# **Modern Hamiltonian Sampling**: HMC, NUTS, and Riemannian Manifold Extensions

Based on Hoffman & Gelman (2014)

Franciszek Hansdorfer

Statistics journal club

March 9, 2026

# The Bayesian Inference Problem

## Goal

Draw samples from a target posterior distribution:

$$p(\boldsymbol{\theta} \mid \text{data}) \propto \exp\{\mathcal{L}(\boldsymbol{\theta})\}$$

where $\mathcal{L}(\boldsymbol{\theta}) = \log p(\boldsymbol{\theta}, \text{data})$.

## Challenge

For complex hierarchical models in $D$ dimensions, exact inference is **rarely tractable**.

**Why MCMC?**

- Asymptotically *unbiased*
- More general than deterministic methods
- Scales to complex models

# Classical MCMC: Why It Struggles

## Random Walk Metropolis (RWM)

- Proposes small random steps
- Cost: $\mathcal{O}(D^2)$ per independent sample
- Sensitive to correlated parameters

## Gibbs Sampling

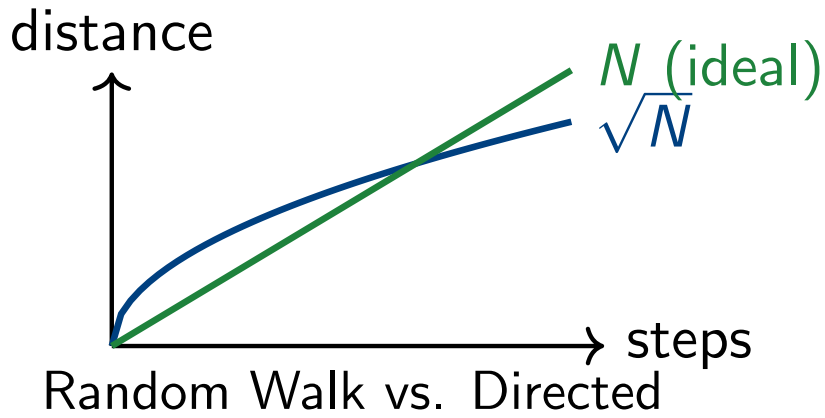
- Updates one coordinate at a time
- Slow to escape correlated regions
- Requires conditional distributions

## The Core Problem

Both methods exhibit **random walk behavior**:

$$\text{distance} \propto \sqrt{N_{\text{steps}}}$$

With large $D$ this becomes catastrophically slow.



# Enter Hamiltonian Monte Carlo

**The idea:** Transform sampling into a *physics simulation problem*.

- Introduce auxiliary **momentum** variables **r**
- Simulate **Hamiltonian dynamics** to make large, directed proposals
- Suppress random walk behavior

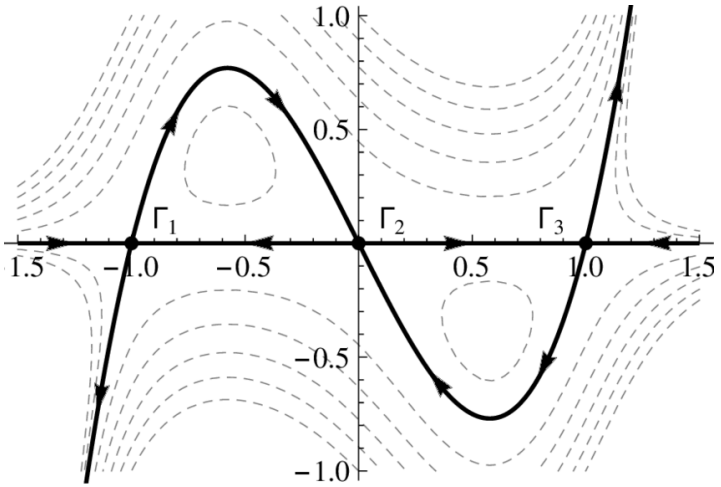



Figure: HMC follows the iso-probability contours (from Baek, Kafri 2015)

**Efficiency Gain**

| Method | Cost per sample |
|---|---|
| Random Walk | $\mathcal{O}(D^2)$ |
| **HMC** | $\mathcal{O}(\mathbf{D^{5/4}})$ |

# The Hamiltonian Framework

**Augmented State Space:**
Introduce momentum $r_d$ for each $\theta_d$. The joint density is:

$$p(\boldsymbol{\theta}, \mathbf{r}) \propto \exp\{\mathcal{L}(\boldsymbol{\theta}) - \tfrac{1}{2}\mathbf{r} \cdot \mathbf{r}\}$$

## Physical Interpretation

- $\boldsymbol{\theta}$: particle position in $D$-dim space
- $\mathbf{r}$: particle momentum
- $\mathcal{L}(\boldsymbol{\theta})$: negative potential energy
- $\frac{1}{2}\mathbf{r} \cdot \mathbf{r}$: kinetic energy
- $-\log p(\boldsymbol{\theta}, \mathbf{r})$: total energy (Hamiltonian)

## Hamilton's Equations

$$\frac{d\boldsymbol{\theta}}{dt} = \mathbf{r}, \quad \frac{d\mathbf{r}}{dt} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

**Key properties of Hamiltonian dynamics:**

- **Energy conservation**: $H = \text{const}$ (exact dynamics)
- **Reversibility**: time-reversal symmetry
- **Volume preservation**: Liouville's theorem

# The Leapfrog Integrator

Since exact simulation is impossible, we use the **Störmer-Verlet (leapfrog)** integrator:

## Leapfrog Update

$$\mathbf{r}^{t+\epsilon/2} = \mathbf{r}^{t} + \frac{\epsilon}{2}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{t})$$

$$\boldsymbol{\theta}^{t+\epsilon} = \boldsymbol{\theta}^{t} + \epsilon\,\mathbf{r}^{t+\epsilon/2}$$

$$\mathbf{r}^{t+\epsilon} = \mathbf{r}^{t+\epsilon/2} + \frac{\epsilon}{2}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{t+\epsilon})$$

**Properties:**

- *Volume-preserving* (Jacobian $= 1$)
- *Time-reversible*
- Energy error $\propto \epsilon^2$ per step

# HMC Algorithm

**Algorithm 1** Hamiltonian Monte Carlo

**Require:** $\theta^0$, step size $\epsilon$, number of steps $L$, samples $M$

1: **for** $m = 1$ **to** $M$ **do**
2: Sample $\mathbf{r}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Gibbs step on momentum
3: Set $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^{m-1}$, $\tilde{\mathbf{r}} \leftarrow \mathbf{r}^0$
4: **for** $i = 1$ **to** $L$ **do**
5: $\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{r}} \leftarrow \texttt{Leapfrog}(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{r}}, \epsilon)$ ▷ Simulate dynamics
6: **end for**
7: With prob. $\alpha = \min\left(1, \frac{\exp\{\mathcal{L}(\tilde{\boldsymbol{\theta}}) - \frac{1}{2}\tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}}\}}{\exp\{\mathcal{L}(\boldsymbol{\theta}^{m-1}) - \frac{1}{2}\mathbf{r}^0 \cdot \mathbf{r}^0\}}\right)$ set $\boldsymbol{\theta}^m \leftarrow \tilde{\boldsymbol{\theta}}$ ▷ MH accept/reject
8: **end for**

## Why does this work?

- Momentum resampling: Gibbs update
- Leapfrog is volume-preserving & reversible
- MH step: corrects for discretization error
- If exact: $\alpha = 1$ always (energy conserved)

## The Critical Problem

Must specify **two parameters**:

- $\epsilon$ (step size): wrong $\Rightarrow$ low acceptance or wasted steps
- $L$ (number of steps): **very hard to tune!**

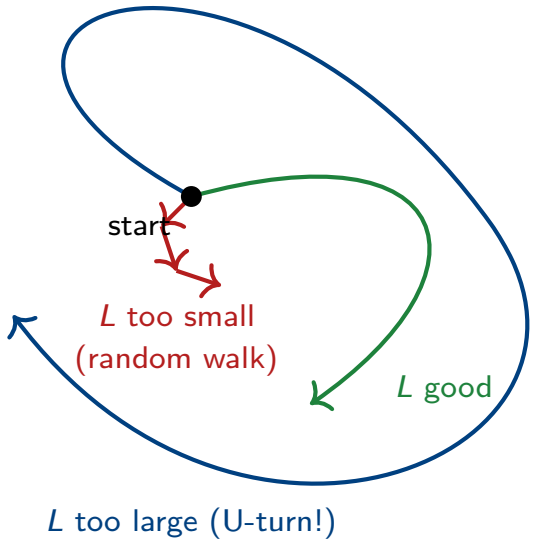
# The Step Size vs. Number of Steps Dilemma

**Effect of wrong $\epsilon$:**

- Too large: large energy errors, low $\alpha$
- Too small: many tiny steps, waste computation
- Can be tuned by targeting a specific acceptance rate

**Effect of wrong $L$:**

- Too small: random-walk behavior, slow mixing
- Too large: trajectory loops back, wastes work
- May create a *non-ergodic* chain!
- **No simple metric to tune $L$**



# Why $L$ is Hard to Tune

**The U-turn problem:**

- A trajectory of length $L\epsilon$ traces a curve in $\theta$-space
- Eventually the trajectory *doubles back* toward the start
- Optimal $L$ depends on **local geometry** of $p(\theta)$

## Practical Difficulties

- Different $L$ optimal for different models
- Across 4 test models: optimal $\lambda = \epsilon L$ varied by **factor of** $\sim 100$
- Tuning requires vast experience + costly preliminary runs

## The **U-turn criterion**

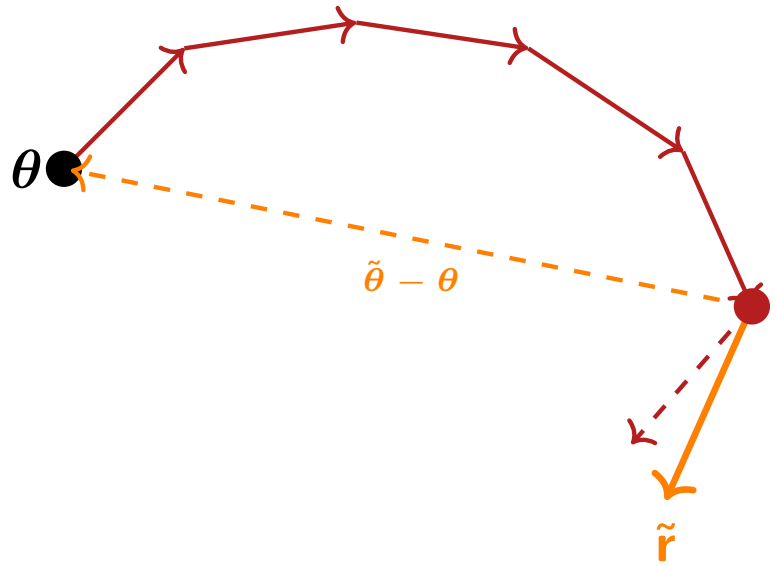
We should stop the simulation when:

$$(\tilde{\theta} - \theta) \cdot \tilde{\mathbf{r}} < 0$$

i.e., the particle starts moving *back* toward the start!

# NUTS: Core Idea

**Goal:** Automatically determine when to stop the leapfrog simulation.




## The U-turn Criterion

Stop when the trajectory starts doubling back. Formally, track:

$$\frac{d}{dt}\frac{(\tilde{\theta}-\theta)\cdot(\tilde{\theta}-\theta)}{2} = (\tilde{\theta}-\theta)\cdot\tilde{\mathbf{r}}$$

This is the rate of increase of the squared distance from start.
**Stop when** $(\tilde{\theta}-\theta)\cdot\tilde{\mathbf{r}} < 0$

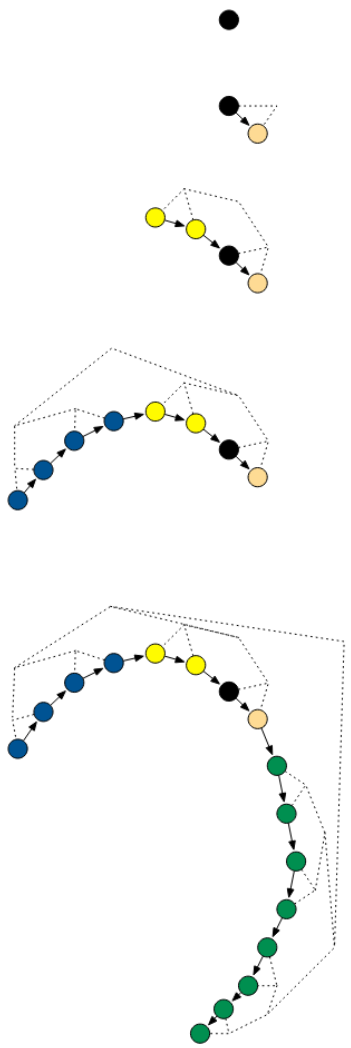
**Problem:** Naive stopping *violates time-reversibility!*

# NUTS: Binary Tree Doubling

**Solution:** Build a balanced binary tree of leapfrog states by *doubling* in random directions.

**Doubling Procedure:**

1. Choose direction $v_j \sim \text{Uniform}(\{-1, +1\})$
2. Take $2^j$ leapfrog steps in direction $v_j$
3. **Stop** when any subtree's endpoints satisfy U-turn criterion



**Why treese?**

- Preserves **time-reversibility**
- Any node could generate the same tree with probability $2^{-j}$
- Satisfies detailed balance

# NUTS: The Slice Variable & Candidate Set

**Slice variable $u$:** Introduced for clean derivation.

$$p(u|\boldsymbol{\theta}, \mathbf{r}) = \text{Uniform}\big(0,\, \exp\{\mathcal{L}(\boldsymbol{\theta}) - \tfrac{1}{2}\mathbf{r}\cdot\mathbf{r}\}\big)$$

This renders $p(\boldsymbol{\theta}, \mathbf{r}|u)$ uniform over states satisfying:

$$\exp\{\mathcal{L}(\boldsymbol{\theta}) - \tfrac{1}{2}\mathbf{r}\cdot\mathbf{r}\} \geq u$$

## Candidate Set $\mathcal{C}$

States from the trajectory that lie *within the slice*:

$$\mathcal{C} = \big\{(\boldsymbol{\theta}', \mathbf{r}') \in \mathcal{B} : \exp\{\mathcal{L}(\boldsymbol{\theta}') - \tfrac{1}{2}\mathbf{r}'\cdot\mathbf{r}'\} \geq u\big\}$$

Final sample drawn *uniformly* from $\mathcal{C}$.

# NUTS: Stopping Criteria

NUTS halts doubling when *any* of these conditions hold:

## Criterion 1: U-turn

For the leftmost $(\boldsymbol{\theta}^-, \mathbf{r}^-)$ and rightmost $(\boldsymbol{\theta}^+, \mathbf{r}^+)$ nodes of any balanced subtree:

$$(\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-) \cdot \mathbf{r}^- < 0$$

$$\text{or} \quad (\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-) \cdot \mathbf{r}^+ < 0$$

Checked for **all** $2^j - 1$ balanced subtrees of height $> 0$.

## Criterion 2: Divergence

Stop if any leaf node has extremely low probability:

$$\mathcal{L}(\boldsymbol{\theta}') - \tfrac{1}{2}\mathbf{r}' \cdot \mathbf{r}' - \log u < -\Delta_{\text{max}}$$

Recommended: $\Delta_{\text{max}} = 1000$.

Catches numerical instability or hard constraints.

## Note on Subtree Checking

Checking all subtrees requires $2^{j+1} - 2$ inner products per doubling — comparable cost to $2^j - 1$ leapfrog steps, so overhead is negligible.

# NUTS: Naïve Algorithm

**Algorithm 2** Naïve NUTS

Sample $\mathbf{r}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $u \sim \text{Unif}[0, e^{\mathcal{L}(\boldsymbol{\theta}^{m-1}) - \frac{1}{2}\mathbf{r}^0 \cdot \mathbf{r}^0}]$
Init $\boldsymbol{\theta}^- = \boldsymbol{\theta}^+ = \boldsymbol{\theta}^{m-1}$, $\mathbf{r}^{\pm} = \mathbf{r}^0$, $j = 0$, $\mathcal{C} = \{(\boldsymbol{\theta}^{m-1}, \mathbf{r}^0)\}$, $s = 1$
**while** $s = 1$ **do**
    $v_j \sim \text{Unif}(\{-1, +1\})$
    **if** $v_j = -1$ **then** build tree backward; add $\mathcal{C}'$ to $\mathcal{C}$
    **else** build tree forward; add $\mathcal{C}'$ to $\mathcal{C}$
    **end if**
    $s \leftarrow s' \cdot \mathbb{1}[(\boldsymbol{\theta}^+ - \boldsymbol{\theta}^-) \cdot \mathbf{r}^- \geq 0] \cdot \mathbb{1}[\cdot \cdots \cdot \mathbf{r}^+ \geq 0]$
    $j \leftarrow j + 1$
**end while**
Sample $\boldsymbol{\theta}^m$, **r uniformly** from $\mathcal{C}$

**Memory Issue**

Naïve NUTS stores **all** $2^j$ position-momentum pairs in $\mathcal{C}$.

For large trajectories this is **unacceptable**.

# NUTS: Efficient Algorithm

**Two key improvements:**

## 1. Memory-Efficient Sampling from $\mathcal{C}'$

Exploit the binary tree structure. For each subtree:

- Keep one *representative sample* per subtree
- When combining two subtrees of sizes $n'$ and $n''$:
  Accept new sample with prob. $\frac{n''}{n'+n''}$
- Only need $\mathcal{O}(j)$ vectors, not $\mathcal{O}(2^j)$!

## 2. Better Transition Kernel

Instead of uniform sampling from all of $\mathcal{C}$, use a *Metropolis-like* kernel:

- Propose move to new half-tree $\mathcal{C}_{\text{new}}$
- Accept with prob. $\min\left(1, \frac{|\mathcal{C}_{\text{new}}|}{|\mathcal{C}_{\text{old}}|}\right)$

# NUTS: BuildTree Recursion

**Algorithm 3** BuildTree (efficient)

**function** $\textsc{BuildTree}(\boldsymbol{\theta}, \mathbf{r}, u, v, j, \epsilon)$
  **if** $j = 0$ **then** ▷ Base case
    $\boldsymbol{\theta}', \mathbf{r}' \leftarrow \texttt{Leapfrog}(\boldsymbol{\theta}, \mathbf{r}, v\epsilon)$
    $n' \leftarrow \mathbb{1}[u \leq e^{\mathcal{L}(\boldsymbol{\theta}') - \frac{1}{2}\mathbf{r}' \cdot \mathbf{r}'}]$
    $s' \leftarrow \mathbb{1}[\mathcal{L}(\boldsymbol{\theta}') - \frac{1}{2}\mathbf{r}' \cdot \mathbf{r}' > \log u - \Delta_{\text{max}}]$
    **return** $\boldsymbol{\theta}', \mathbf{r}', \boldsymbol{\theta}', \mathbf{r}', \boldsymbol{\theta}', n', s'$
  **else** ▷ Recurse
    Build left subtree $\rightarrow$ get $\boldsymbol{\theta}^-, \mathbf{r}^-, \boldsymbol{\theta}^+, \mathbf{r}^+, \theta', n', s'$
    **if** $s' = 1$ **then** Build right subtree; with prob. $\frac{n''}{n'+n''}$ set $\theta' \leftarrow \theta''$
    **end if**
    Check U-turn on combined subtree
    **return** $\boldsymbol{\theta}^-, \mathbf{r}^-, \boldsymbol{\theta}^+, \mathbf{r}^+, \theta', n' + n'', s'$
  **end if**
**end function**

## Memory complexity

Naïve NUTS: $\mathcal{O}(2^j)$ vectors $\rightarrow$ Efficient NUTS: $\mathcal{O}(j)$ vectors

# The Step Size Tuning Problem

Even with NUTS eliminating $L$, we still need to tune $\epsilon$.

**Classic approach: Robbins-Monro (1951)**

Define statistic $H_t$ (e.g., $H_t = \delta - \alpha_t$ where $\alpha_t$ is acceptance prob.). Update:

$$x_{t+1} \leftarrow x_t - \eta_t H_t, \quad x = \log \epsilon$$

If $\sum_t \eta_t = \infty \wedge \sum_t \eta_t^2 < \infty$ then $H_t \to 0$.

**Problem with Robbins-Monro**

Early (bad) iterations get *disproportionately high* weight.

**Target: Acceptance Rate**

For HMC:

$$H_t^{\text{HMC}} = \delta - \alpha_t$$

For NUTS (no single accept step!):

$$H_t^{\text{NUTS}} = \delta - \frac{1}{|\mathcal{B}_t^{\text{final}}|} \sum_{\boldsymbol{\theta}, \mathbf{r} \in \mathcal{B}_t^{\text{final}}} \frac{p(\boldsymbol{\theta}, \mathbf{r})}{p(\boldsymbol{\theta}^{m-1}, \mathbf{r}^0)}$$

Average acceptance prob. of states in final doubling.

# Dual Averaging (Nesterov 2009)

## Dual Averaging Update

$$x_{t+1} \leftarrow \mu - \frac{\sqrt{t}}{\gamma} \frac{1}{t + t_0} \sum_{i=1}^{t} H_i$$

$$\bar{x}_{t+1} \leftarrow \eta_t x_{t+1} + (1 - \eta_t)\bar{x}_t$$

where $\eta_t = t^{-\kappa}$, and $x = \log \epsilon$.

## Practice

Adapt $\epsilon$ during *warmup* iterations, then freeze $\epsilon = \bar{\epsilon}_{M_{\text{adapt}}}$ for sampling.

**Advantages over Robbins-Monro:**

- $t_0 > 0$: stabilizes early iterations; prevents extreme $\epsilon$ values
- $\kappa < 1$: gives more weight to *recent* iterates; forgets early transient phase faster
- $\mu > \epsilon_0$: biases toward *larger* $\epsilon$ values (cheaper per step)
- **Convergence guarantee**: $\bar{x}_t \to x^*$ as $t \to \infty$

# Finding a Good Initial $\epsilon_0$

**Algorithm 4** FindReasonableEpsilon

Initialize $\epsilon = 1$, sample $\mathbf{r} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\boldsymbol{\theta}', \mathbf{r}' \leftarrow \texttt{Leapfrog}(\boldsymbol{\theta}, \mathbf{r}, \epsilon)$
$a \leftarrow 2 \cdot \mathbb{1}\left[\frac{p(\boldsymbol{\theta}', \mathbf{r}')}{p(\boldsymbol{\theta}, \mathbf{r})} > 0.5\right] - 1$
**while** $\left(\frac{p(\boldsymbol{\theta}', \mathbf{r}')}{p(\boldsymbol{\theta}, \mathbf{r})}\right)^a > 2^{-a}$ **do**
    $\epsilon \leftarrow 2^a \epsilon$
    $\boldsymbol{\theta}', \mathbf{r}' \leftarrow \texttt{Leapfrog}(\boldsymbol{\theta}, \mathbf{r}, \epsilon)$
**end whilereturn** $\epsilon$

**Idea:** Double or halve $\epsilon$ until the single-step Langevin acceptance probability crosses 0.5.

## Why this matters

- Good $\epsilon_0 \Rightarrow$ faster dual averaging convergence
- Avoids extreme values during warmup
- Simple one-parameter search

## Setting $\mu = \log(10\epsilon_0)$

- Shrinkage toward $10\times$ initial $\epsilon$
- Encourages trying *larger* step sizes
- Larger $\epsilon =$ fewer steps $=$ cheaper

# Experimental Setup

**Algorithms compared:**

- **NUTS** with dual averaging (15 values of $\delta$)
- **HMC** with dual averaging (10 values of $\lambda$, 8 values of $\delta$)
- Total: $\sim$3800 experiments, 10 seeds each

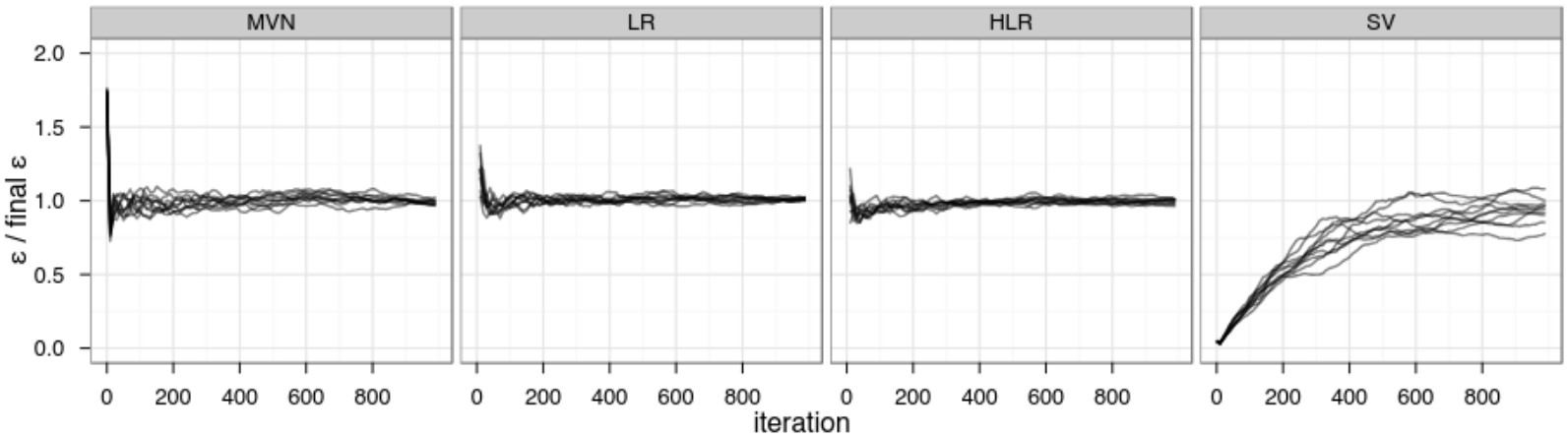
**Setup:**

- 2000 iterations, 1000 warmup
- $\gamma = 0.05$, $t_0 = 10$, $\kappa = 0.75$
- HMC step size jittered $\pm 10\%$ after warmup

**Metric:** ESS / number of gradient evaluations

1. **250-dim MVN**: Zero-mean multivariable normal with strong correlations
2. **Bayesian Logistic Regression (LR)**: German credit data, 25 parameters
3. **Hierarchical LR (HLR)**: Adding two-way interactions; 302 parameters
4. **Stochastic Volatility (SV)**: S&P 500 returns, 3001-dimensional target

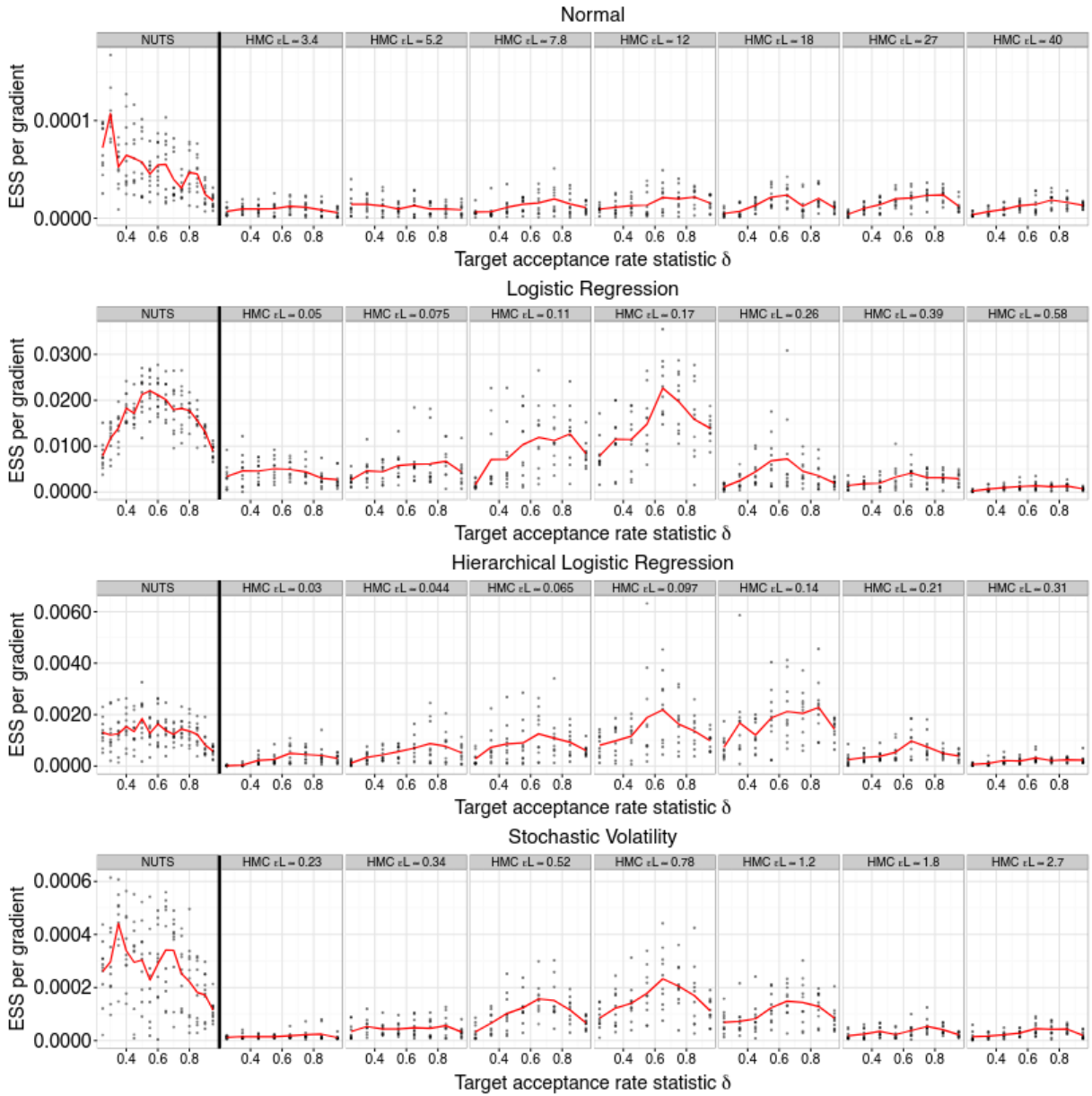
# Dual Averaging Convergence



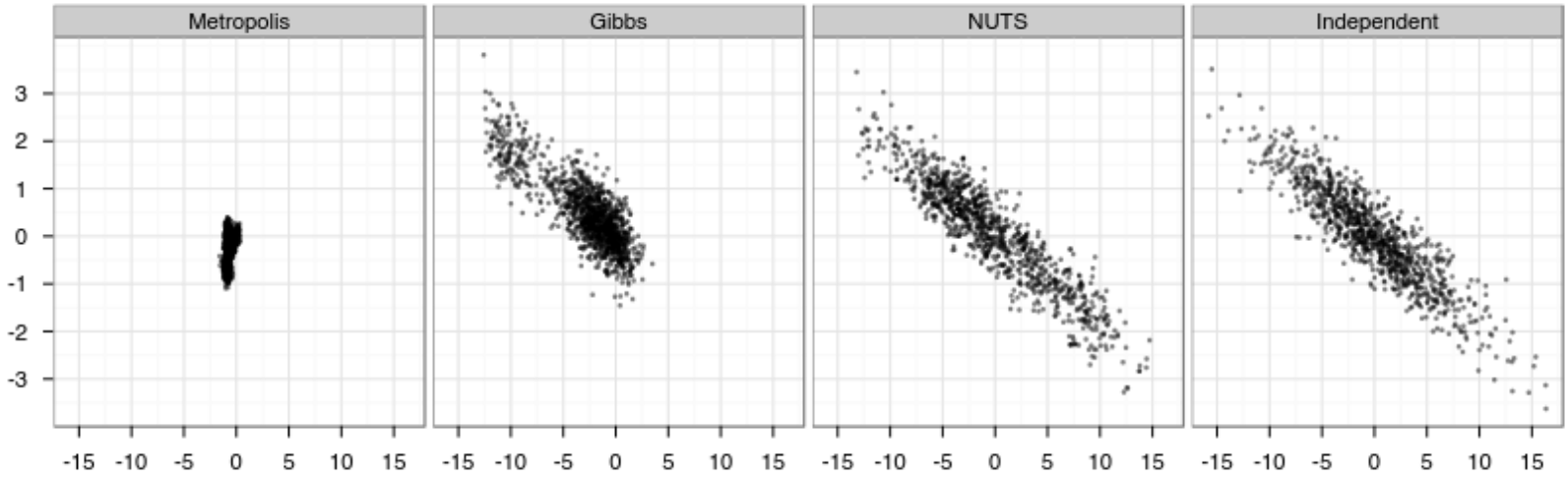

## Observations

- Discrepancy $|h - \delta|$ typically small
- Works well across all 4 models
- Slightly worse for SV model:
  - Longer warmup needed to reach stationarity
  - Less adaptation time available

# NUTS vs. HMC: Efficiency Comparison



# Qualitative Comparison: NUTS vs. RWM vs. Gibbs




**Experiment:** 250-dim correlated MVN, equal total computation:

- **RWM**: 1,000,000 samples (optimal proposal scale)
- **Gibbs**: 1,000,000 samples
- **NUTS**: 2,000 samples (1,000 warmup)

- RWM: barely began to explore the space
- Gibbs: better, but many regions unexplored
- NUTS: effectively independent samples matching true distribution

# Optimal Acceptance Rate

## NUTS empirical finding

- NUTS best performance at $\delta \approx 0.6$
- Robust: good performance for $\delta \in [0.45, 0.65]$
- **Default recommendation:** $\delta = 0.6$

## HMC empirical finding

- Best performance also near $\delta \approx 0.65$
- Consistent with theory
- **Default recommendation:** $\delta = 0.65$

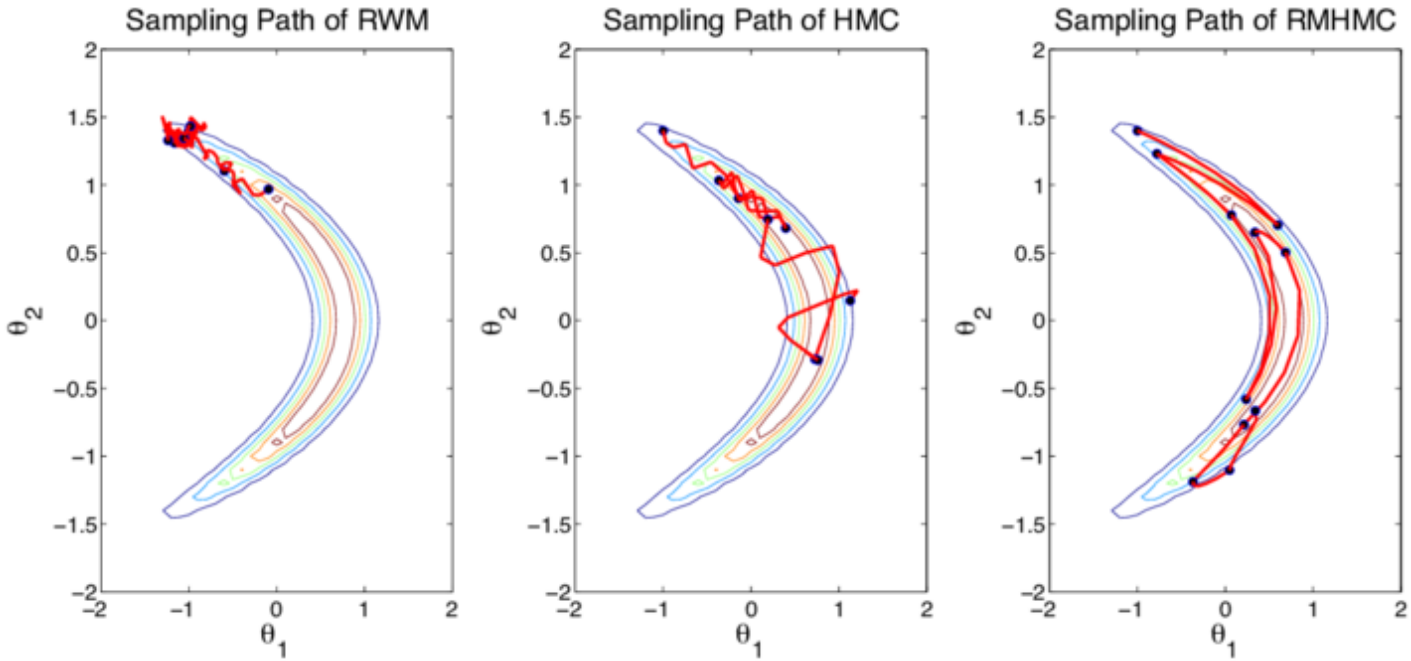
# Motivation: The Geometry Problem

**HMC/NUTS limitation:**

- Standard kinetic energy: $\frac{1}{2}\mathbf{r}^T\mathbf{M}^{-1}\mathbf{r}$
- Fixed mass matrix $\mathbf{M}$
- Fails when curvature of $p(\theta)$ varies **strongly with location**

**Typical problem cases**

- Hierarchical models with funnel geometries
- Posteriors with strong nonlinear correlations
- Heavy-tailed distributions




From *Lagrangian Dynamical Monte Carlo* (Lan et al. 2012)

# Riemannian Manifold HMC (RMHMC)

**Key idea** (Girolami & Calderhead, 2011):
Replace the fixed mass matrix with a *position-dependent* metric tensor $\mathbf{G}(\theta)$. **Modified Hamilton's equations:**

$$\frac{d\theta_d}{dt} = \left[\mathbf{G}(\theta)^{-1}\mathbf{r}\right]_d$$
$$\frac{dr_d}{dt} = -\frac{\partial H}{\partial \theta_d}$$

**Choosing $\mathbf{G}(\theta)$**

Natural choice: the **Fisher information matrix**:

$$\mathbf{G}(\theta) = \mathbb{E}_p\left[-\nabla^2_\theta \mathcal{L}(\theta)\right]$$

**RMHMC Hamiltonian**

$$H(\boldsymbol{\theta}, \mathbf{r}) = -\mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{r}^T\mathbf{G}(\boldsymbol{\theta})^{-1}\mathbf{r} + \frac{1}{2}\log\left|(2\pi)^D\mathbf{G}(\boldsymbol{\theta})\right|$$

# RMHMC: The Generalized Leapfrog

**Problem:** Position-dependent $\mathbf{G}(\theta)$ breaks the standard leapfrog.

The momentum half-step now depends on $\theta$, and the position step depends on $\mathbf{r}$ *and* $\theta$ simultaneously.

**When is $\mathcal{O}(D^3)$ acceptable?**

- $D$ small to moderate ($D \lesssim 100$)
- Posterior geometry highly pathological
- Mixing improvement outweighs per-step cost

**Generalized (Implicit) Leapfrog**

The position update becomes an **implicit** equation:

$$\theta^{t+\epsilon} = \theta^t + \frac{\epsilon}{2}\left[\mathbf{G}(\theta^t)^{-1} + \mathbf{G}(\theta^{t+\epsilon})^{-1}\right]\mathbf{r}^{t+\epsilon/2}$$

Must solve via **fixed-point iteration** at each step.

# RMHMC: Gains and Trade-offs

## Demonstrated gains

- **Log-Gaussian Cox process**: RMHMC $\sim 100\times$ more efficient than HMC per gradient evaluation
- **Stochastic volatility**: dramatically better mixing through funnels
- **Correlated posteriors**: adapts locally, removing need to globally rescale

| Property | HMC | RMHMC |
|---|---|---|
| Cost/step | $\mathcal{O}(D^{5/4})$ | $\mathcal{O}(D^3)$ |
| Mass matrix | Fixed | Position-dep. |
| Funnel geom. | Poor | Good |
| Auto-tuning | NUTS | NUTS ext. |

## Practical verdict

For most modern large-scale models, **NUTS + diagonal mass matrix** (estimated during warmup) offers a better cost-efficiency trade-off than full RMHMC. RMHMC shines on *small, badly-conditioned* posteriors.