos

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

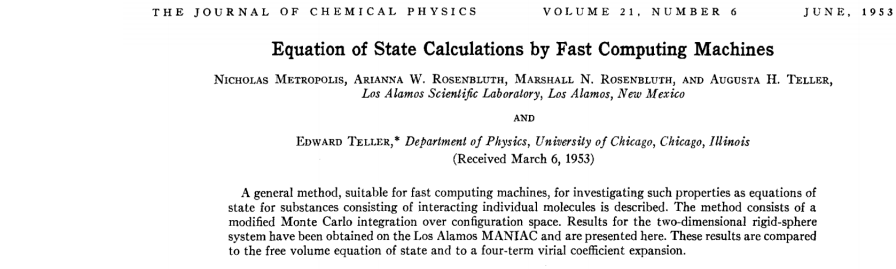
AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

Tales from Los Alamos



THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

- It's a beautifully simple idea. Explore your distribution by randomly walking through it.
- The key idea is to turn a global problem into a local one.
- Later known as the Metropolis algorithm (one of the top 10 algorithms of the 20^{th} century).

Tales from Los Alamos

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

- It's a beautifully simple idea. Explore your distribution by randomly walking through it.
- The key idea is to turn a global problem into a local one.
- Later known as the Metropolis algorithm (one of the top 10 algorithms of the 20^{th} century). Actually Arianna and Marshall did all of the work.

The Metropolis algorithm

DEMO

1: **Sample**:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

2: **Evaluate**:

$$\alpha(\theta, \theta') = \min\left\{1, \frac{\pi(\theta')}{\pi(\theta)}\right\}$$

3: **Accept-Reject**: With probability α accept $\theta = \theta'$ otherwise $\theta = \theta$

The Metropolis algorithm

DEMO

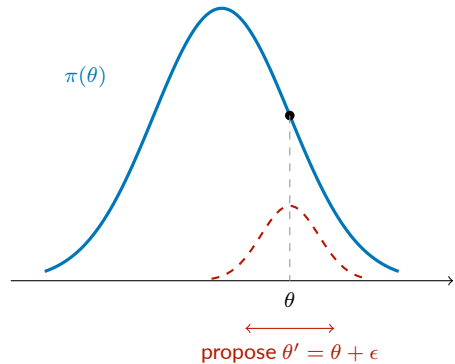
1: **Sample**:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

2: **Evaluate**:

$$\alpha(\theta, \theta') = \min\left\{1, \frac{\pi(\theta')}{\pi(\theta)}\right\}$$

3: **Accept-Reject**: With probability α accept $\theta = \theta'$ otherwise $\theta = \theta$



The Metropolis algorithm

DEMO

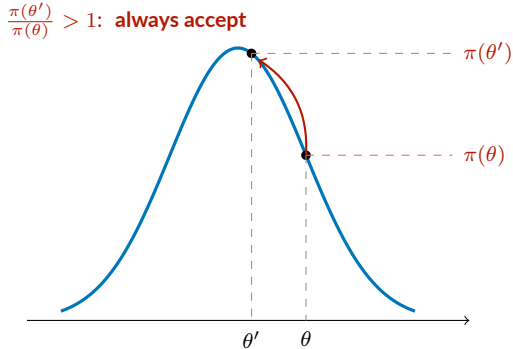
1: **Sample**:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

2: **Evaluate**:

$$\alpha(\theta, \theta') = \min\left\{1, \frac{\pi(\theta')}{\pi(\theta)}\right\}$$

3: **Accept-Reject**: With probability α accept $\theta = \theta'$ otherwise $\theta = \theta$



The Metropolis algorithm

DEMO

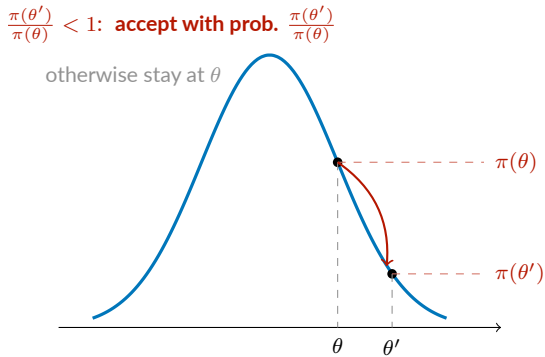
1: **Sample**:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

2: **Evaluate**:

$$\alpha(\theta, \theta') = \min\left\{1, \frac{\pi(\theta')}{\pi(\theta)}\right\}$$

3: **Accept-Reject**: With probability α accept $\theta = \theta'$ otherwise $\theta = \theta$



Metropolis-Hastings

Currently, we're proposing new samples θ' by perturbing the current value θ. We can use more general proposal mechanisms via a proposal density $q(\theta'|\theta)$.

This gives us the Metropolis-Hastings algorithm, which is the same as before, but with acceptance probability

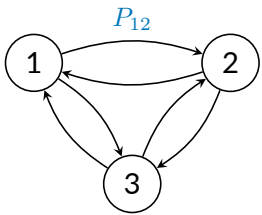
$$\alpha(\theta'|\theta) = \min\left\{1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)}\right\}.$$

A key benefit of this algorithm is that the unknown normalising constant cancels!

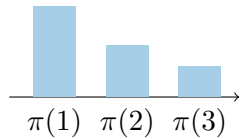
A Brief Primer on Markov chains

Dynamics:

$$P_{ij} = \Pr(X_t = j \mid X_{t-1} = i)$$



Invariant distribution: $\pi = \pi P$

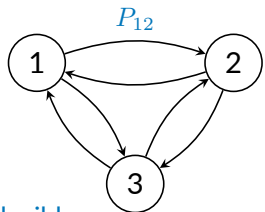


start in π, stay in π

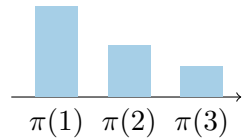
A Brief Primer on Markov chains

Dynamics:

$$P_{ij} = \Pr(X_t = j \mid X_{t-1} = i)$$



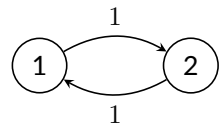
Invariant distribution: $\pi = \pi P$



start in π, stay in π

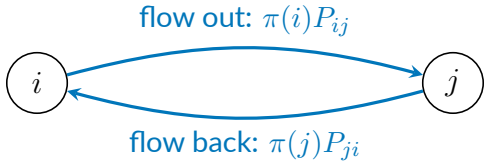
× reducible: can never cross over

× periodic: alternates forever, never settles



If irreducible (can reach anywhere) and aperiodic (no forced cycles), then π is unique and $\pi_t \to \pi$ for any starting distribution π_0.

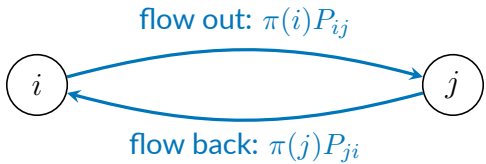
Detailed Balance



A sufficient condition for π to be invariant is detailed balance: for all i, j

$$\pi(i)P_{ij} = \pi(j)P_{ji} \qquad \text{“flow } i \to j = \text{flow } j \to i\text{”}$$

Detailed Balance

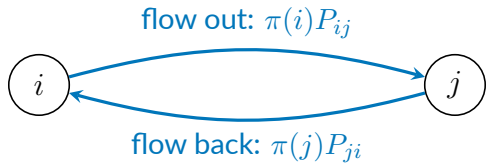


A sufficient condition for π to be invariant is detailed balance: for all i, j

$$\pi(i)P_{ij} = \pi(j)P_{ji} \qquad \text{“flow } i \to j = \text{flow } j \to i\text{”}$$

If all pairwise flows balance, no state gains or loses mass – and the chain is π-reversible: started from π and run backwards, it looks the same.

Detailed Balance



A sufficient condition for π to be invariant is detailed balance: for all i, j

$$\pi(i)P_{ij} = \pi(j)P_{ji} \qquad \text{“flow } i \to j = \text{flow } j \to i\text{”}$$

If all pairwise flows balance, no state gains or loses mass – and the chain is π-reversible: started from π and run backwards, it looks the same.

Continuous state spaces: same story, with a transition density in place of the matrix,

$$\pi(\theta)p(\theta'|\theta) = \pi(\theta')p(\theta|\theta'), \text{ for almost all } \theta, \theta'.$$

This is the condition we will use to design MCMC algorithms.

Markov chain Monte Carlo (MCMC)

We are going to consider Markov chains that are designed to have a specific stationary distribution, $\pi(\theta)$ – commonly a Bayesian posterior, known only up to proportionality.

Markov chain Monte Carlo (MCMC)

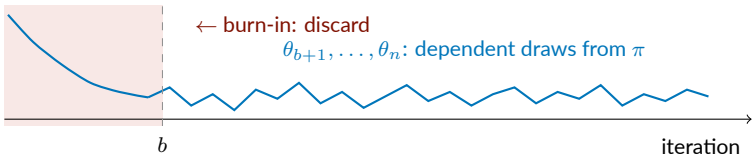
We are going to consider Markov chains that are designed to have a specific stationary distribution, $\pi(\theta)$ – commonly a Bayesian posterior, known only up to proportionality.

The idea is that we can simulate our Markov chain, to produce draws $\theta_1, \ldots, \theta_n$. Drop the first b draws (as burn-in), and then use $\theta_{b+1}, \ldots, \theta_n$ as (approximate, dependent) samples from π.

Markov chain Monte Carlo (MCMC)

We are going to consider Markov chains that are designed to have a specific stationary distribution, $\pi(\theta)$ – commonly a Bayesian posterior, known only up to proportionality.

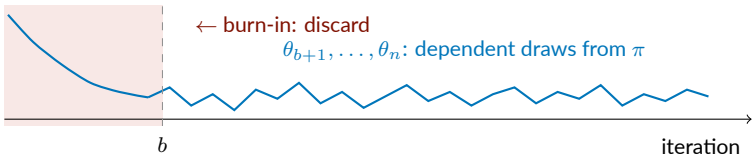
The idea is that we can simulate our Markov chain, to produce draws $\theta_1, \ldots, \theta_n$. Drop the first b draws (as burn-in), and then use $\theta_{b+1}, \ldots, \theta_n$ as (approximate, dependent) samples from π.



Markov chain Monte Carlo (MCMC)

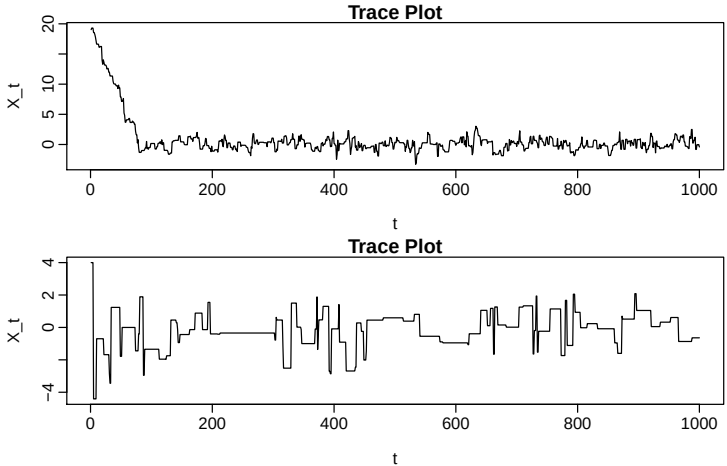
We are going to consider Markov chains that are designed to have a specific stationary distribution, $\pi(\theta)$ – commonly a Bayesian posterior, known only up to proportionality.

The idea is that we can simulate our Markov chain, to produce draws $\theta_1, \ldots, \theta_n$. Drop the first b draws (as burn-in), and then use $\theta_{b+1}, \ldots, \theta_n$ as (approximate, dependent) samples from π.



These can be used to estimate posterior expectations/probabilities. The accuracy depends on the speed of convergence to stationarity (how much burn-in) and the auto-correlation of the chain.

MCMC: Two Examples



Top: started far from the posterior mass – a transient (burn-in) before reaching stationarity.

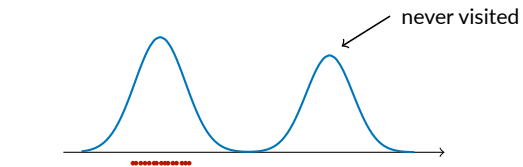
Bottom: a poorly mixing chain – long flat stretches where proposals are rejected.

Convergence

- It can be difficult to know whether your MCMC algorithm has converged to stationarity.
- The key issues tend to be (i) missing modes if the target is multi-modal; or (ii) not sampling the tails if the target is heavy-tailed.
- Often to try and detect convergence people run independent MCMC algorithms from different starting values.

Convergence

- It can be difficult to know whether your MCMC algorithm has converged to stationarity.
- The key issues tend to be (i) missing modes if the target is multi-modal; or (ii) not sampling the tails if the target is heavy-tailed.
- Often to try and detect convergence people run independent MCMC algorithms from different starting values.



Assessing accuracy

To estimate $\mathsf{E}_\pi[g(\theta)]$ from MCMC we use the empirical average

$$\frac{1}{n-b}\sum_{i=b+1}^{n} g(\theta_i).$$

The variance of this estimator is approximately

$$\frac{1}{n-b}\mathsf{Var}[g(\theta)]\left(1+2\sum_{l=1}^{\infty}\mathsf{Cor}[g(\theta_1), g(\theta_{1+l})]\right)$$

Correlation in the chain inflates the Monte Carlo variance.

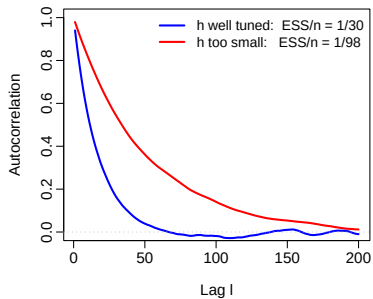
Assessing accuracy

So we can assess the accuracy by (i) choosing some function $g(\cdot)$; (ii) plotting the empirical autocorrelation of $g(\theta_i)$; and estimating the auto-correlation time

$$\tau = 1 + 2\sum_{l=1}^{\infty} \text{Cor}[g(\theta_1), g(\theta_{1+l})]$$

or the effective sample size $(n-b)/\tau$.

These measures depend on the choice of $g(\cdot)$.

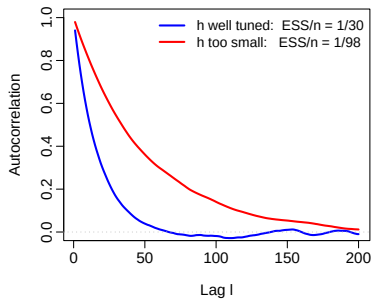


Assessing accuracy

So we can assess the accuracy by (i) choosing some function $g(\cdot)$; (ii) plotting the empirical autocorrelation of $g(\theta_i)$; and estimating the auto-correlation time

$$\tau = 1 + 2\sum_{l=1}^{\infty} \text{Cor}[g(\theta_1), g(\theta_{1+l})]$$

or the effective sample size $(n-b)/\tau$.



These measures depend on the choice of $g(\cdot)$.

Estimating τ is difficult – often people assume that the correlation is negligible beyond some large lag L and use $\hat{\tau} = 1 + 2\sum_{l=1}^{L} \hat{\text{Cor}}[g(\theta_1), g(\theta_{1+l})]$.

Part 1: Checkpoint

- Monte Carlo replaces intractable integrals with sample averages, converging at rate $\mathcal{O}(m^{-1/2})$ regardless of dimension.

Part 1: Checkpoint

- Monte Carlo replaces intractable integrals with sample averages, converging at rate $\mathcal{O}(m^{-1/2})$ regardless of dimension.
- MCMC lets us sample from unnormalised targets by constructing a Markov chain with π as its stationary distribution.

Part 1: Checkpoint

- Monte Carlo replaces intractable integrals with sample averages, converging at rate $\mathcal{O}(m^{-1/2})$ regardless of dimension.
- MCMC lets us sample from unnormalised targets by constructing a Markov chain with π as its stationary distribution.
- Correlation in the chain reduces the effective sample size – efficiency is about how far we move per unit of computation.

Part 1: Checkpoint

- Monte Carlo replaces intractable integrals with sample averages, converging at rate $\mathcal{O}(m^{-1/2})$ regardless of dimension.
- MCMC lets us sample from unnormalised targets by constructing a Markov chain with π as its stationary distribution.
- Correlation in the chain reduces the effective sample size – efficiency is about how far we move per unit of computation.

Next: How should we choose the proposal $q(\theta'|\theta)$?
And what happens as the dimension d grows?

Part 2

Proposals and scaling with dimension

Random-walk Metropolis

Arguably the simplest and most common MCMC algorithm is random walk Metropolis (RWM). This uses a proposal distribution that is a random walk:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, h\,\mathbf{I}),$$

where ϵ is independent of θ.

Since the proposal is symmetric, $q(\theta'|\theta) = q(\theta|\theta')$, the proposal density cancels in the acceptance probability.

Random-walk Metropolis

Arguably the simplest and most common MCMC algorithm is random walk Metropolis (RWM). This uses a proposal distribution that is a random walk:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, h\,\mathbf{I}),$$

where ϵ is independent of θ.

Since the proposal is symmetric, $q(\theta'|\theta) = q(\theta|\theta')$, the proposal density cancels in the acceptance probability.

A user then needs to choose the step size h.

Random-walk Metropolis

Arguably the simplest and most common MCMC algorithm is random walk Metropolis (RWM). This uses a proposal distribution that is a random walk:

$$\theta' = \theta + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, h\,\mathbf{I}),$$

where ϵ is independent of θ.

Since the proposal is symmetric, $q(\theta'|\theta) = q(\theta|\theta')$, the proposal density cancels in the acceptance probability.

A user then needs to choose the step size h. This choice is crucial for the mixing of the algorithm.

The step-size dilemma

DEMO

- h too small: almost every proposal is accepted, but the chain moves in tiny steps – exploration is a slow diffusion.

The step-size dilemma

DEMO

- h too small: almost every proposal is accepted, but the chain moves in tiny steps – exploration is a slow diffusion.
- h too large: proposals frequently land in low-probability regions and are rejected – the chain sticks.

The step-size dilemma

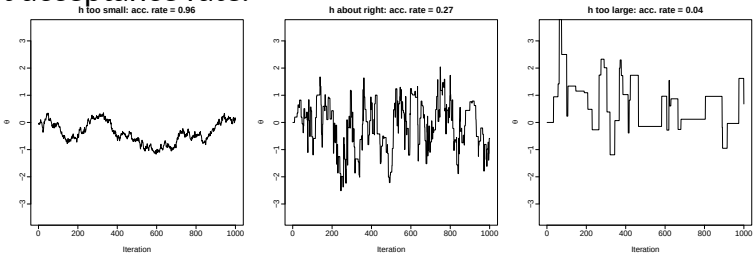
DEMO

- h too small: almost every proposal is accepted, but the chain moves in tiny steps – exploration is a slow diffusion.
- h too large: proposals frequently land in low-probability regions and are rejected – the chain sticks.
- Somewhere in between is a sweet spot that balances jump distance against acceptance rate.

The step-size dilemma

DEMO

- h too small: almost every proposal is accepted, but the chain moves in tiny steps – exploration is a slow diffusion.
- h too large: proposals frequently land in low-probability regions and are rejected – the chain sticks.
- Somewhere in between is a sweet spot that balances jump distance against acceptance rate.



What should we optimise?

A natural criterion – the expected squared jump distance: how far does the chain move per iteration?

$$\mathrm{ESJD} := \mathbb{E}\left[\|\theta_{k+1} - \theta_k\|^2\right] = \mathbb{E}\left[\|\theta' - \theta\|^2 \, \alpha(\theta, \theta')\right]$$

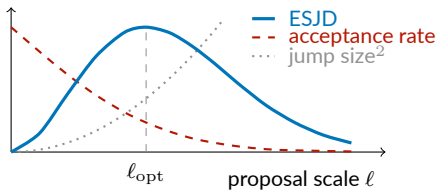
Rejected proposals contribute zero: ESJD trades jump size^2 against acceptance.

What should we optimise?

A natural criterion – the expected squared jump distance: how far does the chain move per iteration?

$$\mathrm{ESJD} := \mathbb{E}\left[\|\theta_{k+1} - \theta_k\|^2\right] = \mathbb{E}\left[\|\theta' - \theta\|^2 \, \alpha(\theta, \theta')\right]$$

Rejected proposals contribute zero: ESJD trades jump size^2 against acceptance.

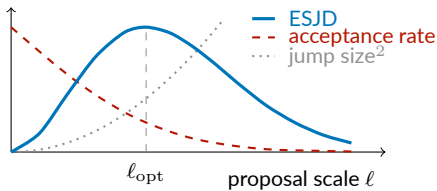


What should we optimise?

A natural criterion – the expected squared jump distance: how far does the chain move per iteration?

$$\mathrm{ESJD} := \mathbb{E}\left[\|\theta_{k+1} - \theta_k\|^2\right] = \mathbb{E}\left[\|\theta' - \theta\|^2 \, \alpha(\theta, \theta')\right]$$

Rejected proposals contribute zero: ESJD trades jump size^2 against acceptance.



Maximising ESJD $\approx$ minimising lag-1 autocorrelation; in the $d \to \infty$ limit all efficiency measures agree with it (Roberts & Rosenthal, 2001) – the acceptance rate is the dial.

Optimal scaling of RWM

Question: how should h change with dimension d?

Optimal scaling of RWM

Question: how should h change with dimension d?

Classic analysis (Roberts, Gelman & Gilks, 1997): let the target be d IID copies, $\pi(\theta) = \prod_{j=1}^{d} f(\theta^{(j)})$, and let $d \to \infty$.

Optimal scaling of RWM

Question: how should h change with dimension d?

Classic analysis (Roberts, Gelman & Gilks, 1997): let the target be d IID copies, $\pi(\theta) = \prod_{j=1}^{d} f(\theta^{(j)})$, and let $d \to \infty$.

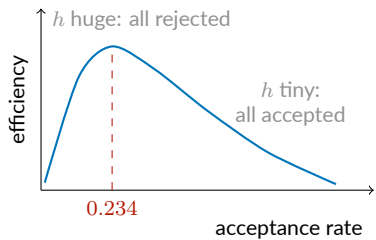
- The ESJD-maximising step size (proposal variance) scales as $h \propto d^{-1}$: std. dev. $\sqrt{h} \approx 2.38/\sqrt{d}$.
- At this optimum, the acceptance rate converges to 0.234.

Optimal scaling of RWM

Question: how should h change with dimension d?

Classic analysis (Roberts, Gelman & Gilks, 1997): let the target be d IID copies, $\pi(\theta) = \prod_{j=1}^{d} f(\theta^{(j)})$, and let $d \to \infty$.

- The ESJD-maximising step size (proposal variance) scales as $h \propto d^{-1}$: std. dev. $\sqrt{h} \approx 2.38/\sqrt{d}$.
- At this optimum, the acceptance rate converges to 0.234.
- The chain needs $\mathcal{O}(d)$ iterations per effective sample.

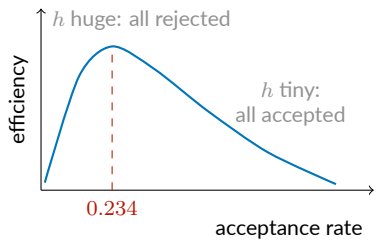


Optimal scaling of RWM

Question: how should h change with dimension d?

Classic analysis (Roberts, Gelman & Gilks, 1997): let the target be d IID copies, $\pi(\theta) = \prod_{j=1}^{d} f(\theta^{(j)})$, and let $d \to \infty$.

- The ESJD-maximising step size (proposal variance) scales as $h \propto d^{-1}$: std. dev. $\sqrt{h} \approx 2.38/\sqrt{d}$.
- At this optimum, the acceptance rate converges to 0.234.
- The chain needs $\mathcal{O}(d)$ iterations per effective sample.



Practical rule: tune RWM to accept roughly a quarter of proposals.

Where do 2.38 and 0.234 come from? (1/2)

Product target, so the log acceptance ratio is a sum over coordinates. Taylor expand each one, with $s := (\log f)'$ the score and $\theta'_j = \theta_j + \sqrt{h} Z_j$:

$$\log \frac{\pi(\theta')}{\pi(\theta)} = \sum_{j=1}^{d} \left[\log f(\theta'_j) - \log f(\theta_j)\right] \approx \sum_{j=1}^{d} \Big[\underbrace{\sqrt{h}\, Z_j\, s(\theta_j)}_{\text{mean } 0,\ \text{var } hI} + \underbrace{\tfrac{h}{2} Z_j^2\, s'(\theta_j)}_{\text{mean } -\frac{h}{2} I} \Big]$$

where $I := \mathbb{E}_f\left[s(\theta)^2\right]$ is the average squared score, and $\mathbb{E}_f\left[s'\right] = -\mathbb{E}_f\left[s^2\right] = -I$ by integration by parts.

Where do 2.38 and 0.234 come from? (1/2)

Product target, so the log acceptance ratio is a sum over coordinates. Taylor expand each one, with $s := (\log f)'$ the score and $\theta'_j = \theta_j + \sqrt{h} Z_j$:

$$\log \frac{\pi(\theta')}{\pi(\theta)} = \sum_{j=1}^{d} \left[\log f(\theta'_j) - \log f(\theta_j)\right] \approx \sum_{j=1}^{d} \Big[\underbrace{\sqrt{h}\, Z_j\, s(\theta_j)}_{\text{mean } 0,\ \text{var } hI} + \underbrace{\tfrac{h}{2} Z_j^2\, s'(\theta_j)}_{\text{mean } -\frac{h}{2} I} \Big]$$

where $I := \mathbb{E}_f\left[s(\theta)^2\right]$ is the average squared score, and $\mathbb{E}_f\left[s'\right] = -\mathbb{E}_f\left[s^2\right] = -I$ by integration by parts.

A sum of d small independent terms $\Rightarrow$ CLT:

$$\log \frac{\pi(\theta')}{\pi(\theta)} \approx \mathcal{N}\big(-\tfrac{1}{2} h d I,\ h d I\big)\,.$$

Where do 2.38 and 0.234 come from? (1/2)

Product target, so the log acceptance ratio is a sum over coordinates. Taylor expand each one, with $s := (\log f)'$ the score and $\theta'_j = \theta_j + \sqrt{h} Z_j$:

$$\log \frac{\pi(\theta')}{\pi(\theta)} = \sum_{j=1}^{d} \left[\log f(\theta'_j) - \log f(\theta_j)\right] \approx \sum_{j=1}^{d} \Big[\underbrace{\sqrt{h}\, Z_j\, s(\theta_j)}_{\text{mean } 0,\ \text{var } hI} + \underbrace{\tfrac{h}{2} Z_j^2\, s'(\theta_j)}_{\text{mean } -\frac{h}{2} I} \Big]$$

where $I := \mathbb{E}_f\left[s(\theta)^2\right]$ is the average squared score, and
$\mathbb{E}_f\left[s'\right] = -\mathbb{E}_f\left[s^2\right] = -I$ by integration by parts.
A sum of d small independent terms $\Rightarrow$ CLT:

$$\log \frac{\pi(\theta')}{\pi(\theta)} \approx \mathcal{N}\big(-\tfrac{1}{2} hdI,\ hdI\big)\,.$$

The d^{-1} rate: a non-degenerate limit needs $hd = \mathcal{O}(1)$: write $h = \ell^2/d$ – any larger and acceptance collapses to 0.

Where do 2.38 and 0.234 come from? (2/2)

With $h = \ell^2/d$, the log acceptance ratio is $G \approx \mathcal{N}(-\sigma^2/2, \sigma^2)$, $\sigma^2 = \ell^2 I$. The average acceptance rate splits into two Gaussian probabilities:

$$a(\ell) = \mathbb{E}\left[1 \wedge e^G\right] = \underbrace{\mathbb{P}\left(G > 0\right)}_{\text{uphill: always accept}} + \underbrace{\mathbb{E}\left[e^G;\, G \leq 0\right]}_{\text{downhill: accept w.p. } e^G}$$

- First term: standardising, $\mathbb{P}\left(G > 0\right) = \Phi(-\sigma/2)$.

Where do 2.38 and 0.234 come from? (2/2)

With $h = \ell^2/d$, the log acceptance ratio is $G \approx \mathcal{N}(-\sigma^2/2, \sigma^2)$, $\sigma^2 = \ell^2 I$. The average acceptance rate splits into two Gaussian probabilities:

$$a(\ell) = \mathbb{E}\left[1 \wedge e^G\right] = \underbrace{\mathbb{P}\left(G > 0\right)}_{\text{uphill: always accept}} + \underbrace{\mathbb{E}\left[e^G;\, G \leq 0\right]}_{\text{downhill: accept w.p. } e^G}$$

- First term: standardising, $\mathbb{P}\left(G > 0\right) = \Phi(-\sigma/2)$.
- Second term: the tilt e^G shifts the Gaussian's mean from $-\sigma^2/2$ to $+\sigma^2/2$ (since $\mathbb{E}\left[e^G\right] = 1$), so it equals $\Phi(-\sigma/2)$ too.

Where do 2.38 and 0.234 come from? (2/2)

With $h = \ell^2/d$, the log acceptance ratio is $G \approx \mathcal{N}(-\sigma^2/2, \sigma^2)$, $\sigma^2 = \ell^2 I$. The average acceptance rate splits into two Gaussian probabilities:

$$a(\ell) = \mathbb{E}\left[1 \wedge e^G\right] = \underbrace{\mathbb{P}\left(G > 0\right)}_{\text{uphill: always accept}} + \underbrace{\mathbb{E}\left[e^G;\, G \leq 0\right]}_{\text{downhill: accept w.p. } e^G}$$

- First term: standardising, $\mathbb{P}\left(G > 0\right) = \Phi(-\sigma/2)$.
- Second term: the tilt e^G shifts the Gaussian's mean from $-\sigma^2/2$ to $+\sigma^2/2$ (since $\mathbb{E}\left[e^G\right] = 1$), so it equals $\Phi(-\sigma/2)$ too.

$$\Rightarrow \quad a(\ell) = 2\Phi(-\ell\sqrt{I}/2)$$

Where do 2.38 and 0.234 come from? (2/2)

With $h = \ell^2/d$, the log acceptance ratio is $G \approx \mathcal{N}(-\sigma^2/2, \sigma^2)$, $\sigma^2 = \ell^2 I$. The average acceptance rate splits into two Gaussian probabilities:

$$a(\ell) = \mathbb{E}\left[1 \wedge e^G\right] = \underbrace{\mathbb{P}\left(G > 0\right)}_{\text{uphill: always accept}} + \underbrace{\mathbb{E}\left[e^G;\, G \le 0\right]}_{\text{downhill: accept w.p. } e^G}$$

- First term: standardising, $\mathbb{P}\left(G > 0\right) = \Phi(-\sigma/2)$.
- Second term: the tilt e^G shifts the Gaussian's mean from $-\sigma^2/2$ to $+\sigma^2/2$ (since $\mathbb{E}\left[e^G\right] = 1$), so it equals $\Phi(-\sigma/2)$ too.

$$\Rightarrow \quad a(\ell) = 2\Phi(-\ell\sqrt{I}/2)$$

Optimise the ESJD: maximise $\ell^2 a(\ell)$ – the trade-off curve from earlier. Numerically: $\ell_{\text{opt}} = 2.38/\sqrt{I}$ and $a(\ell_{\text{opt}}) = 0.234$. Everything model-specific enters through the single number I: that is why 0.234 is (nearly) universal.

Can we do better? Use the gradient

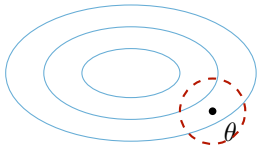
The random walk proposal is blind: it knows nothing about the shape of π.

Can we do better? Use the gradient

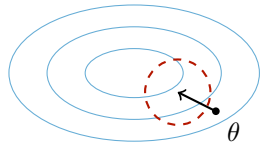
The random walk proposal is blind: it knows nothing about the shape of π. If we can evaluate $\nabla \log \pi(\theta)$, we can push proposals towards regions of high probability.

Can we do better? Use the gradient

The random walk proposal is blind: it knows nothing about the shape of π. If we can evaluate $\nabla \log \pi(\theta)$, we can push proposals towards regions of high probability.



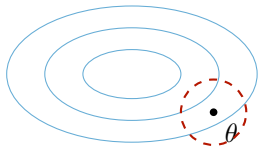
RWM: centred at θ, blind



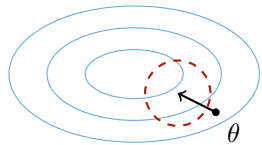
MALA: pushed uphill by $\frac{h}{2}\nabla \log \pi(\theta)$

Can we do better? Use the gradient

The random walk proposal is blind: it knows nothing about the shape of π. If we can evaluate $\nabla \log \pi(\theta)$, we can push proposals towards regions of high probability.



RWM: centred at θ, blind



MALA: pushed uphill by $\frac{h}{2}\nabla \log \pi(\theta)$

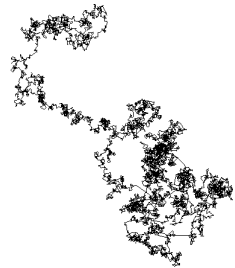
This is the idea behind the Langevin family of algorithms: use the gradient to build a smarter proposal, at the price of one gradient evaluation per iteration.

Langevin Diffusion

The Langevin diffusion is a continuous-time process, defined by the SDE

$$\mathrm{d}\theta_t = \frac{1}{2}\nabla \log \pi(\theta_t)\mathrm{d}t + \mathrm{d}B_t$$

where B_t is standard Brownian motion, produces samples with invariant distribution π.



Langevin Diffusion

The Langevin diffusion is a continuous-time process, defined by the SDE

$$\mathrm{d}\theta_t = \frac{1}{2}\nabla \log \pi(\theta_t)\mathrm{d}t + \mathrm{d}B_t$$

where B_t is standard Brownian motion, produces samples with invariant distribution π.

Using an Euler scheme we can simulate from the discretised Langevin dynamics:

$$\theta_{t+h} = \theta_t + \frac{h}{2}\nabla \log \pi(\theta_t) + \sqrt{h}\epsilon_t,$$

where ϵ_t is a vector of IID standard Gaussian rvs.

Metropolis-adjusted Langevin Algorithm (MALA)

DEMO

The discretisation introduces an error meaning that we're no longer sampling exactly from the diffusion (and hence from π).

Metropolis-adjusted Langevin Algorithm (MALA)

DEMO

The discretisation introduces an error meaning that we're no longer sampling exactly from the diffusion (and hence from π).

We can instead use the Langevin dynamics as a proposal distribution

$$q(\theta'|\theta) = \mathcal{N}\left(\theta' \,\middle|\, \theta + \frac{h}{2}\nabla \log \pi(\theta),\, h\right)$$

and correct for the discretisation with a Metropolis-Hastings accept-reject step.

Metropolis-adjusted Langevin Algorithm (MALA)

DEMO

The discretisation introduces an error meaning that we're no longer sampling exactly from the diffusion (and hence from π).

We can instead use the Langevin dynamics as a proposal distribution

$$q(\theta'|\theta) = \mathcal{N}\left(\theta' \,\middle|\, \theta + \frac{h}{2}\nabla \log \pi(\theta),\, h\right)$$

and correct for the discretisation with a Metropolis-Hastings accept-reject step.

This algorithm converges faster than the random walk scheme – gradient information pays off in high dimensions.

Optimal scaling of MALA

Repeating the optimal scaling analysis for MALA (Roberts & Rosenthal, 1998):

Optimal scaling of MALA

Repeating the optimal scaling analysis for MALA (Roberts & Rosenthal, 1998):

- The optimal step size scales as $h \propto d^{-1/3}$ – much larger steps than RWM's d^{-1}.
- At this optimum, the acceptance rate converges to 0.574.

Optimal scaling of MALA

Repeating the optimal scaling analysis for MALA (Roberts & Rosenthal, 1998):

- The optimal step size scales as $h \propto d^{-1/3}$ – much larger steps than RWM's d^{-1}.
- At this optimum, the acceptance rate converges to 0.574.
- The chain needs only $\mathcal{O}(d^{1/3})$ iterations per effective sample.

Optimal scaling of MALA

Repeating the optimal scaling analysis for MALA (Roberts & Rosenthal, 1998):

- The optimal step size scales as $h \propto d^{-1/3}$ – much larger steps than RWM's d^{-1}.
- At this optimum, the acceptance rate converges to 0.574.
- The chain needs only $\mathcal{O}(d^{1/3})$ iterations per effective sample.

Hamiltonian Monte Carlo: the idea

DEMO

Even MALA's moves are local: one gradient nudge plus noise. HMC borrows an idea from physics to make global moves.

Hamiltonian Monte Carlo: the idea

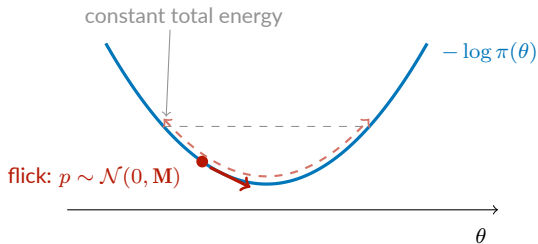
DEMO

Even MALA's moves are local: one gradient nudge plus noise. HMC borrows an idea from physics to make global moves. Think of a ball rolling on the energy surface $U(\theta) = -\log \pi(\theta)$: give it a random flick (kinetic energy $\frac{1}{2} p^\top \mathbf{M}^{-1} p$) and let it glide.

Hamiltonian Monte Carlo: the idea

DEMO

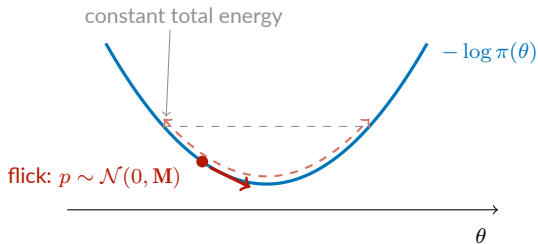
Even MALA's moves are local: one gradient nudge plus noise. HMC borrows an idea from physics to make global moves. Think of a ball rolling on the energy surface $U(\theta) = -\log \pi(\theta)$: give it a random flick (kinetic energy $\frac{1}{2} p^\top \mathbf{M}^{-1} p$) and let it glide.



Hamiltonian Monte Carlo: the idea

DEMO

Even MALA's moves are local: one gradient nudge plus noise. HMC borrows an idea from physics to make global moves. Think of a ball rolling on the energy surface $U(\theta) = -\log \pi(\theta)$: give it a random flick (kinetic energy $\frac{1}{2} p^\top \mathbf{M}^{-1} p$) and let it glide.



The ball moves fast through the valley (high π) and slows as it climbs – stop it after a while: a distant proposal that is still probable.

Hamiltonian dynamics

Augment θ with a momentum $p \sim \mathcal{N}(0, \mathbf{M})$
and target the joint distribution

$$\pi(\theta, p) \propto \exp\{-H(\theta, p)\},$$

$$H(\theta, p) = -\log \pi(\theta) + \tfrac{1}{2} p^\top \mathbf{M}^{-1} p,$$

whose θ-marginal is exactly π.

Hamiltonian dynamics

Augment θ with a momentum $p \sim \mathcal{N}(0, \mathbf{M})$ and target the joint distribution

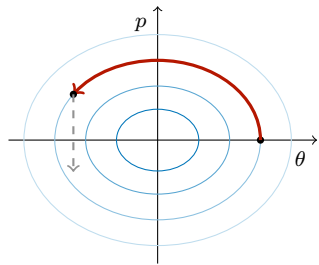
$$\pi(\theta, p) \propto \exp\{-H(\theta, p)\},$$

$$H(\theta, p) = -\log \pi(\theta) + \tfrac{1}{2} p^\top \mathbf{M}^{-1} p,$$

whose θ-marginal is exactly π.

Simulate the Hamiltonian dynamics

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = \mathbf{M}^{-1} p_t, \qquad \frac{\mathrm{d}p_t}{\mathrm{d}t} = \nabla \log \pi(\theta_t).$$



dynamics: move along a contour of H
refresh p: jump between contours

Hamiltonian dynamics

Augment θ with a momentum $p \sim \mathcal{N}(0, \mathbf{M})$ and target the joint distribution

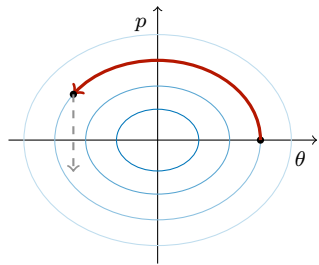
$$\pi(\theta, p) \propto \exp\{-H(\theta, p)\},$$

$$H(\theta, p) = -\log \pi(\theta) + \tfrac{1}{2} p^\top \mathbf{M}^{-1} p,$$

whose θ-marginal is exactly π.

Simulate the Hamiltonian dynamics

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = \mathbf{M}^{-1} p_t, \qquad \frac{\mathrm{d}p_t}{\mathrm{d}t} = \nabla \log \pi(\theta_t).$$



dynamics: move along a contour of H
refresh p: jump between contours

The dynamics conserve H, are reversible and volume-preserving – run exactly, the move would always be accepted. Refreshing $p \sim \mathcal{N}(0, \mathbf{M})$ each iteration moves between energy levels.

HMC in practice: the leapfrog integrator

Hamilton's equations rarely have a closed form – discretise with L leapfrog steps of size h, alternating half-steps in p with full steps in θ:

$$
\begin{aligned}
p_{t+h/2} &= p_t + \tfrac{h}{2}\nabla \log \pi(\theta_t), \\
\theta_{t+h} &= \theta_t + h\mathbf{M}^{-1}p_{t+h/2}, \\
p_{t+h} &= p_{t+h/2} + \tfrac{h}{2}\nabla \log \pi(\theta_{t+h}).
\end{aligned}
$$

HMC in practice: the leapfrog integrator

Hamilton's equations rarely have a closed form – discretise with L leapfrog steps of size h, alternating half-steps in p with full steps in θ:

$$\begin{aligned} p_{t+h/2} &= p_t + \tfrac{h}{2}\nabla \log \pi(\theta_t), \\ \theta_{t+h} &= \theta_t + h\mathbf{M}^{-1} p_{t+h/2}, \\ p_{t+h} &= p_{t+h/2} + \tfrac{h}{2}\nabla \log \pi(\theta_{t+h}). \end{aligned}$$

The endpoint (θ', p') is the proposal; correct the discretisation error with a Metropolis–Hastings step:

$$\alpha = \min\left\{1, \exp\left(H(\theta, p) - H(\theta', p')\right)\right\}.$$

HMC in practice: the leapfrog integrator

Hamilton's equations rarely have a closed form – discretise with L leapfrog steps of size h, alternating half-steps in p with full steps in θ:

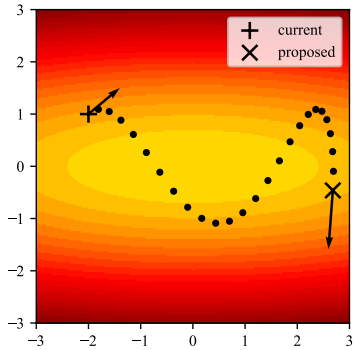
$$\begin{aligned} p_{t+h/2} &= p_t + \tfrac{h}{2}\nabla \log \pi(\theta_t), \\ \theta_{t+h} &= \theta_t + h\mathbf{M}^{-1} p_{t+h/2}, \\ p_{t+h} &= p_{t+h/2} + \tfrac{h}{2}\nabla \log \pi(\theta_{t+h}). \end{aligned}$$

The endpoint (θ', p') is the proposal; correct the discretisation error with a Metropolis–Hastings step:

$$\alpha = \min\left\{1, \exp\left(H(\theta, p) - H(\theta', p')\right)\right\}.$$

- Leapfrog is reversible and volume-preserving – a valid MH proposal.
- It nearly conserves H: acceptance stays high even after L long steps. Cost: L gradient evaluations per proposal.

One HMC proposal: L leapfrog steps



The trajectory sweeps across the target from the current state ($+$) to a distant proposal ($\times$); arrows show the initial and final momenta.

Optimal scaling of HMC

- The leapfrog step size scales as $h \propto d^{-1/4}$ (Beskos et al., 2013).
- Optimal acceptance rate: 0.651.

Optimal scaling of HMC

- The leapfrog step size scales as $h \propto d^{-1/4}$ (Beskos et al., 2013).
- Optimal acceptance rate: 0.651.
- Cost per effective sample: $\mathcal{O}(d^{1/4})$ trajectories, each of L gradient evaluations.

Optimal scaling of HMC

- The leapfrog step size scales as $h \propto d^{-1/4}$ (Beskos et al., 2013).
- Optimal acceptance rate: 0.651.
- Cost per effective sample: $\mathcal{O}(d^{1/4})$ trajectories, each of L gradient evaluations.

Tuning matters: the trajectory length Lh determines how far around the contours you travel – too far and you loop back to where you started (hence adaptive schemes like NUTS, used in Stan).

Scaling with dimension: summary

Algorithm	Optimal step size	Acceptance rate	Iterations per effective sample
Random Walk	$h \propto d^{-1}$	0.234	$\mathcal{O}(d)$
MALA	$h \propto d^{-1/3}$	0.574	$\mathcal{O}(d^{1/3})$
HMC	$h \propto d^{-1/4}$	0.651	$\mathcal{O}(d^{1/4})$

Scaling with dimension: summary

Algorithm	Optimal step size	Acceptance rate	Iterations per effective sample
Random Walk	$h \propto d^{-1}$	0.234	$\mathcal{O}(d)$
MALA	$h \propto d^{-1/3}$	0.574	$\mathcal{O}(d^{1/3})$
HMC	$h \propto d^{-1/4}$	0.651	$\mathcal{O}(d^{1/4})$

- Gradient information buys dramatically better dimension scaling.

Scaling with dimension: summary

Algorithm	Optimal step size	Acceptance rate	Iterations per effective sample
Random Walk	$h \propto d^{-1}$	0.234	$\mathcal{O}(d)$
MALA	$h \propto d^{-1/3}$	0.574	$\mathcal{O}(d^{1/3})$
HMC	$h \propto d^{-1/4}$	0.651	$\mathcal{O}(d^{1/4})$

- Gradient information buys dramatically better dimension scaling.
- These results assume product-form targets, but the ranking is robust in practice.

Scaling with dimension: summary

Algorithm	Optimal step size	Acceptance rate	Iterations per effective sample
Random Walk	$h \propto d^{-1}$	0.234	$\mathcal{O}(d)$
MALA	$h \propto d^{-1/3}$	0.574	$\mathcal{O}(d^{1/3})$
HMC	$h \propto d^{-1/4}$	0.651	$\mathcal{O}(d^{1/4})$

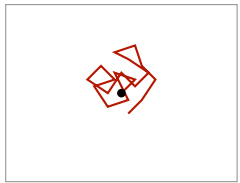
- Gradient information buys dramatically better dimension scaling.
- These results assume product-form targets, but the ranking is robust in practice.
- Related theory: match the shape of the proposal covariance to the target (preconditioning) – otherwise the worst-conditioned direction dictates h.

Where do the gains come from? Removing diffusivity

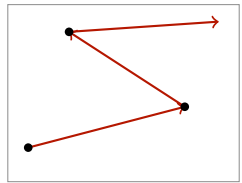
A reversible chain has no memory of its direction: each step is as likely to undo the last as to extend it. Exploration is a diffusion – n iterations move the chain a distance of order $\sqrt{n}$.

Where do the gains come from? Removing diffusivity

A reversible chain has no memory of its direction: each step is as likely to undo the last as to extend it. Exploration is a diffusion – n iterations move the chain a distance of order $\sqrt{n}$.



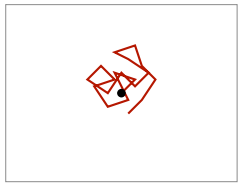
direction forgotten each step:
distance $\approx \sqrt{n}$



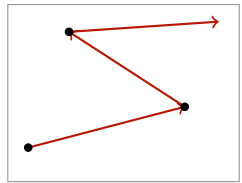
persistent direction:
distance $\approx n$

Where do the gains come from? Removing diffusivity

A reversible chain has no memory of its direction: each step is as likely to undo the last as to extend it. Exploration is a diffusion – n iterations move the chain a distance of order $\sqrt{n}$.



direction forgotten each step:
distance $\approx \sqrt{n}$

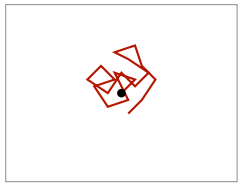


persistent direction:
distance $\approx n$

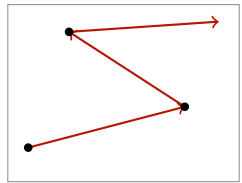
RWM's $\mathcal{O}(d)$ is this diffusivity made quantitative; HMC escapes it through persistence – momentum carries the chain onwards.

Where do the gains come from? Removing diffusivity

A reversible chain has no memory of its direction: each step is as likely to undo the last as to extend it. Exploration is a diffusion – n iterations move the chain a distance of order $\sqrt{n}$.



direction forgotten each step:
distance $\approx \sqrt{n}$



persistent direction:
distance $\approx n$

RWM's $\mathcal{O}(d)$ is this diffusivity made quantitative; HMC escapes it through persistence – momentum carries the chain onwards. **Can we get persistence without simulating Hamiltonian dynamics?**

The discrete bouncy particle sampler

DEMO

Lift the state to (θ, p), with velocity p and step size δ (Sherlock & Thiery, 2022):

The discrete bouncy particle sampler

DEMO

Lift the state to (θ, p), with velocity p and step size δ (Sherlock & Thiery, 2022):

- Glide: propose $\theta' = \theta + \delta p$, accept with $\min\{1, \pi(\theta')/\pi(\theta)\}$, keeping the same velocity.

The discrete bouncy particle sampler

DEMO

Lift the state to (θ, p), with velocity p and step size δ (Sherlock & Thiery, 2022):

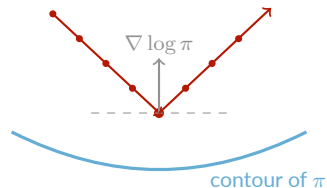
- Glide: propose $\theta' = \theta + \delta p$, accept with $\min\{1, \pi(\theta')/\pi(\theta)\}$, keeping the same velocity.
- Bounce: on rejection, reflect the velocity in the hyperplane $\perp \nabla \log \pi(\theta')$: $p \leftarrow p - 2(p \cdot \hat{g})\,\hat{g}$ (a delayed-rejection move – still exactly π-invariant).

The discrete bouncy particle sampler

DEMO

Lift the state to (θ, p), with velocity p and step size δ (Sherlock & Thiery, 2022):

- Glide: propose $\theta' = \theta + \delta p$, accept with $\min\{1, \pi(\theta')/\pi(\theta)\}$, keeping the same velocity.
- Bounce: on rejection, reflect the velocity in the hyperplane $\perp \nabla \log \pi(\theta')$: $p \leftarrow p - 2(p \cdot \hat{g})\,\hat{g}$ (a delayed-rejection move – still exactly π-invariant).
- Refresh: occasionally perturb p (irreducibility); a $d \to \infty$ analysis gives a robust tuning rule.



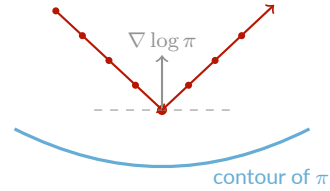
glide in steps of δp; bounce off the contour when rejected

The discrete bouncy particle sampler

DEMO

Lift the state to (θ, p), with velocity p and step size δ (Sherlock & Thiery, 2022):

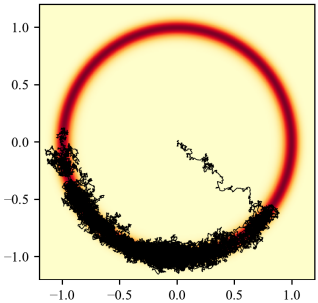
- Glide: propose $\theta' = \theta + \delta p$, accept with $\min\{1, \pi(\theta')/\pi(\theta)\}$, keeping the same velocity.
- Bounce: on rejection, reflect the velocity in the hyperplane $\perp \nabla \log \pi(\theta')$: $p \leftarrow p - 2(p \cdot \hat{g})\,\hat{g}$ (a delayed-rejection move – still exactly π-invariant).
- Refresh: occasionally perturb p (irreducibility); a $d \to \infty$ analysis gives a robust tuning rule.



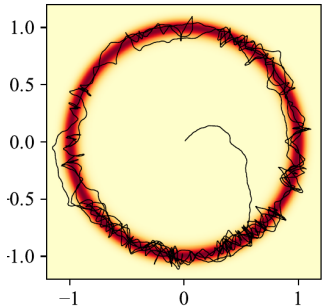
glide in steps of δp; bounce off the contour when rejected

Cost: one density evaluation per step, a gradient only at bounces – nothing to integrate, no trajectory length to tune.

Removing diffusivity: does it work?

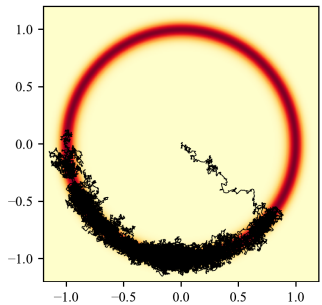


RWM, 50,000 iterations:
half the ring

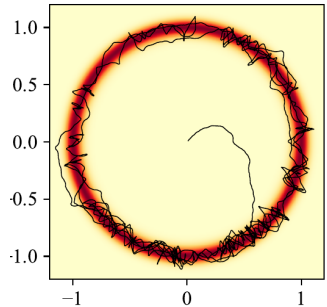


dBPS, 5,000 iterations:
the whole ring

Removing diffusivity: does it work?



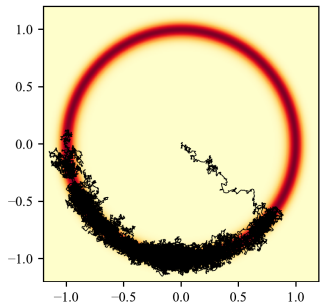
RWM, 50,000 iterations:
half the ring



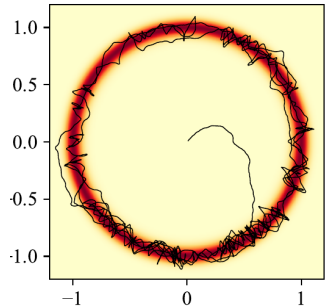
dBPS, 5,000 iterations:
the whole ring

Same step size; full exploration takes RWM $\sim$300,000 iterations – a $60\times$ gap, driven purely by persistence.

Removing diffusivity: does it work?



RWM, 50,000 iterations:
half the ring



dBPS, 5,000 iterations:
the whole ring

Same step size; full exploration takes RWM ∼300,000 iterations – a $60\times$ gap, driven purely by persistence. As $\delta \to 0$ the dBPS becomes a continuous-time sampler: the Bouncy Particle Sampler (a PDMP – see Part 3).

Part 2: Checkpoint

- Proposal choice determines how MCMC scales with dimension:
  - RWM – $\mathcal{O}(d)$,
  - MALA – $\mathcal{O}(d^{1/3})$,
  - HMC – $\mathcal{O}(d^{1/4})$

  iterations per effective sample.

Part 2: Checkpoint

- Proposal choice determines how MCMC scales with dimension:
  - RWM – $\mathcal{O}(d)$,
  - MALA – $\mathcal{O}(d^{1/3})$,
  - HMC – $\mathcal{O}(d^{1/4})$

  iterations per effective sample.
- The best methods all rely on the gradient of the log-posterior.

Part 2: Checkpoint

- Proposal choice determines how MCMC scales with dimension:
  - RWM – $\mathcal{O}(d)$,
  - MALA – $\mathcal{O}(d^{1/3})$,
  - HMC – $\mathcal{O}(d^{1/4})$

  iterations per effective sample.
- The best methods all rely on the gradient of the log-posterior.
- The underlying principle: efficiency comes from removing diffusivity – via momentum (HMC) or persistent velocities with bounces (dBPS).

Part 2: Checkpoint

- Proposal choice determines how MCMC scales with dimension:
  - RWM – $\mathcal{O}(d)$,
  - MALA – $\mathcal{O}(d^{1/3})$,
  - HMC – $\mathcal{O}(d^{1/4})$

  iterations per effective sample.
- The best methods all rely on the gradient of the log-posterior.
- The underlying principle: efficiency comes from removing diffusivity – via momentum (HMC) or persistent velocities with bounces (dBPS).
- But every iteration still evaluates the likelihood (and its gradient) on all N data points...

Next: what happens when $N = 10^6, 10^9, \ldots$?

Part 3

Scaling with data size: stochastic gradient MCMC

Big data and MCMC

We want to simulate from the posterior distribution – $\pi(\theta) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\theta)$.

Big data and MCMC

We want to simulate from the posterior distribution – $\pi(\theta) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\theta)$.

Metropolis-Hastings algorithm

2: **Evaluate**:

$$\begin{aligned}\alpha(\theta, \theta') &= \min\left\{1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)}\right\} \\ &= \min\left\{1, \frac{p(\theta')\prod_{i=1}^{N} p(y_i|\theta')q(\theta|\theta')}{p(\theta)\prod_{i=1}^{N} p(y_i|\theta)q(\theta'|\theta)}\right\}\end{aligned}$$

Q: What happens if N = 1,000,000? Every iteration costs $\mathcal{O}(N)$.

Big data and MCMC

We want to simulate from the posterior distribution – $\pi(\theta) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\theta)$.

Metropolis-Hastings algorithm

2: **Evaluate**:

$$\begin{aligned}\alpha(\theta, \theta') &= \text{min} \left\{ 1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)} \right\} \\ &= \text{min} \left\{ 1, \frac{p(\theta')\prod_{i=1}^{N} p(y_i|\theta')q(\theta|\theta')}{p(\theta)\prod_{i=1}^{N} p(y_i|\theta)q(\theta'|\theta)} \right\}\end{aligned}$$

Q: What happens if N = 1,000,000?

Big data and MCMC

We want to simulate from the posterior distribution – $\pi(\theta) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\theta)$.

Metropolis-Hastings algorithm

2: **Evaluate**:

$$\begin{aligned}\alpha(\theta, \theta') &= \text{min} \left\{ 1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)} \right\} \\ &= \text{min} \left\{ 1, \frac{p(\theta')\prod_{i=1}^{N} p(y_i|\theta')q(\theta|\theta')}{p(\theta)\prod_{i=1}^{N} p(y_i|\theta)q(\theta'|\theta)} \right\}\end{aligned}$$

Q: What happens if N = 1,000,000? Every iteration costs $\mathcal{O}(N)$.

MALA and Scalability

Consider the proposal with Langevin dynamics

$$q(\theta'|\theta_k) = \mathcal{N}\left(\theta_k + \frac{h}{2}\sum_{i=1}^{N} \nabla \log p(y_i|\theta_k) + \nabla \log p(\theta_k), h\right)$$

where we accept $\theta_{k+1} = \theta'$ with probability

$$\min\left\{1, \frac{p(\theta')\prod_{i=1}^{N} p(y_i|\theta')q(\theta_k|\theta')}{p(\theta_k)\prod_{i=1}^{N} p(y_i|\theta_k)q(\theta'|\theta_k)}\right\}.$$

We have an $\mathcal{O}(N)$ calculation for q and for the acceptance probability.

MALA and Scalability

Consider the proposal with Langevin dynamics

$$q(\theta'|\theta_k) = \mathcal{N}\left(\theta_k + \frac{h}{2}\sum_{i=1}^{N} \nabla \log p(y_i|\theta_k) + \nabla \log p(\theta_k), h\right)$$

where we accept $\theta_{k+1} = \theta'$ with probability

$$\min\left\{1, \frac{p(\theta')\prod_{i=1}^{N} p(y_i|\theta')q(\theta_k|\theta')}{p(\theta_k)\prod_{i=1}^{N} p(y_i|\theta_k)q(\theta'|\theta_k)}\right\}.$$

Idea 1: drop the accept-reject step entirely.

Unadjusted Langevin Algorithm (ULA)

DEMO

We can simulate from the distribution $\pi(\theta)$ by simply running the discretised Langevin algorithm, which is known as the unadjusted Langevin algorithm, using some fixed step size h.

$$\theta_{t+h} = \theta_t + \frac{h}{2}\nabla \log \pi(\theta_t) + \sqrt{h}\epsilon_t,$$

where ϵ_t is a vector of IID standard Gaussian rvs.

Unadjusted Langevin Algorithm (ULA)

DEMO

We can simulate from the distribution $\pi(\theta)$ by simply running the discretised Langevin algorithm, which is known as the unadjusted Langevin algorithm, using some fixed step size h.

$$\theta_{t+h} = \theta_t + \frac{h}{2}\nabla \log \pi(\theta_t) + \sqrt{h}\epsilon_t,$$

where ϵ_t is a vector of IID standard Gaussian rvs.

It is important to note that the ULA algorithm does not produce samples exactly from $\pi(\theta)$. This is because, for fixed h, ULA only approximately follows the Langevin diffusion. Only in the case where $h \to 0$ will the dynamics be exact.

How wrong is ULA?

For strongly log-concave targets and small enough h, non-asymptotic bounds exist (Dalalyan, 2017; Durmus & Moulines, 2017; this form: Dalalyan & Karagulyan, 2019). After K iterations, with $\tilde{\pi}_K$ the distribution of θ_K:

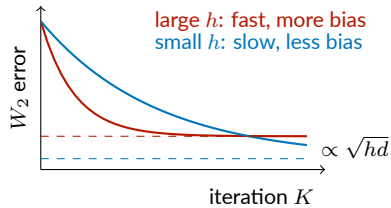
$$W_2(\tilde{\pi}_K, \pi) \leq \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{initialisation: decays geometrically}} + \underbrace{C_2 (hd)^{1/2}}_{\text{discretisation bias}}$$

How wrong is ULA?

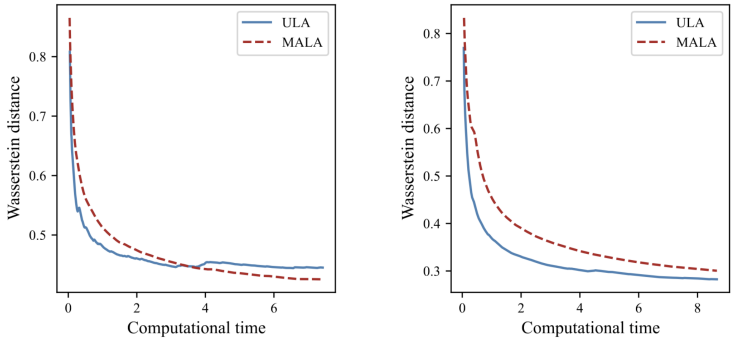
For strongly log-concave targets and small enough h, non-asymptotic bounds exist (Dalalyan, 2017; Durmus & Moulines, 2017; this form: Dalalyan & Karagulyan, 2019). After K iterations, with $\tilde{\pi}_K$ the distribution of θ_K:

$$W_2(\tilde{\pi}_K, \pi) \leq \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{initialisation: decays geometrically}} + \underbrace{C_2 (hd)^{1/2}}_{\text{discretisation bias}}$$

- The bias term does not vanish as $K \to \infty$: ULA targets an approximation to π.
- Smaller h = smaller bias but slower convergence: a bias-variance trade-off chosen by the user.

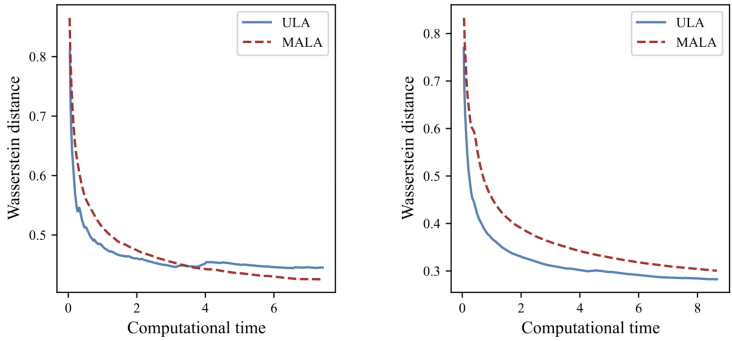


ULA vs MALA: accuracy per unit of compute



Wasserstein distance to π against computational time; $N = 1000$ (left), $N = 10{,}000$ (right).

ULA vs MALA: accuracy per unit of compute



Wasserstein distance to π against computational time; $N = 1000$ (left), $N = 10{,}000$ (right).

ULA races ahead (no $\mathcal{O}(N)$ accept-reject step) but levels off at its bias floor; exact MALA keeps improving.

ULA didn't fix the $\mathcal{O}(N)$ problem...

ULA removed the accept-reject step, but the gradient

$$\nabla \log \pi(\theta) = \nabla \log p(\theta) + \sum_{i=1}^{N} \nabla \log p(y_i|\theta)$$

is still an $\mathcal{O}(N)$ calculation at every iteration.

ULA didn't fix the $\mathcal{O}(N)$ problem…

ULA removed the accept-reject step, but the gradient

$$\nabla \log \pi(\theta) = \nabla \log p(\theta) + \sum_{i=1}^{N} \nabla \log p(y_i|\theta)$$

is still an $\mathcal{O}(N)$ calculation at every iteration.

Idea 2: replace the full-data gradient with a subsampled estimate.

This is exactly the trick that makes stochastic gradient descent work in machine learning – here we apply it to sampling.

Stochastic gradient Langevin dynamics

DEMO

Welling & Teh (2011) proposed replacing the log-posterior gradient with the following unbiased estimate

$$\widehat{\nabla \log \pi}(\theta) := \nabla \log p(\theta) + \frac{N}{n} \sum_{i \in S_k} \nabla \log p(y_i|\theta)$$

for some subsample S_k of $\{1, \ldots, N\}$, with $|S_k| = n$, where $n \ll N$.

each iteration: touch only the n blue points, reweight by N/n

Stochastic gradient Langevin dynamics

DEMO

Welling & Teh (2011) proposed replacing the log-posterior gradient with the following unbiased estimate

$$\widehat{\nabla \log \pi}(\theta) := \nabla \log p(\theta) + \frac{N}{n} \sum_{i \in S_k} \nabla \log p(y_i|\theta)$$

for some subsample S_k of $\{1, \ldots, N\}$, with $|S_k| = n$, where $n \ll N$.

each iteration: touch only the n blue points, reweight by N/n

A single update of SGLD is then

$$\theta_{k+1} = \theta_k + \frac{h}{2} \widehat{\nabla \log \pi}(\theta_k) + \sqrt{h}\zeta_k,$$

where ζ_k is a d-dimensional Gaussian random variable.

Stochastic gradient Langevin dynamics

DEMO

Welling & Teh (2011) proposed replacing the log-posterior gradient with the following unbiased estimate

$$\widehat{\nabla \log \pi}(\theta) := \nabla \log p(\theta) + \frac{N}{n} \sum_{i \in S_k} \nabla \log p(y_i|\theta)$$

for some subsample S_k of $\{1, \ldots, N\}$, with $|S_k| = n$, where $n \ll N$.

each iteration: touch only the n blue points, reweight by N/n

A single update of SGLD is then

$$\theta_{k+1} = \theta_k + \frac{h}{2} \widehat{\nabla \log \pi}(\theta_k) + \sqrt{h}\zeta_k,$$

where ζ_k is a d-dimensional Gaussian random variable.
Note: We now have discretisation error + subsampling error.

Stochastic gradient Langevin dynamics

DEMO

Welling & Teh (2011) proposed replacing the log-posterior gradient with the following unbiased estimate

$$\widehat{\nabla \log \pi}(\theta) := \nabla \log p(\theta) + \frac{N}{n} \sum_{i \in S_k} \nabla \log p(y_i|\theta)$$

for some subsample S_k of $\{1, \ldots, N\}$, with $|S_k| = n$, where $n \ll N$.

each iteration: touch only the n blue points, reweight by N/n

A single update of SGLD is then

$$\theta_{k+1} = \theta_k + \frac{h}{2} \widehat{\nabla \log \pi}(\theta_k) + \sqrt{h}\zeta_k,$$

where ζ_k is a d-dimensional Gaussian random variable.
Note: We now have discretisation error + subsampling error. But each iteration costs $\mathcal{O}(n)$, not $\mathcal{O}(N)$!

Convergence of SGLD

Let θ_K be the Kth iterate of the SGLD algorithm and $\tilde{\pi}_K$ be its distribution. If π satisfies some strongly log-concave assumptions, then (Dalalyan & Karagulyan, 2019)

$$W_2(\tilde{\pi}_K, \pi) \leq \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{initialisation}} + \underbrace{C_2 (hd)^{1/2}}_{\text{discretisation}} + \underbrace{C_3 \sigma (hd)^{1/2}}_{\text{gradient noise}},$$

where $C_1, C_2, C_3 \geq 0$ are constants, σ^2 bounds the variance of each component of $\widehat{\nabla \log \pi}(\theta)$, and $W_2(\cdot, \cdot)$ is the 2-Wasserstein distance. Same theorem as for ULA: that was the $\sigma = 0$ case.

Convergence of SGLD

Let θ_K be the Kth iterate of the SGLD algorithm and $\tilde{\pi}_K$ be its distribution. If π satisfies some strongly log-concave assumptions, then (Dalalyan & Karagulyan, 2019)

$$W_2(\tilde{\pi}_K, \pi) \leq \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{initialisation}} + \underbrace{C_2 (hd)^{1/2}}_{\text{discretisation}} + \underbrace{C_3 \sigma (hd)^{1/2}}_{\text{gradient noise}},$$

where $C_1, C_2, C_3 \geq 0$ are constants, σ^2 bounds the variance of each component of $\widehat{\nabla \log \pi}(\theta)$, and $W_2(\cdot, \cdot)$ is the 2-Wasserstein distance. Same theorem as for ULA: that was the $\sigma = 0$ case.

Problem: Under SGLD, the variance σ^2 is of order N^2/n. Therefore the bound on $W_2(\tilde{\pi}_K, \pi)$ is dominated by a term of order $N(hd/n)^{1/2}$.

Control variates (SGLD-CV)

Let $\hat{\theta}$ be a fixed value of the parameter that is within $\mathcal{O}(N^{-\frac{1}{2}})$ of the posterior mode.

Control variates (SGLD-CV)

Let $\hat{\theta}$ be a fixed value of the parameter that is within $\mathcal{O}(N^{-\frac{1}{2}})$ of the posterior mode.

The log-posterior gradient can then be re-written as

$$\nabla \log \pi(\theta) = \nabla \log \pi(\hat{\theta}) + [\nabla \log \pi(\theta) - \nabla \log \pi(\hat{\theta})],$$

Control variates (SGLD-CV)

Let $\hat{\theta}$ be a fixed value of the parameter that is within $\mathcal{O}(N^{-\frac{1}{2}})$ of the posterior mode.

The log-posterior gradient can then be re-written as

$$\nabla \log \pi(\theta) = \nabla \log \pi(\hat{\theta}) + [\nabla \log \pi(\theta) - \nabla \log \pi(\hat{\theta})],$$

the bracketed term on the right-hand side can be unbiasedly estimated by

$$\begin{aligned}\left[\nabla \widehat{\log \pi}(\theta) - \nabla \widehat{\log \pi}(\hat{\theta})\right] =& \nabla \log p(\theta) - \nabla \log p(\hat{\theta}) \\ &+ \frac{N}{n} \sum_{i \in S} \left[\nabla \log p(y_i|\theta) - \nabla \log p(y_i|\hat{\theta})\right]\end{aligned}$$

Control variates (SGLD-CV)

Let $\hat{\theta}$ be a fixed value of the parameter that is within $\mathcal{O}(N^{-\frac{1}{2}})$ of the posterior mode.

The log-posterior gradient can then be re-written as

$$\nabla \log \pi(\theta) = \nabla \log \pi(\hat{\theta}) + [\nabla \log \pi(\theta) - \nabla \log \pi(\hat{\theta})],$$

the bracketed term on the right-hand side can be unbiasedly estimated by

$$\begin{aligned}\left[\widehat{\nabla \log \pi}(\theta) - \widehat{\nabla \log \pi}(\hat{\theta})\right] =& \nabla \log p(\theta) - \nabla \log p(\hat{\theta}) \\ &+ \frac{N}{n} \sum_{i \in S} \left[\nabla \log p(y_i|\theta) - \nabla \log p(y_i|\hat{\theta})\right]\end{aligned}$$

Finding $\hat{\theta}$ requires an $\mathcal{O}(N)$ pre-processing step, where we use *stochastic gradient descent*. This step is quick and requires only one pass through the data.

Control variates (SGLD-CV)

Intuition: If the gradient of the likelihood for a single observation is smooth in θ then we will have

$$\nabla \log p(y_i|\theta) \approx \nabla \log p(y_i|\hat{\theta}) \text{ if } \theta \approx \hat{\theta}.$$

Control variates (SGLD-CV)

Intuition: If the gradient of the likelihood for a single observation is smooth in θ then we will have

$$\nabla \log p(y_i|\theta) \approx \nabla \log p(y_i|\hat{\theta}) \text{ if } \theta \approx \hat{\theta}.$$

So when θ is close to $\hat{\theta}$ the two terms in the bracket are highly correlated and the variance of the estimator collapses.

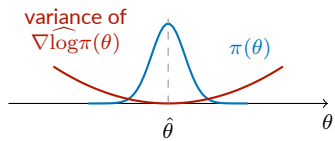
Control variates (SGLD-CV)

Intuition: If the gradient of the likelihood for a single observation is smooth in θ then we will have

$$\nabla \log p(y_i|\theta) \approx \nabla \log p(y_i|\hat{\theta}) \text{ if } \theta \approx \hat{\theta}.$$

So when θ is close to $\hat{\theta}$ the two terms in the bracket are highly correlated and the variance of the estimator collapses.

Since the posterior concentrates around $\hat{\theta}$ at rate $N^{-1/2}$, this is exactly the region where the chain spends its time.



variance is small exactly where π has its mass

Similar control variate ideas are used in Zig-Zag (Bierkens et al., 2019), ScaLE (Pollock et al., 2020) – and in Part 4 of this course.

Why the variance actually drops

Both estimators are unbiased – what changes is what the subsample must estimate. Writing $g_i(\theta) := \nabla \log p(y_i|\theta)$, with i a uniformly drawn index:

$$\mathrm{Var}\left[\widehat{\nabla \log \pi}_{\mathrm{SGLD}}\right] = \frac{N^2}{n}\mathrm{Var}_i\left[g_i(\theta)\right], \qquad \mathrm{Var}\left[\widehat{\nabla \log \pi}_{\mathrm{CV}}\right] = \frac{N^2}{n}\mathrm{Var}_i\left[g_i(\theta) - g_i(\hat{\theta})\right].$$

Why the variance actually drops

Both estimators are unbiased – what changes is what the subsample must estimate. Writing $g_i(\theta) := \nabla \log p(y_i|\theta)$, with i a uniformly drawn index:

$$\text{Var}\left[\widehat{\nabla \log \pi}_{\text{SGLD}}\right] = \frac{N^2}{n}\text{Var}_i\left[g_i(\theta)\right], \qquad \text{Var}\left[\widehat{\nabla \log \pi}_{\text{CV}}\right] = \frac{N^2}{n}\text{Var}_i\left[g_i(\theta) - g_i(\hat{\theta})\right].$$

- SGLD: the $g_i(\theta)$ are $\mathcal{O}(1)$ apart – so $\sigma^2 = \mathcal{O}(N^2/n)$.

Why the variance actually drops

Both estimators are unbiased – what changes is what the subsample must estimate. Writing $g_i(\theta) := \nabla \log p(y_i|\theta)$, with i a uniformly drawn index:

$$\text{Var}\left[\widehat{\nabla \log \pi}_{\text{SGLD}}\right] = \frac{N^2}{n}\text{Var}_i\left[g_i(\theta)\right], \qquad \text{Var}\left[\widehat{\nabla \log \pi}_{\text{CV}}\right] = \frac{N^2}{n}\text{Var}_i\left[g_i(\theta) - g_i(\hat{\theta})\right].$$

- SGLD: the $g_i(\theta)$ are $\mathcal{O}(1)$ apart – so $\sigma^2 = \mathcal{O}(N^2/n)$.
- SGLD-CV: smoothness, $\|g_i(\theta) - g_i(\hat{\theta})\| \leq L_i\|\theta - \hat{\theta}\|$, plus concentration, $\|\theta - \hat{\theta}\|^2 = \mathcal{O}_p(1/N)$ in stationarity:

$$\text{Var}\left[\widehat{\nabla \log \pi}_{\text{CV}}\right] \leq \frac{N^2}{n}\bar{L}^2\|\theta - \hat{\theta}\|^2 = \mathcal{O}\left(\frac{N}{n}\right)$$ – a factor-N reduction.

Why the variance actually drops

Both estimators are unbiased – what changes is what the subsample must estimate. Writing $g_i(\theta) := \nabla \log p(y_i|\theta)$, with i a uniformly drawn index:

$$\mathrm{Var}\left[\widehat{\nabla \log \pi}_{\mathrm{SGLD}}\right] = \frac{N^2}{n}\mathrm{Var}_i\left[g_i(\theta)\right], \qquad \mathrm{Var}\left[\widehat{\nabla \log \pi}_{\mathrm{CV}}\right] = \frac{N^2}{n}\mathrm{Var}_i\left[g_i(\theta) - g_i(\hat{\theta})\right].$$

- SGLD: the $g_i(\theta)$ are $\mathcal{O}(1)$ apart – so $\sigma^2 = \mathcal{O}(N^2/n)$.
- SGLD-CV: smoothness, $\|g_i(\theta) - g_i(\hat{\theta})\| \le L_i\|\theta - \hat{\theta}\|$, plus concentration, $\|\theta - \hat{\theta}\|^2 = \mathcal{O}_p(1/N)$ in stationarity:

$$\mathrm{Var}\left[\widehat{\nabla \log \pi}_{\mathrm{CV}}\right] \le \frac{N^2}{n}\bar{L}^2\|\theta - \hat{\theta}\|^2 = \mathcal{O}\left(\frac{N}{n}\right) \text{ – a factor-}N\text{ reduction.}$$

The classical control-variate principle: $\mathrm{Var}\left[A - B\right] = \mathrm{Var}\left[A\right] + \mathrm{Var}\left[B\right] - 2\,\mathrm{Cov}\left[A, B\right]$ – subtracting a correlated, computable proxy cancels the shared variation; in the W_2 bound the gradient-noise term shrinks by $\sqrt{N}$.

Does it help? Computational cost for fixed accuracy

Baker, Fearnhead, Fox & Nemeth (2019): for a fixed accuracy target, how must the per-iteration subsample size n grow with N?

Does it help? Computational cost for fixed accuracy

Baker, Fearnhead, Fox & Nemeth (2019): for a fixed accuracy target, how must the per-iteration subsample size n grow with N?

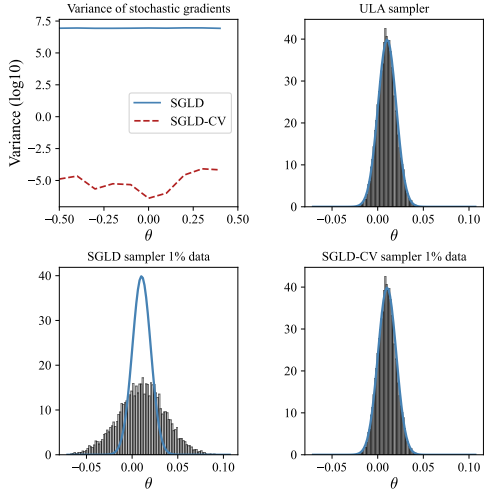
Algorithm	Subsample size n for fixed accuracy
SGLD	$\mathcal{O}(N)$
SGLD-CV	$\mathcal{O}(1)$

Does it help? Computational cost for fixed accuracy

Baker, Fearnhead, Fox & Nemeth (2019): for a fixed accuracy target, how must the per-iteration subsample size n grow with N?

Algorithm	Subsample size n for fixed accuracy
SGLD	$\mathcal{O}(N)$
SGLD-CV	$\mathcal{O}(1)$

With control variates, the per-iteration cost does not grow with the dataset size (after the one-off $\mathcal{O}(N)$ preparation step).



Posterior approximation to the first marginal component $\theta^{(1)}$ for ULA, SGLD and SGLD-CV.

A general SGMCMC framework

The same subsampling trick applies to other diffusions with invariant distribution π:

Stochastic Gradient Hamiltonian Monte Carlo (Chen et al., 2014) – Part 2's (θ, p) pair with subsampled gradients (taking $\mathbf{M} = \mathbf{I}$):

$$\theta_{k+1} = \theta_k + h\, p_k,$$
$$p_{k+1} = (1 - \gamma h)\, p_k + h \widehat{\nabla \log \pi}(\theta_{k+1}) + \zeta_k, \qquad \zeta_k \sim \mathcal{N}\big(0,\, 2(\gamma - \hat{\beta}) h \mathbf{I}\big),$$

where the friction γ partially refreshes the momentum – replacing HMC's full refresh and accept-reject – and $\hat{\beta}$ estimates the gradient-noise variance.

A general SGMCMC framework

The same subsampling trick applies to other diffusions with invariant distribution π:

Stochastic Gradient Hamiltonian Monte Carlo (Chen et al., 2014) – Part 2's (θ, p) pair with subsampled gradients (taking $\mathbf{M} = \mathbf{I}$):

$$\theta_{k+1} = \theta_k + h\, p_k,$$
$$p_{k+1} = (1 - \gamma h)\, p_k + h\widehat{\nabla \log \pi}(\theta_{k+1}) + \zeta_k, \qquad \zeta_k \sim \mathcal{N}\big(0,\, 2(\gamma - \hat{\beta})h\mathbf{I}\big),$$

where the friction γ partially refreshes the momentum – replacing HMC's full refresh and accept-reject – and $\hat{\beta}$ estimates the gradient-noise variance.
Stochastic Gradient Nosé-Hoover thermostat (Ding et al., 2014): as SGHMC, but with an adaptive friction term that absorbs unknown gradient noise.

How to choose the step size

ULA and SGLD trade speed against bias through a single dial. Recall:

$$W_2(\tilde{\pi}_K, \pi) \lesssim \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{wants } h \text{ large}} + \underbrace{C_2 (hd)^{1/2} + C_3 \sigma (hd)^{1/2}}_{\text{wants } h \text{ small}}$$

How to choose the step size

ULA and SGLD trade speed against bias through a single dial. Recall:

$$W_2(\tilde{\pi}_K, \pi) \lesssim \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{wants } h \text{ large}} + \underbrace{C_2 (hd)^{1/2} + C_3 \sigma (hd)^{1/2}}_{\text{wants } h \text{ small}}$$

- No accept-reject means no feedback: MH samplers self-diagnose through their acceptance rate (0.234, 0.574, 0.45...) – unadjusted samplers emit no such signal.

How to choose the step size

ULA and SGLD trade speed against bias through a single dial. Recall:

$$W_2(\tilde{\pi}_K, \pi) \lesssim \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{wants } h \text{ large}} + \underbrace{C_2 (hd)^{1/2} + C_3 \sigma (hd)^{1/2}}_{\text{wants } h \text{ small}}$$

- No accept-reject means no feedback: MH samplers self-diagnose through their acceptance rate (0.234, 0.574, 0.45...) – unadjusted samplers emit no such signal.
- In practice: grid-search over h with repeated runs – expensive, and the best h changes with d, N and the scale of the target.

How to choose the step size

ULA and SGLD trade speed against bias through a single dial. Recall:

$$W_2(\tilde{\pi}_K, \pi) \lesssim \underbrace{(1 - C_1 h)^K W_2(\tilde{\pi}_0, \pi)}_{\text{wants } h \text{ large}} + \underbrace{C_2 (hd)^{1/2} + C_3 \sigma (hd)^{1/2}}_{\text{wants } h \text{ small}}$$

- No accept-reject means no feedback: MH samplers self-diagnose through their acceptance rate (0.234, 0.574, 0.45...) – unadjusted samplers emit no such signal.
- In practice: grid-search over h with repeated runs – expensive, and the best h changes with d, N and the scale of the target.

Can the sampler learn its own step size, while it runs?

Fuse: tuning-free step sizes

DEMO

ULA is gradient descent on $\mathrm{KL}(\mu\|\pi)$ over the space of probability measures, so optimisation theory gives an upper bound: after T steps,

$$\mathrm{KL} \lesssim \frac{D^2}{hT} + hG^2, \qquad \text{minimised by} \quad h^* = \frac{D}{G\sqrt{T}},$$

with $D = W_2(\mu_0, \pi)$ and G the gradient scale – both unknown.

Fuse: tuning-free step sizes

DEMO

ULA is gradient descent on $\mathrm{KL}(\mu\|\pi)$ over the space of probability measures, so optimisation theory gives an upper bound: after T steps,

$$\mathrm{KL} \lesssim \frac{D^2}{hT} + hG^2, \qquad \text{minimised by} \quad h^* = \frac{D}{G\sqrt{T}},$$

with $D = W_2(\mu_0, \pi)$ and G the gradient scale – both unknown. Fuse (Sharrock & Nemeth, 2025): plug in running estimates from n parallel chains,

$$h_t = \frac{\max_{s \le t} \widehat{W}_2(\mu_s, \mu_0)}{\sqrt{\sum_{s \le t} \widehat{\mathbb{E}}_{\mu_s} \|\nabla \log \pi\|^2}}$$

– distance over root cumulative gradients.

Fuse: tuning-free step sizes

DEMO

ULA is gradient descent on $\mathrm{KL}(\mu\|\pi)$ over the space of probability measures, so optimisation theory gives an upper bound: after T steps,

$$\mathrm{KL} \lesssim \frac{D^2}{hT} + hG^2, \qquad \text{minimised by} \quad h^* = \frac{D}{G\sqrt{T}},$$

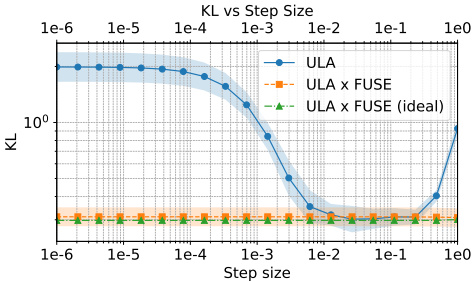
with $D = W_2(\mu_0, \pi)$ and G the gradient scale – both unknown. Fuse (Sharrock & Nemeth, 2025): plug in running estimates from n parallel chains,

$$h_t = \frac{\max_{s \le t} \widehat{W}_2(\mu_s, \mu_0)}{\sqrt{\sum_{s \le t} \widehat{\mathbb{E}}_{\mu_s} \|\nabla \log \pi\|^2}}$$ – distance over root cumulative gradients.

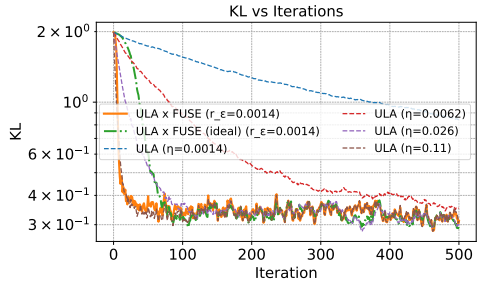
- Transient: numerator grows $\Rightarrow$ bold steps; stationarity: numerator plateaus $\Rightarrow h_t \sim 1/\sqrt{t}$, the classical optimal decay – automatically.
- Matches the optimally tuned rate up to log factors; same recipe tunes SGLD, SVGD, MFLD, ...

Fuse in action

ULA vs ULA$\times$Fuse on a 10-dimensional Gaussian target:



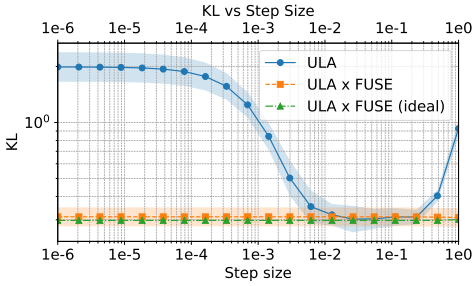
KL after T steps, against the fixed step size h (ULA) or the initialisation r_ε (Fuse)

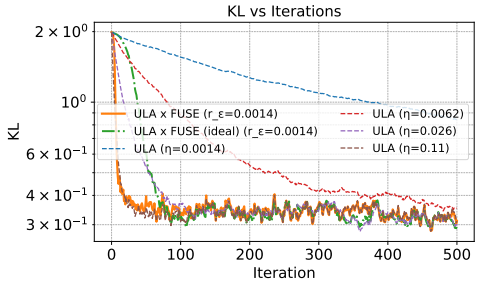


KL against iterations: Fuse tracks the best fixed-h run

Fuse in action

ULA vs ULA$\times$Fuse on a 10-dimensional Gaussian target:





KL after T steps, against the fixed step size h (ULA) or the initialisation r_ε (Fuse)

KL against iterations: Fuse tracks the best fixed-h run

Fuse is flat – one run, near-optimal, for any r_ε across eight orders of magnitude.

Aside: Continuous-time samplers

DEMO

Piecewise deterministic Markov processes (PDMPs: Zig-Zag, Bouncy Particle Sampler) glide with velocity v and change direction at random event times, at rate

$$\lambda(\theta, v) = \big[-v \cdot \nabla \log \pi(\theta) \big]_+$$

(the BPS: the $\delta \to 0$ limit of Part 2's dBPS).

Aside: Continuous-time samplers

DEMO

Piecewise deterministic Markov processes (PDMPs: Zig-Zag, Bouncy Particle Sampler) glide with velocity v and change direction at random event times, at rate

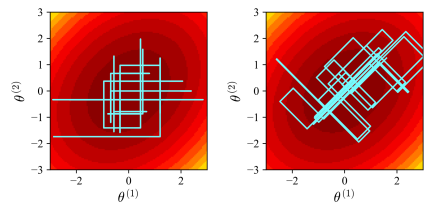
$$\lambda(\theta, v) = \big[-v \cdot \nabla \log \pi(\theta) \big]_+$$

(the BPS: the $\delta \to 0$ limit of Part 2's dBPS).

Invariance for π constrains only the difference of rates, and $[a]_+ - [-a]_+ = a$, so

$$\lambda(\theta, v) - \lambda(\theta, -v) = -v \cdot \nabla \log \pi(\theta)$$

survives replacing $\nabla \log \pi$ by any unbiased estimate – in particular the SGLD-CV estimator $\widehat{\nabla \log \pi}$ from earlier.



Aside: Continuous-time samplers

DEMO

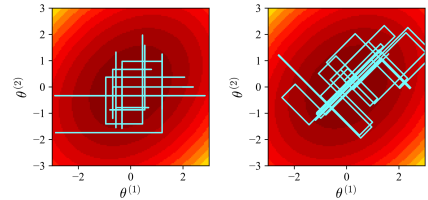
Piecewise deterministic Markov processes (PDMPs: Zig-Zag, Bouncy Particle Sampler) glide with velocity v and change direction at random event times, at rate

$$\lambda(\theta, v) = \big[-v \cdot \nabla \log \pi(\theta) \big]_+$$ (the BPS: the $\delta \to 0$ limit of Part 2's dBPS).

Invariance for π constrains only the difference of rates, and $[a]_+ - [-a]_+ = a$, so

$$\lambda(\theta, v) - \lambda(\theta, -v) = -v \cdot \nabla \log \pi(\theta)$$

survives replacing $\nabla \log \pi$ by any unbiased estimate – in particular the SGLD-CV estimator $\widehat{\nabla \log \pi}$ from earlier.



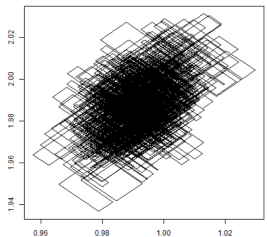
Gradient noise adds only extra events – never bias. Control variates make the excess vanish near the mode: exact, ballistic, $\mathcal{O}(1)$ data per event.

Subsampling in continuous time: control variates again

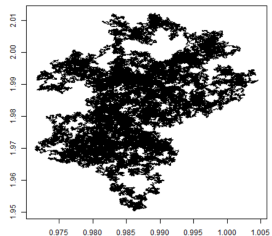
Zig-Zag on a logistic regression posterior, in stationarity:

Subsampling in continuous time: control variates again

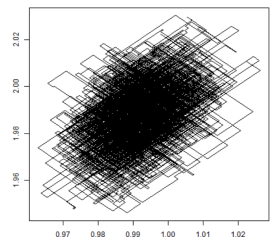
Zig-Zag on a logistic regression posterior, in stationarity:



ZZ (full data): ballistic



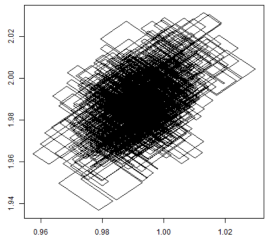
ZZ-SS: diffusive again!



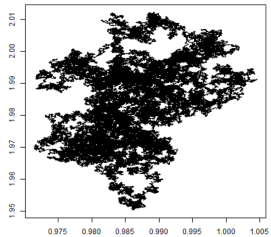
ZZ-CV: ballistic restored

Subsampling in continuous time: control variates again

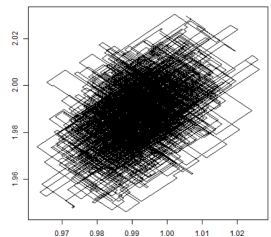
Zig-Zag on a logistic regression posterior, in stationarity:



ZZ (full data): ballistic



ZZ-SS: diffusive again!



ZZ-CV: ballistic restored

Naive subsampling is still exact, but re-introduces the diffusivity we worked to remove. Control variates around the mode restore ballistic motion: the same convergence rate as full-data Zig-Zag, at cost reduced by a factor $\mathcal{O}(N)$. (Bierkens, Fearnhead & Roberts, 2019; Agrawal, Bierkens & Roberts, 2024; figure: Bierkens, 2026.)

Part 3: Checkpoint

- Dropping the MH correction (ULA) and subsampling the gradient (SGLD) reduces the per-iteration cost from $\mathcal{O}(N)$ to $\mathcal{O}(n)$.

Part 3: Checkpoint

- Dropping the MH correction (ULA) and subsampling the gradient (SGLD) reduces the per-iteration cost from $\mathcal{O}(N)$ to $\mathcal{O}(n)$.
- The price is bias: discretisation error + gradient noise, controlled by the step size.

Part 3: Checkpoint

- Dropping the MH correction (ULA) and subsampling the gradient (SGLD) reduces the per-iteration cost from $\mathcal{O}(N)$ to $\mathcal{O}(n)$.
- The price is bias: discretisation error + gradient noise, controlled by the step size.
- Control variates make the gradient noise – and hence the cost for fixed accuracy – essentially independent of N.

Part 3: Checkpoint

- Dropping the MH correction (ULA) and subsampling the gradient (SGLD) reduces the per-iteration cost from $\mathcal{O}(N)$ to $\mathcal{O}(n)$.
- The price is bias: discretisation error + gradient noise, controlled by the step size.
- Control variates make the gradient noise – and hence the cost for fixed accuracy – essentially independent of N.

Next: do we really have to give up exactness to gain scalability?

Part 4

Exact MCMC with subsampling: Metropolis-Hastings with Scalable Subsampling

Do we have to accept bias?

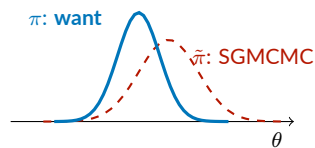
Stochastic gradient methods are approximate: the invariant distribution is not π, and the bias is hard to quantify in practice.

Do we have to accept bias?

Stochastic gradient methods are approximate: the invariant distribution is not π, and the bias is hard to quantify in practice.

Goal of this part: an MCMC algorithm that

- evaluates only a small random subsample of the data per iteration, and
- satisfies detailed balance with respect to the true posterior $\pi(\theta)$.

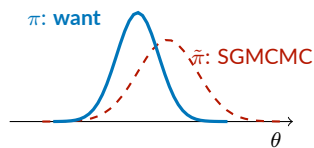


Do we have to accept bias?

Stochastic gradient methods are approximate: the invariant distribution is not π, and the bias is hard to quantify in practice.

Goal of this part: an MCMC algorithm that

- evaluates only a small random subsample of the data per iteration, and
- satisfies detailed balance with respect to the true posterior $\pi(\theta)$.



Prado, Nemeth & Sherlock (2026). Metropolis-Hastings with Scalable Subsampling. *JASA (to appear).*

Reminder: Metropolis–Hastings

Propose $\theta' \sim q(\cdot|\theta)$ and accept with probability

$$\alpha(\theta, \theta') = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \times \frac{p(\mathbf{y}|\theta')}{p(\mathbf{y}|\theta)} \right]$$

Reminder: Metropolis–Hastings

Propose $\theta' \sim q(\cdot|\theta)$ and accept with probability

$$\alpha(\theta, \theta') = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \times \frac{p(\mathbf{y}|\theta')}{p(\mathbf{y}|\theta)} \right]$$

On the log scale, the likelihood ratio is a sum of N small terms:

$$\log \frac{p(\mathbf{y}|\theta')}{p(\mathbf{y}|\theta)} = \sum_{i=1}^{N} \left[\ell_i(\theta') - \ell_i(\theta) \right], \qquad \ell_i(\theta) := \log p(y_i|\theta)$$

Reminder: Metropolis–Hastings

Propose $\theta' \sim q(\cdot|\theta)$ and accept with probability

$$\alpha(\theta, \theta') = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \times \frac{p(\mathbf{y}|\theta')}{p(\mathbf{y}|\theta)} \right]$$

On the log scale, the likelihood ratio is a sum of N small terms:

$$\log \frac{p(\mathbf{y}|\theta')}{p(\mathbf{y}|\theta)} = \sum_{i=1}^{N} \left[\ell_i(\theta') - \ell_i(\theta) \right], \qquad \ell_i(\theta) := \log p(y_i|\theta)$$

Everything expensive lives in this sum. This part: estimate it from a subsample without breaking detailed balance.

Why naive subsampling fails

Could we just estimate the log-likelihood ratio from a subsample?

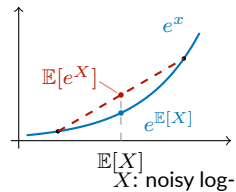
$$\sum_{i=1}^{N} [\ell_i(\theta') - \ell_i(\theta)] \approx \frac{N}{n} \sum_{i \in S} [\ell_i(\theta') - \ell_i(\theta)]$$

Why naive subsampling fails

Could we just estimate the log-likelihood ratio from a subsample?

$$\sum_{i=1}^{N} [\ell_i(\theta') - \ell_i(\theta)] \approx \frac{N}{n} \sum_{i \in S} [\ell_i(\theta') - \ell_i(\theta)]$$

- Unbiased on the log scale, but α needs $\exp(\cdot)$ – and exponentiation destroys unbiasedness (Jensen).
- Pseudo-marginal MCMC needs a non-negative unbiased likelihood estimate: hard from subsamples.
- Plugging the noisy ratio in anyway: the invariant distribution is not π (Bardenet et al., 2017).



noise + convexity ⇒ upward bias

Exact subsampling MH: prior art

- Firefly Monte Carlo (Maclaurin & Adams, 2014): requires strictly positive lower bounds on each likelihood term; augments with N Bernoulli latent variables. Mixing can be very slow.

Exact subsampling MH: prior art

- Firefly Monte Carlo (Maclaurin & Adams, 2014): requires strictly positive lower bounds on each likelihood term; augments with N Bernoulli latent variables. Mixing can be very slow.
- Scalable MH (SMH; Cornish et al., 2019): factorised acceptance rule + control variates; exact, but the bounds on the log-likelihood differences are loose in moderate-to-high dimension.

Exact subsampling MH: prior art

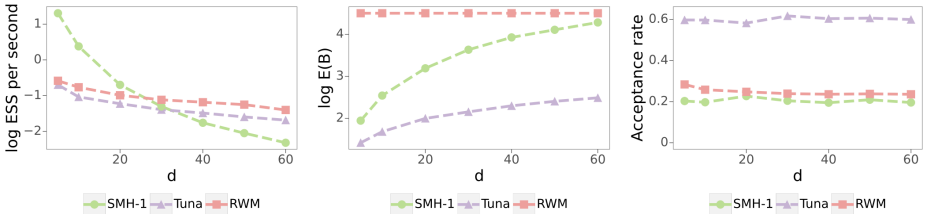
- Firefly Monte Carlo (Maclaurin & Adams, 2014): requires strictly positive lower bounds on each likelihood term; augments with N Bernoulli latent variables. Mixing can be very slow.
- Scalable MH (SMH; Cornish et al., 2019): factorised acceptance rule + control variates; exact, but the bounds on the log-likelihood differences are loose in moderate-to-high dimension.
- TunaMH (Zhang et al., 2020): stabilises the acceptance rate with an extra tuning parameter, but requires such small proposal steps that it barely beats optimally tuned RWM.

Exact subsampling MH: prior art

- Firefly Monte Carlo (Maclaurin & Adams, 2014): requires strictly positive lower bounds on each likelihood term; augments with N Bernoulli latent variables. Mixing can be very slow.
- Scalable MH (SMH; Cornish et al., 2019): factorised acceptance rule + control variates; exact, but the bounds on the log-likelihood differences are loose in moderate-to-high dimension.
- TunaMH (Zhang et al., 2020): stabilises the acceptance rate with an extra tuning parameter, but requires such small proposal steps that it barely beats optimally tuned RWM.

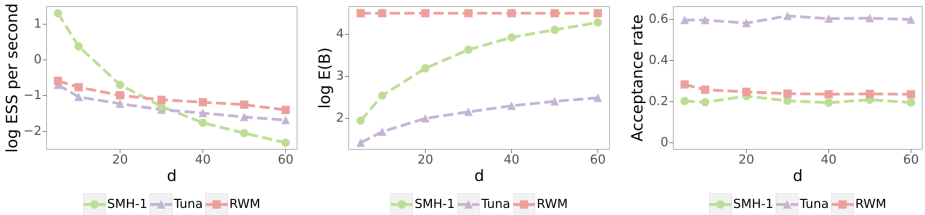
Common ingredients: bounds on likelihood terms + auxiliary variables. The quality of the bounds determines everything.

Prior art: the evidence



ESS/sec, $\log \mathbb{E}[B]$ and acceptance rate against dimension d for SMH-1, TunaMH and RWM (logistic regression, $N \approx 30{,}000$).

Prior art: the evidence



ESS/sec, $\log \mathbb{E}[B]$ and acceptance rate against dimension d for SMH-1, TunaMH and RWM (logistic regression, $N \approx 30{,}000$).

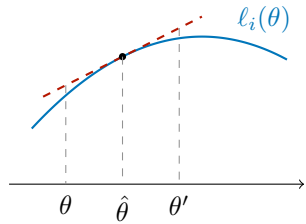
TunaMH holds its acceptance rate only by taking tiny steps, so its efficiency tracks RWM; SMH's loose bounds make $\mathbb{E}[B]$ blow up with d. Neither convincingly beats well-tuned RWM.

Step 1: predict each term for free (control variates)

As in SGLD-CV, let $\hat{\theta}$ be (close to) the posterior mode.

Predict each difference with a first-order Taylor expansion about $\hat{\theta}$:

$$\ell_i(\theta') - \ell_i(\theta) \approx r_i(\theta, \theta')$$
$$r_i(\theta, \theta') := (\theta' - \theta)^\top \nabla \ell_i(\hat{\theta})$$



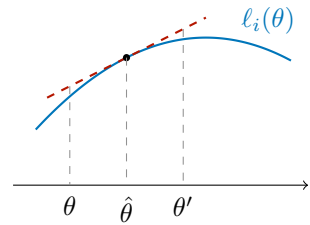
the tangent at $\hat{\theta}$ predicts the change

Step 1: predict each term for free (control variates)

As in SGLD-CV, let $\hat{\theta}$ be (close to) the posterior mode.

Predict each difference with a first-order Taylor expansion about $\hat{\theta}$:

$$\ell_i(\theta') - \ell_i(\theta) \approx r_i(\theta, \theta')$$
$$r_i(\theta, \theta') := (\theta' - \theta)^\top \nabla \ell_i(\hat{\theta})$$



the tangent at $\hat{\theta}$ predicts the change

The total prediction

$$\sum_{i=1}^{N} r_i(\theta, \theta') = (\theta' - \theta)^\top \textstyle\sum_{i=1}^{N} \nabla \ell_i(\hat{\theta})$$

needs only a precomputed d-vector: cost $\mathcal{O}(d)$ per iteration, no data access.

How good is the prediction?

Define the prediction error for datum i:

$$\Delta_i := r_i(\theta, \theta') - [\ell_i(\theta') - \ell_i(\theta)] \,.$$

How good is the prediction?

Define the prediction error for datum i:

$$\Delta_i := r_i(\theta, \theta') - [\ell_i(\theta') - \ell_i(\theta)]\,.$$

The key requirement: computable constants $c_i > 0$ and a symmetric function $M(\theta, \theta')$ with

$$|\Delta_i| \leq c_i M(\theta, \theta').$$

How good is the prediction?

Define the prediction error for datum i:

$$\Delta_i := r_i(\theta, \theta') - [\ell_i(\theta') - \ell_i(\theta)]\,.$$

The key requirement: computable constants $c_i > 0$ and a symmetric function $M(\theta, \theta')$ with

$$|\Delta_i| \le c_i M(\theta, \theta').$$

- Writing $\ell_i(\theta) = h(x_i^\top \theta;\, y_i)$ for GLMs: $c_i = \|x_i\|^2 K(y_i)$, with $K(y_i) = \sup_\eta |h''(\eta; y_i)|$ the max curvature in the linear predictor (logistic: $K = \frac{1}{4}$).

How good is the prediction?

Define the prediction error for datum i:

$$\Delta_i := r_i(\theta, \theta') - [\ell_i(\theta') - \ell_i(\theta)]\,.$$

The key requirement: computable constants $c_i > 0$ and a symmetric function $M(\theta, \theta')$ with

$$|\Delta_i| \leq c_i M(\theta, \theta').$$

- Writing $\ell_i(\theta) = h(x_i^\top \theta;\, y_i)$ for GLMs: $c_i = \|x_i\|^2 K(y_i)$, with $K(y_i) = \sup_\eta |h''(\eta; y_i)|$ the max curvature in the linear predictor (logistic: $K = \frac{1}{4}$).
- $M(\theta, \theta') \approx \|\theta' - \theta\| \max\{\|\theta - \hat{\theta}\|, \|\theta' - \hat{\theta}\|\}$: (step size) $\times$ (distance from the mode) – small for local moves near the mode.

How good is the prediction?

Define the prediction error for datum i:

$$\Delta_i := r_i(\theta, \theta') - [\ell_i(\theta') - \ell_i(\theta)]\,.$$

The key requirement: computable constants $c_i > 0$ and a symmetric function $M(\theta, \theta')$ with

$$|\Delta_i| \le c_i M(\theta, \theta').$$

- Writing $\ell_i(\theta) = h(x_i^\top \theta;\, y_i)$ for GLMs: $c_i = \|x_i\|^2 K(y_i)$, with $K(y_i) = \sup_\eta |h''(\eta; y_i)|$ the max curvature in the linear predictor (logistic: $K = \frac{1}{4}$).
- $M(\theta, \theta') \approx \|\theta' - \theta\| \max\{\|\theta - \hat\theta\|, \|\theta' - \hat\theta\|\}$: (step size) $\times$ (distance from the mode) – small for local moves near the mode.
- Second-order version (CV2): keep the Hessian term of the Taylor expansion too – tighter bound, at an $\mathcal{O}(d^2)$ precomputation cost.

Step 2: what is still missing from the ratio?

Rearrange the exact MH acceptance probability around the prediction (recall $\ell_i(\theta') - \ell_i(\theta) = r_i - \Delta_i$):

$$\alpha(\theta, \theta') = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \underbrace{\exp\left\{ \sum_{i=1}^{N} r_i(\theta, \theta') \right\}}_{\text{prediction: free, no data access}} \underbrace{\exp\left\{ -\sum_{i=1}^{N} \Delta_i \right\}}_{\text{missing: needs every } \ell_i} \right]$$

Step 2: what is still missing from the ratio?

Rearrange the exact MH acceptance probability around the prediction (recall $\ell_i(\theta') - \ell_i(\theta) = r_i - \Delta_i$):

$$\alpha(\theta, \theta') = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \underbrace{\exp\left\{ \sum_{i=1}^{N} r_i(\theta, \theta') \right\}}_{\text{prediction: free, no data access}} \underbrace{\exp\left\{ -\sum_{i=1}^{N} \Delta_i \right\}}_{\text{missing: needs every } \ell_i} \right]$$

Everything we cannot afford is concentrated in the factors $e^{-\Delta_i}$ – and all we know cheaply is $|\Delta_i| \leq c_i M(\theta, \theta')$.

Step 2: what is still missing from the ratio?

Rearrange the exact MH acceptance probability around the prediction (recall $\ell_i(\theta') - \ell_i(\theta) = r_i - \Delta_i$):

$$\alpha(\theta, \theta') = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \underbrace{\exp\left\{ \sum_{i=1}^{N} r_i(\theta, \theta') \right\}}_{\text{prediction: free, no data access}} \underbrace{\exp\left\{ -\sum_{i=1}^{N} \Delta_i \right\}}_{\text{missing: needs every } \ell_i} \right]$$

Everything we cannot afford is concentrated in the factors $e^{-\Delta_i}$ – and all we know cheaply is $|\Delta_i| \leq c_i M(\theta, \theta')$.

Idea: we cannot compute $e^{-\Delta_i}$ – but we can simulate a random stand-in with exactly this expectation, and stay exact (randomised MH; Nicholls et al., 2012).

Poisson powers average to exponentials

For $S \sim \text{Poisson}(\phi)$: $\mathbb{E}[t^S] = e^{\phi(t-1)}$ – the only count distribution whose $\mathbb{E}[t^S]$ is an exponential (a Bernoulli's is linear in t: hence Firefly's different machinery).

Poisson powers average to exponentials

For $S \sim \text{Poisson}(\phi)$: $\mathbb{E}[t^S] = e^{\phi(t-1)}$ – the only count distribution whose $\mathbb{E}[t^S]$ is an exponential (a Bernoulli's is linear in t: hence Firefly's different machinery).

So give each datum an audit count $S_i \sim \text{Poisson}(\phi_i)$, with audit rates $\phi_i, \phi_i' \in [0,\ c_i M(\theta, \theta')]$ (for the move and its reverse) chosen so that $\phi_i - \phi_i' = \Delta_i$:

$$\mathbb{E}\left[(\phi_i'/\phi_i)^{S_i}\right] = e^{\phi_i' - \phi_i} = e^{-\Delta_i} \quad \text{– the missing factor, in expectation.}$$

Poisson powers average to exponentials

For $S \sim \text{Poisson}(\phi)$: $\mathbb{E}[t^S] = e^{\phi(t-1)}$ – the only count distribution whose $\mathbb{E}[t^S]$ is an exponential (a Bernoulli's is linear in t: hence Firefly's different machinery).

So give each datum an audit count $S_i \sim \text{Poisson}(\phi_i)$, with audit rates $\phi_i, \phi_i' \in [0,\ c_i M(\theta, \theta')]$ (for the move and its reverse) chosen so that $\phi_i - \phi_i' = \Delta_i$:

$$\mathbb{E}\left[(\phi_i'/\phi_i)^{S_i}\right] = e^{\phi_i' - \phi_i} = e^{-\Delta_i} \quad \text{– the missing factor, in expectation.}$$

- $\phi_i \leq c_i M \ll 1$: almost all $S_i = 0$ – factor $(\phi_i'/\phi_i)^0 = 1$, datum never touched.
- Larger possible error $c_i M \Rightarrow$ higher audit rate: checked more often where the prediction could be more wrong. $S_i > 0$: evaluate $\ell_i(\theta), \ell_i(\theta')$.

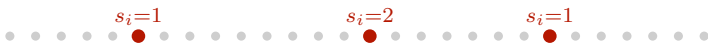
Step 2, assembled: audit and accept

Step 2, assembled: audit and accept

Replace the missing exponential by the audit factors – accept θ' with probability

$$\alpha(\theta, \theta'; s) = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \underbrace{\exp\left\{ \sum_{i=1}^{N} r_i(\theta, \theta') \right\}}_{\text{prediction (free)}} \underbrace{\prod_{i:\, s_i>0} \left(\frac{\phi_i'}{\phi_i} \right)^{s_i}}_{\text{correction (audited data only)}} \right]$$

Step 2, assembled: audit and accept



Replace the missing exponential by the audit factors – accept θ' with probability

$$\alpha(\theta, \theta'; s) = 1 \wedge \left[\frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)} \underbrace{\exp\left\{ \sum_{i=1}^{N} r_i(\theta, \theta') \right\}}_{\text{prediction (free)}} \underbrace{\prod_{i:\, s_i > 0} \left(\frac{\phi_i'}{\phi_i} \right)^{s_i}}_{\text{correction (audited data only)}} \right]$$

Only the handful of data with $s_i > 0$ ever enter – and each factor is the unbiased stand-in for its $e^{-\Delta_i}$.

Why is this exact?

- In the joint chain on (θ, s), the ratio of Poisson pmfs for the move and its reverse contributes $e^{\Delta_i}(\phi_i'/\phi_i)^{s_i}$ – the intractable e^{Δ_i} cancels algebraically: an instance of randomised MH (Nicholls et al., 2012).

Why is this exact?

- In the joint chain on (θ, s), the ratio of Poisson pmfs for the move and its reverse contributes $e^{\Delta_i}(\phi_i'/\phi_i)^{s_i}$ – the intractable e^{Δ_i} cancels algebraically: an instance of randomised MH (Nicholls et al., 2012).
- Note: each audit factor can be < 1 or > 1 – it corrects the prediction in whichever direction it erred.

Why is this exact?

- In the joint chain on (θ, s), the ratio of Poisson pmfs for the move and its reverse contributes $e^{\Delta_i}(\phi_i'/\phi_i)^{s_i}$ – the intractable e^{Δ_i} cancels algebraically: an instance of randomised MH (Nicholls et al., 2012).
- Note: each audit factor can be < 1 or > 1 – it corrects the prediction in whichever direction it erred.
- **Proposition:** the chain satisfies detailed balance with respect to the true posterior $\pi(\theta)$. Exactness for free!

Simulating all N audits in $\mathcal{O}(B)$

Drawing N Poisson counts each iteration would cost $\mathcal{O}(N)$ – superposition + thinning works top-down instead:

Simulating all N audits in $\mathcal{O}(B)$

Drawing N Poisson counts each iteration would cost $\mathcal{O}(N)$ – superposition + thinning works top-down instead:

1. How many? Sums of Poissons are Poisson, so using the computable bounding rates $c_i M \geq \phi_i$: draw one total, $B \sim \text{Poisson}(CM(\theta, \theta'))$, $C = \sum_i c_i$.

Simulating all N audits in $\mathcal{O}(B)$

Drawing N Poisson counts each iteration would cost $\mathcal{O}(N)$ – superposition + thinning works top-down instead:

1. How many? Sums of Poissons are Poisson, so using the computable bounding rates $c_i M \geq \phi_i$: draw one total, $B \sim \text{Poisson}(CM(\theta, \theta'))$, $C = \sum_i c_i$.
2. Who? Given the total, candidate audits land on data independently, hitting datum i w.p. $c_i M / CM = c_i / C$ – one precomputed alias-table lookup each.

Simulating all N audits in $\mathcal{O}(B)$

Drawing N Poisson counts each iteration would cost $\mathcal{O}(N)$ – superposition + thinning works top-down instead:

1. How many? Sums of Poissons are Poisson, so using the computable bounding rates $c_i M \geq \phi_i$: draw one total, $B \sim \text{Poisson}(CM(\theta, \theta'))$, $C = \sum_i c_i$.
2. Who? Given the total, candidate audits land on data independently, hitting datum i w.p. $c_i M / CM = c_i / C$ – one precomputed alias-table lookup each.
3. Thin: keep a candidate audit of datum i w.p. $\phi_i / (c_i M)$ – the ratio of true to bounding rate at that datum (only now is ℓ_i evaluated).

Simulating all N audits in $\mathcal{O}(B)$

Drawing N Poisson counts each iteration would cost $\mathcal{O}(N)$ – superposition + thinning works top-down instead:

1. How many? Sums of Poissons are Poisson, so using the computable bounding rates $c_i M \geq \phi_i$: draw one total, $B \sim \text{Poisson}(CM(\theta, \theta'))$, $C = \sum_i c_i$.
2. Who? Given the total, candidate audits land on data independently, hitting datum i w.p. $c_i M / CM = c_i / C$ – one precomputed alias-table lookup each.
3. Thin: keep a candidate audit of datum i w.p. $\phi_i / (c_i M)$ – the ratio of true to bounding rate at that datum (only now is ℓ_i evaluated).

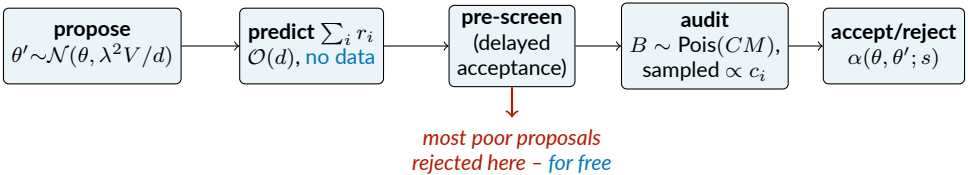
The kept counts are exactly independent $\text{Poisson}(\phi_i)$. Expected work: $\mathbb{E}[B] = CM(\theta, \theta')$ – small when the bounds are tight, proposals local, the chain near the mode. (The same thinning trick that simulates PDMP event times.)

MH-SS: the algorithm

One-off setup ($\mathcal{O}(N)$): find $\hat{\theta}$; precompute the $\nabla \ell_i(\hat{\theta})$ and their sum, the c_i and $C = \sum_i c_i$.

MH-SS: the algorithm

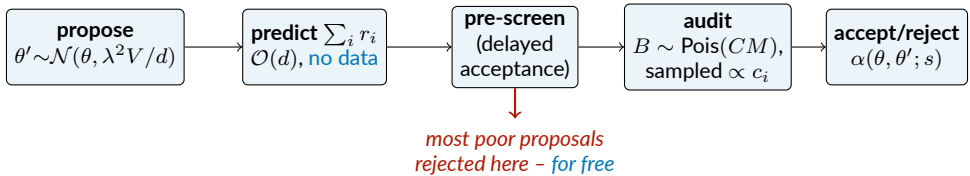
One-off setup ($\mathcal{O}(N)$): find $\hat{\theta}$; precompute the $\nabla \ell_i(\hat{\theta})$ and their sum, the c_i and $C = \sum_i c_i$. **Each iteration:**



V: estimate of the posterior covariance; λ: tuning parameter.

MH-SS: the algorithm

One-off setup ($\mathcal{O}(N)$): find $\hat{\theta}$; precompute the $\nabla \ell_i(\hat{\theta})$ and their sum, the c_i and $C = \sum_i c_i$. **Each iteration:**



V: estimate of the posterior covariance; λ: tuning parameter.

- Only the audited data (B points, typically $\ll N$) are ever touched – and the chain is still exact.
- The pre-screen is valid delayed acceptance: $1 \wedge \rho_1\rho_2 \to (1 \wedge \rho_1)(1 \wedge \rho_2)$ keeps detailed balance (Christen & Fox, 2005).
- If $CM(\theta, \theta') \geq N$, just do a full-data MH step.

MH-SS: theoretical results

- **Optimality:** the MH-SS choice of weights ϕ_i maximises the acceptance rate – and hence the ESS – at no extra cost.

MH-SS: theoretical results

- **Optimality:** the MH-SS choice of weights ϕ_i maximises the acceptance rate – and hence the ESS – at no extra cost.
- **Tighter bounds:** for regression models the bounds are at least a factor $d^{1/2}$ tighter than those of SMH – much smaller subsamples.

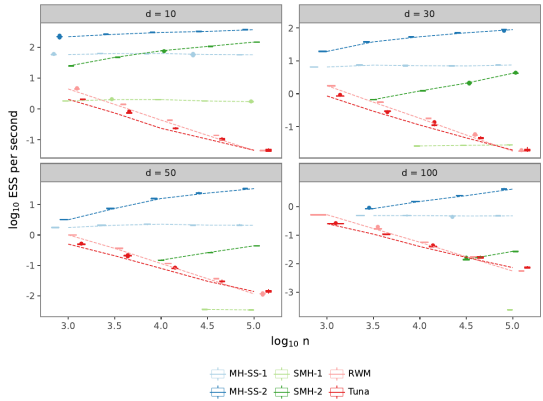
MH-SS: theoretical results

- **Optimality:** the MH-SS choice of weights ϕ_i maximises the acceptance rate – and hence the ESS – at no extra cost.
- **Tighter bounds:** for regression models the bounds are at least a factor $d^{1/2}$ tighter than those of SMH – much smaller subsamples.
- **Optimal scaling:** audits get dearer as steps grow ($\mathbb{E}[B] \propto \lambda$), so maximise ESJD per unit of compute $\propto \lambda\, a(\lambda)$: acceptance ≈ 0.45, not 0.234 – costlier jumps $\Rightarrow$ shorter, surer steps.

MH-SS: theoretical results

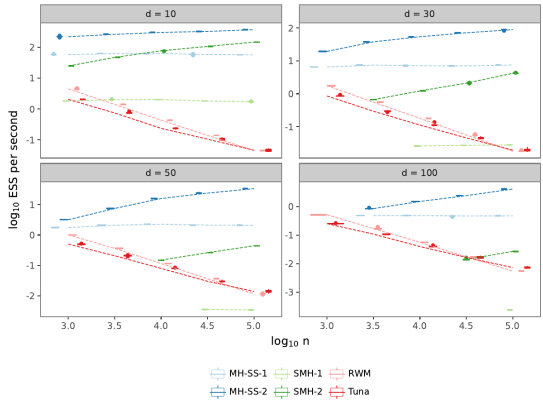
- **Optimality:** the MH-SS choice of weights ϕ_i maximises the acceptance rate – and hence the ESS – at no extra cost.
- **Tighter bounds:** for regression models the bounds are at least a factor $d^{1/2}$ tighter than those of SMH – much smaller subsamples.
- **Optimal scaling:** audits get dearer as steps grow ($\mathbb{E}[B] \propto \lambda$), so maximise ESJD per unit of compute $\propto \lambda\, a(\lambda)$: acceptance ≈ 0.45, not 0.234 – costlier jumps $\Rightarrow$ shorter, surer steps.
- **Cost per iteration:** $\mathcal{O}(d)$ (CV1) or $\mathcal{O}(d^2)$ (CV2) for the control variates, plus the subsample. With CV2, $\mathbb{E}[B]$ decreases with N.

MH-SS: how it compares



ESS per second against data size (log–log scale) for logistic regression, $d = 10$ to 100.

MH-SS: how it compares



ESS per second against data size (log–log scale) for logistic regression, $d = 10$ to 100.

MH-SS's efficiency grows with the data size (concentration $\Rightarrow$ smaller audits), while RWM and TunaMH decay at rate $1/N$; SMH's looser bounds cost it as d grows.

MH-SS: takeaways and limitations

Takeaways:

- Exactness and subsampling can coexist: detailed balance w.r.t. the true posterior.
- The engine is the same as in Part 3: control variates around the mode + posterior concentration.

MH-SS: takeaways and limitations

Takeaways:

- Exactness and subsampling can coexist: detailed balance w.r.t. the true posterior.
- The engine is the same as in Part 3: control variates around the mode + posterior concentration.

Limitations:

- Needs computable bounds $c_i M(\theta, \theta')$: available for GLM-type models, model-by-model work otherwise.
- Needs a good $\hat{\theta}$ (one optimisation pass) – as do all control-variate methods.
- Random-walk proposals: dimension scaling is that of RWM, so best suited to large N, moderate d.

The scalable MCMC landscape

Method	Exact?	Cost per iteration	Scaling regime
RWM / MALA / HMC	✓	$\mathcal{O}(N)$	small N, d up to large
ULA	$\times$	$\mathcal{O}(N)$	gradient-only, no MH step
SGLD	$\times$	$\mathcal{O}(n)$	huge N, rough accuracy
SGLD-CV	$\times$	$\mathcal{O}(1)^*$	huge N, better accuracy
MH-SS	✓	$\mathcal{O}(d) + \mathbb{E}[B]$	large N, moderate d
PDMP + subsampling	✓	$\mathcal{O}(1)^*$	non-reversible, active research

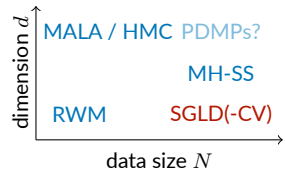
* after an $\mathcal{O}(N)$ preparation step.

The scalable MCMC landscape

Method	Exact?	Cost per iteration	Scaling regime
RWM / MALA / HMC	✓	$\mathcal{O}(N)$	small N, d up to large
ULA	×	$\mathcal{O}(N)$	gradient-only, no MH step
SGLD	×	$\mathcal{O}(n)$	huge N, rough accuracy
SGLD-CV	×	$\mathcal{O}(1)^*$	huge N, better accuracy
MH-SS	✓	$\mathcal{O}(d) + \mathbb{E}[B]$	large N, moderate d
PDMP + subsampling	✓	$\mathcal{O}(1)^*$	non-reversible, active research

* after an $\mathcal{O}(N)$ preparation step.

Recurring theme: posterior concentration + control variates around the mode is what makes subsampling work – with or without exactness.



Open problems

- Non-IID data: time series, spatial data, networks – subsampling assumes exchangeable likelihood contributions.

Open problems

- Non-IID data: time series, spatial data, networks – subsampling assumes exchangeable likelihood contributions.
- High dimension and big data together: MH-SS handles big N; combining exact subsampling with gradient-based proposals is open.

Open problems

- Non-IID data: time series, spatial data, networks – subsampling assumes exchangeable likelihood contributions.
- High dimension and big data together: MH-SS handles big N; combining exact subsampling with gradient-based proposals is open.
- Automatic tuning of stochastic gradient MCMC (step sizes, subsample sizes).

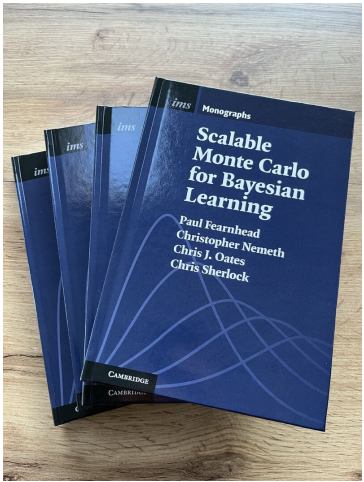
Open problems

- Non-IID data: time series, spatial data, networks – subsampling assumes exchangeable likelihood contributions.
- High dimension and big data together: MH-SS handles big N; combining exact subsampling with gradient-based proposals is open.
- Automatic tuning of stochastic gradient MCMC (step sizes, subsample sizes).
- Diagnostics for approximate MCMC: how do we know how wrong we are?

Open problems

- Non-IID data: time series, spatial data, networks – subsampling assumes exchangeable likelihood contributions.
- High dimension and big data together: MH-SS handles big N; combining exact subsampling with gradient-based proposals is open.
- Automatic tuning of stochastic gradient MCMC (step sizes, subsample sizes).
- Diagnostics for approximate MCMC: how do we know how wrong we are?
- Hybrids with ML: normalising flows and diffusion-based proposals for multimodal targets.

Want to learn more?



Fearnhead, Nemeth, Oates and Sherlock (2025).
Scalable Monte Carlo for Bayesian Learning.
Cambridge University Press.

Free preprint: arXiv:2407.12751

Questions?

c.nemeth@lancaster.ac.uk

Explore the samplers from this course interactively:
https://chris-nemeth.github.io/sampling-viz/