From Drift-Diffusion Processes to Monte Carlo Sampling

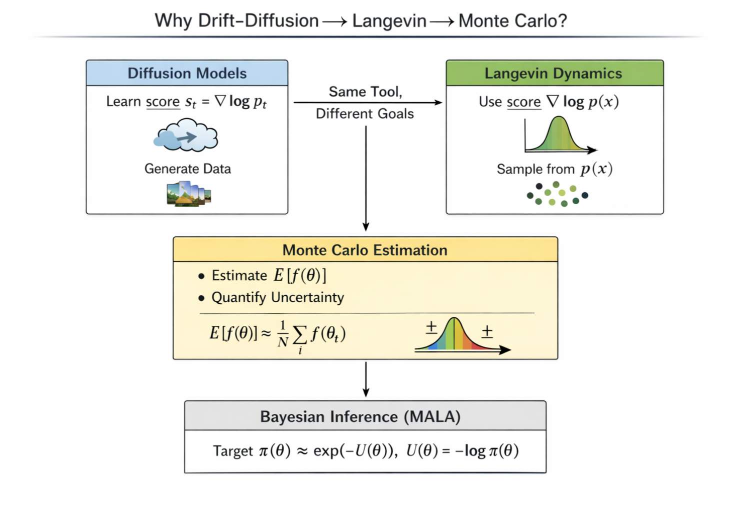
ELG 5218 - Uncertainty Evaluation in Engineering Measurements and Machine Learning

Miodrag Bolic

University of Ottawa

February 1, 2026

Why Drift–Diffusion → Langevin → Monte Carlo?


Why Drift-Diffusion ⟶ Langevin ⟶ Monte Carlo?
Diffusion Models
Learn score $s_t = \nabla \log p_t$
Generate Data
Same Tool,
Different Goals
Langevin Dynamics
Use score $\nabla \log p(x)$
Sample from $p(x)$
Monte Carlo Estimation
• Estimate $E[f(\theta)]$
• Quantify Uncertainty
$E[f(\theta)] \approx \frac{1}{N}\sum_i f(\theta_t)$
±
±
Bayesian Inference (MALA)
Target $\pi(\theta) \approx \exp(-U(\theta)),\ U(\theta) = -\log \pi(\theta)$

The notebook is langevin_MALA.ipynb

We will cover:

1. Drift–diffusion processes in one dimension:
   - Step 1.1. Pure diffusion (no drift)
   - Step 1.2. Pure drift (no diffusion)
   - Step 1.3. Drift $+$ diffusion together
   - Step 1.4. Langevin diffusion view (drift from a potential)
   - Step 1.5. Monte Carlo / law-of-large-numbers view via many trajectories
2. From Langevin to MALA
   - Step 2.1. Understand potential energy: $U(\theta) = -\log \pi(\theta)$
   - Step 2.2. Learn Langevin dynamics and why it samples from $\pi(\theta)$
   - Step 2.3. Understand discretization bias in ULA
   - Step 2.4. Master MALA algorithm with Metropolis correction
   - Step 2.5. Apply to Bayesian logistic regression
   - Step 2.6. Compare MALA vs Random Walk MH vs HMC
   - Step 2.7. Verify convergence with diagnostics

Step 1.1: Pure Diffusion — No Preferred Direction

Stochastic Differential Equation (SDE):

$$dX_t = \sigma\, dW_t \quad (\mu = 0)$$

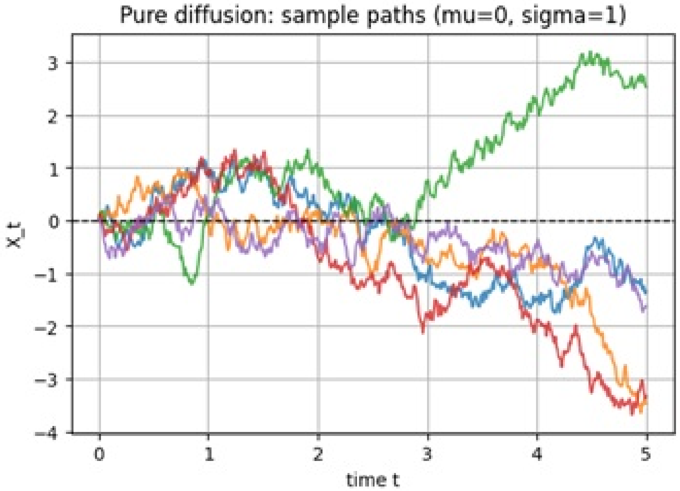
Key Properties:

- No drift term: Position wanders randomly with zero mean
- Symmetric spread: Distribution stays centered at $X_0 = 0$
- Variance grows: $\mathrm{Var}(X_t) = \sigma^2 t$
- Diffusion: spreading without a preferred direction
- Example: Standard Brownian motion (scaled by σ)

Intuition: Like a particle undergoing random thermal motion with no bias.

Numerical Solution: $X_{k+1} = X_k + \sigma\sqrt{\Delta t} \cdot Z_k, \quad Z_k \sim \mathcal{N}(0,1)$

Pure Diffusion: Sample Trajectories



Observation: Multiple paths spread out symmetrically around zero.

Step 1.2: Pure Drift — Deterministic Motion

Stochastic Differential Equation (SDE):

$$dX_t = \mu \, dt \quad (\sigma = 0)$$

Analytical Solution:

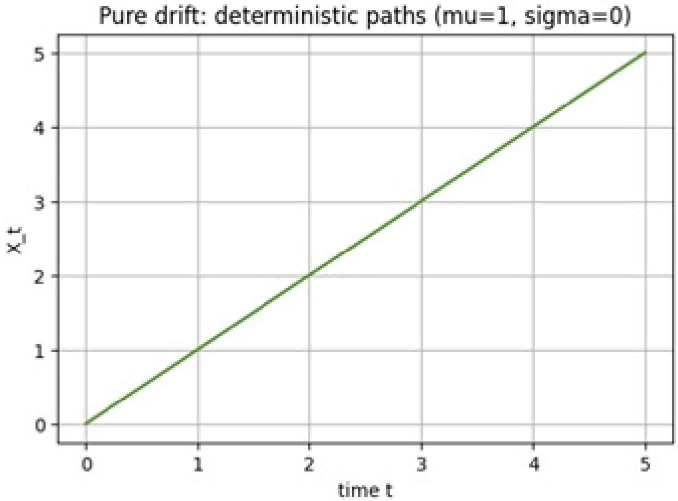
$$X_t = X_0 + \mu t \quad \text{(straight line)}$$

Key Properties:

- Deterministic: No randomness; same path every time
- Linear motion: Moves at constant velocity μ
- No uncertainty: All paths identical
- Example: $\mu = 1.0$ gives $X_t = t$

Intuition: Like an object rolling downhill with fixed velocity.

Pure Drift: Deterministic Paths



Observation: All paths are identical (deterministic)—no randomness!

Step 1.3: Drift + Diffusion — Noisy Upward Trend

Stochastic Differential Equation (SDE):

$$dX_t = \mu\,dt + \sigma\,dW_t \quad (\mu \neq 0, \sigma \neq 0)$$

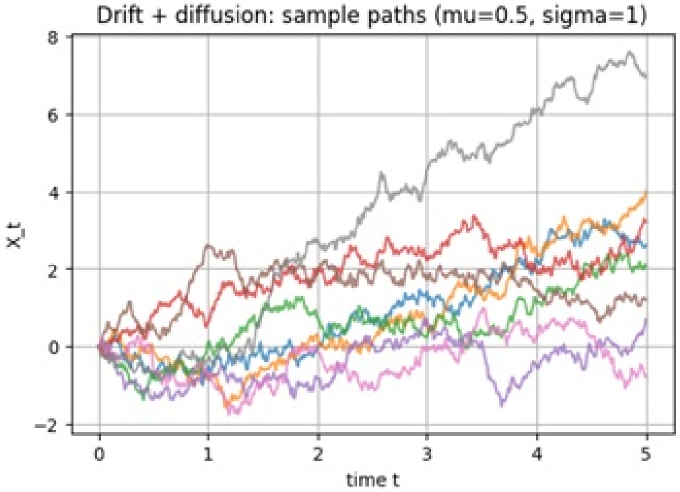
Key Properties:

- Drift + noise: Trajectory trends upward on average, with random fluctuations
- Mean motion: $\mathbb{E}[X_t] = X_0 + \mu t$ (like pure drift)
- Growing variance: $\mathrm{Var}(X_t) = \sigma^2 t$ (like pure diffusion)
- Example: $\mu = 0.5$, $\sigma = 1.0$ gives upward trend with noise

Intuition: Like a particle biased downhill but with thermal random kicks.

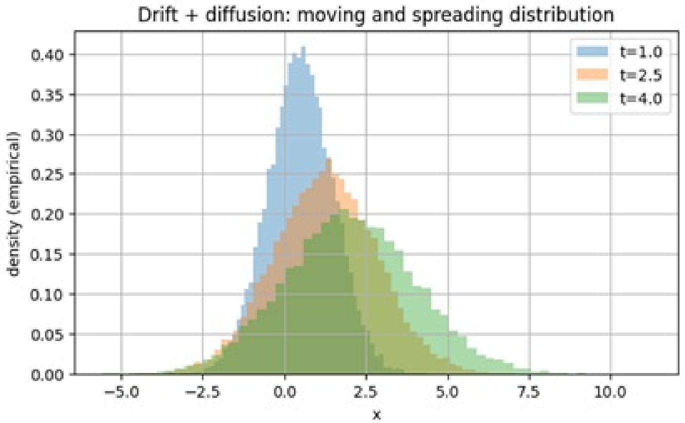
Numerical Solution: $X_{k+1} = X_k + \mu\Delta t + \sigma\sqrt{\Delta t} \cdot Z_k$

Drift + Diffusion: Sample Trajectories



Observation: Paths trend upward on average but with visible random fluctuations.

Drift + Diffusion: Moving and Spreading Distribution



Key Insight: Distribution shifts (mean $= \mu t$) *and* spreads (variance $\propto t$).

Step 1.4: Langevin Dynamics — Sampling from a Known Distribution

Key Idea: Use drift from gradient of log-target (score) to sample efficiently: score$(x) = \nabla_x \log p(x)$

Langevin dynamics: a stochastic process that uses this score to move toward high-probability regions, while noise keeps exploration (so we sample rather than optimize). (drift + Brownian noise)

Langevin SDE:

$$dX_t = \frac{1}{2}\nabla_x \log p(x)\, dt + dW_t$$

Simple Example — Standard Gaussian:

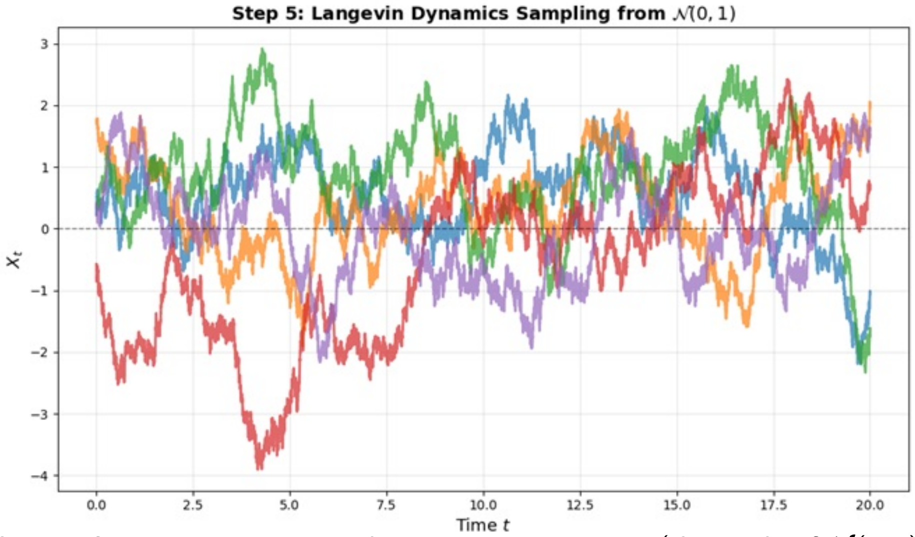
- Target: $p(x) = \mathcal{N}(0,1)$
- Log-density: $\log p(x) = -\frac{1}{2}x^2 + \text{const}$
- Gradient: $\nabla_x \log p(x) = -x$
- Drift: $b(x) = -\frac{1}{2}x$ (pulls toward zero)

Result: Trajectories concentrate around the mode (here, $x = 0$).

Why It Works:

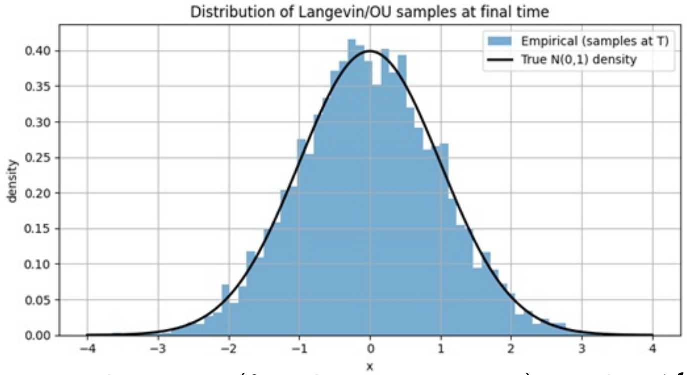
- Drift pulls toward high-probability regions
- Noise prevents convergence to single point
- Stationary distribution = target $p(x)$

Langevin: Trajectories Concentrate Near Mode



Observation: Trajectories spend more time near $x = 0$ (the mode of $\mathcal{N}(0,1)$).

Langevin: Samples Match Target Distribution



Key Result: Empirical histogram (from Langevin samples) matches $\mathcal{N}(0,1)$ perfectly!

Step 1.5: Monte Carlo Estimation — Law of Large Numbers

Problem: Estimate $\mathbb{E}[X_T]$ from many drift-diffusion trajectories.

Setup:

- Generate N independent trajectories of $dX_t = \mu dt + \sigma dW_t$
- Observe value X_T at time T for each trajectory
- Estimate: $\hat{\mathbb{E}}[X_T] = \frac{1}{N}\sum_{n=1}^{N} X_T^{(n)}$
- True value: $\mathbb{E}[X_T] = X_0 + \mu T$ (by linearity of expectation)

Law of Large Numbers:

$$\hat{\mathbb{E}}[X_T] \xrightarrow{N\to\infty} \mathbb{E}[X_T] \quad \text{almost surely}$$

Error decreases as: MSE $\sim \frac{1}{N}$ (independent of dimension!)

Monte Carlo Advantage: Dimension-independent convergence rate, unlike quadrature methods.

From Langevin to MALA: Gradient-Based Bayesian Inference for Complex Posteriors

1. Understand potential energy: $U(\theta) = -\log \pi(\theta)$
2. Learn Langevin dynamics and why it samples from $\pi(\theta)$
3. Understand discretization bias in ULA
4. Master MALA algorithm with Metropolis correction
5. Apply to Bayesian logistic regression
6. Compare MALA vs Random Walk MH vs HMC
7. Verify convergence with diagnostics

Outcome: Gradient-based sampling for any Bayesian model

Step 2.1: What is $U(\theta)$?

Potential Energy = Energy Landscape of Posterior

Bayes' theorem:

$$\pi(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \propto p(y|\theta)p(\theta)$$

Define potential energy:

$$U(\theta) = -\log p(y|\theta) - \log p(\theta)$$

Then:

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

where $Z = \int \exp(-U(\theta))d\theta$ is the partition function.

Intuition:

- Low U = high posterior density
- Samples concentrate where U is small
- Gradient ∇U points uphill

Step 2.1 (continued): Logistic Regression

Computing $U(w)$ and $\nabla U(w)$

Model:

$$p(y_i = 1 | x_i, w) = \sigma(x_i^T w) \tag{1}$$

$$p(w) = \mathcal{N}(0, \lambda^{-1} I) \tag{2}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid.

Potential energy:

$$U(w) = -\sum_{i=1}^{n} [y_i \log \sigma_i + (1 - y_i) \log(1 - \sigma_i)] + \frac{\lambda}{2} \|w\|_2^2$$

Gradient:

$$\nabla U(w) = X^T (\sigma - y) + \lambda w$$

where $\sigma_i = \sigma(x_i^T w)$.

Cost: $O(nd)$ for gradient computation.

Step 2.2: Langevin Dynamics

Continuous-Time Sampling via SDE

Langevin SDE (continuous time):

$$dX_t = -\frac{1}{2}\nabla U(X_t)dt + dW_t$$

Key insight: As $t \to \infty$, the distribution of X_t converges to $\pi(\theta) \propto \exp(-U(\theta))$.

Two components:

- $-\frac{1}{2}\nabla U(X_t)dt$: Gradient descent (push toward mode)
- dW_t: Brownian noise (exploration)
- Balance = samples from correct posterior!

Why this works: the Langevin diffusion is constructed so that its stationary distribution is

$$\pi(\theta) \propto e^{-U(\theta)}.$$

(We use this fact rather than proving it via PDEs in this lecture.)

Step 2.3: Discretization - ULA

Unadjusted Langevin Algorithm (Biased)

Discrete Euler-Maruyama approximation:

$$\theta_{k+1} = \theta_k - \frac{1}{2}\nabla U(\theta_k)\Delta t + \sqrt{\Delta t}\, Z_k$$

where $Z_k \sim \mathcal{N}(0, I)$ are i.i.d. standard normals.

Problem: The stationary distribution is **NOT** exactly $\pi(\theta)$.

Discretization bias: The true stationary distribution is

$$\pi_{\Delta t}(\theta) \approx \pi(\theta) + O(\Delta t) \text{ correction terms}$$

Trade-off:

- Small Δt: Less bias, but slow mixing
- Large Δt: Fast mixing, but large bias!

Solution: Add Metropolis acceptance step $\Rightarrow$ MALA (exact!)

Step 2.4: MALA (Metropolis-Adjusted Langevin)

Exact MCMC via Gradient + Metropolis Correction

Algorithm:

1. **Propose:** $\tilde{\theta} = \theta_k - \frac{1}{2}\nabla U(\theta_k)\Delta t + \sqrt{\Delta t}\, Z_k$
2. **Acceptance probability:**
$$\alpha = \min\left(1, \exp\left(-U(\tilde{\theta}) + U(\theta_k)\right)\right)$$
3. **Update:**
$$\theta_{k+1} = \begin{cases} \tilde{\theta} & \text{with probability } \alpha \\ \theta_k & \text{with probability } 1-\alpha \end{cases}$$

Key properties:

- **Exact:** Stationary distribution is exactly $\pi(\theta)$ (no bias!)
- **Efficient:** Typically 50–70% acceptance rate
- **Fast:** 5–10x faster than random walk MH
- **Cost:** Same $O(nd)$ per iteration

Step 2.4 (continued): Why This Acceptance Ratio?

General Metropolis-Hastings:

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

For Langevin proposal: The proposal is Gaussian

$$q(\tilde{\theta}|\theta_k) = \frac{1}{(2\pi\Delta t)^{d/2}} \exp\left(-\frac{\|\tilde{\theta} - \mu_k\|^2}{2\Delta t}\right)$$

where $\mu_k = \theta_k - \frac{1}{2}\nabla U(\theta_k)\Delta t$.

Click here for visualization.

Step 2.5 Bayesian Logistic Regression with MALA

Data: $X \in \mathbb{R}^{n\times d}$, $y \in \{0,1\}^n$ (0=functional, 1=failed)

Model:

$$P(y_i = 1 \mid x_i, w) = \sigma(x_i^\top w), \qquad \sigma(z) = \frac{1}{1+e^{-z}}$$

$$w \sim \mathcal{N}(0, \lambda^{-1} I)$$

Target posterior:

$$\pi(w) \propto p(y \mid X, w)\, p(w) \quad \Longleftrightarrow \quad U(w) = -\log \pi(w)$$

Why MALA (vs random-walk MH):

- Random-walk MH proposes blindly: $w^* = w + \epsilon$
- MALA proposes using local geometry (a drift term):

$$w^* = w - \tfrac{\Delta t}{2}\nabla U(w) + \sqrt{\Delta t}\, Z, \qquad Z \sim \mathcal{N}(0, I)$$

Step 2.6 Algorithm Comparison

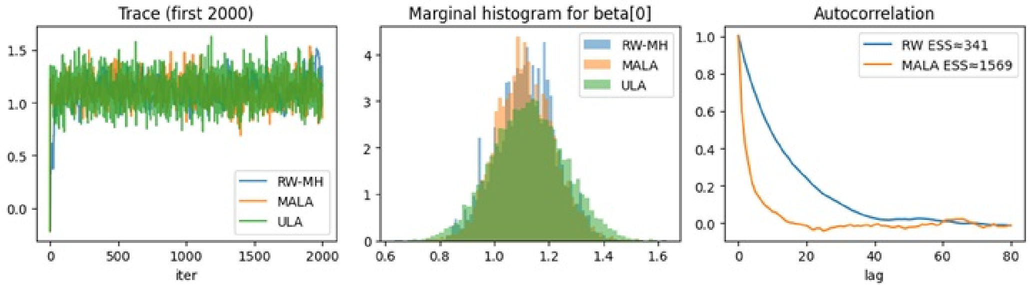


Figure: Comparing MALA and MH for a binary regression problem

Computing Gradients

General template:

$$\nabla U(\theta) = -\nabla \log p(y|\theta) - \nabla \log p(\theta)$$

Common models:

1. **Linear regression:** $\nabla U = X^T(X\beta - y) + \lambda\beta$
2. **Logistic regression:** $\nabla U = X^T(\sigma(X\beta) - y) + \lambda\beta$
3. **Poisson GLM:** $\nabla U = X^T(\exp(X\beta) - y) + \lambda\beta$
4. **Gaussian process:** $\nabla U = (y - \mu)^T K^{-1} \nabla\mu$

Automatic differentiation: Use PyTorch, TensorFlow, or JAX!

Testing: Always verify with finite differences:

$$\frac{\partial U}{\partial \theta_j} \approx \frac{U(\theta + \epsilon e_j) - U(\theta - \epsilon e_j)}{2\epsilon}$$

Step 2.7 Checking Your Chains: Diagnostics

1. **Trace plots:** Plot θ_k vs. iteration k
   - Good: Random scatter, no trends
   - Bad: Visible trends, stuck values
2. **Acceptance rate:**
   - Good: 50–70% (ideal for MALA)
   - Too low: Increase Δt
   - Too high: Decrease Δt
3. **Autocorrelation function (ACF):**

$$\text{ACF}(k) = \frac{\text{Cov}(\theta_t, \theta_{t+k})}{\text{Var}(\theta_t)}$$

Effective sample size: $\text{ESS} = \frac{N}{1+2\sum_{k=1}^{\infty} \rho_k}$

4. **Gelman-Rubin statistic ($\hat{R}$):** $\hat{R} = \sqrt{\frac{\text{Var}_{\text{between}}}{\text{Var}_{\text{within}}}}$
   - $\hat{R} < 1.01$: Converged
   - $\hat{R} > 1.05$: Not converged

Practical Tips for MALA

1. **Data preprocessing:**
   - Standardize: $X = (X - \bar{X})/\sigma_X$
   - Avoids numerical issues
2. **Tune step size Δt:**
   - Start with $\Delta t = 0.01$
   - Adjust for approximately 60% acceptance
   - Test gradient with finite differences first!
3. **Burn-in and chains:**
   - Burn-in: Discard first approximately 25% of iterations
   - Run 3–4 independent chains
   - Check $\hat{R} < 1.01$ for convergence
4. **If things go wrong:**
   - Low acceptance? Decrease Δt
   - Gradient NaN? Standardize data
   - Divergent? Use stronger priors
   - Slow mixing? Reparameterize model

Summary: From Langevin to MALA

1. **Potential energy** $U(\theta) = -\log \pi(\theta)$ defines posterior
2. **Langevin SDE** has stationary distribution $\pi(\theta)$
3. **Discretization** introduces bias, Metropolis fixes it
4. **MALA:** Propose via Langevin, accept/reject via Metropolis
5. **Efficiency:** 50–70% acceptance, 10x faster than RW-MH
6. **Gradient computation:** Essential! Use autodiff
7. **Diagnostics:** Trace plots, ACF, $\hat{R}$, ESS
8. **Applicable:** Any model where you can compute $\nabla U(\theta)$

Gradient-based sampling is the future of Bayesian inference!

Further Reading

- Pavliotis, G.A. (2014). *Stochastic Processes and Applications*. Springer. - Rigorous treatment of SDEs and Fokker-Planck equations
- Girolami, M. & Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *JMLR*. - MALA on manifolds; metric-adaptive proposals
- Roberts, G.O. & Rosenthal, J.S. (2004). General state space Markov chains and MCMC algorithms. *Probability Surveys*. - Theory of convergence and efficiency
- Hoffman, M.D. & Gelman, A. (2014). The No-U-Turn Sampler. *JMLR*. - Modern HMC with adaptive trajectory length
- Betancourt, M. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo. `arXiv:1701.02434`. - Intuitive explanation of gradient-based sampling
- **Probabilistic Programming:** Stan, PyMC, Turing.jl all use NUTS (refinement of HMC/MALA ideas).

Implementation Resources

For real Bayesian inference, use probabilistic programming:

Tool	**Highs**
Stan	Automatic NUTS sampler, wide model support
PyMC	Python-native, integrates with numpy/pandas
Turing.jl	Julia (fast), embedded in language
Numpyro	weight, JAX backend, modern autodiff
Blackjax	Composable MCMC kernels, JAX

Acknowledgements/ Document preparation

This document was developed and converted into LaTeX slides and formatted with assistance from ChatGPT (OpenAI) and CoPilot (Microsoft). The instructor modified the source text and verified the final structure and wording.