NUTS for NUTS: Advances in No-U-Turn Sampling and the Future of Markov Chain Monte Carlo

Nawaf Bou-Rabee (Rutgers, Flatiron, & Wharton)

arXiv: 2506.18746 **GitHub:** `flatironinstitute/walnuts`

Markov Chain Monte Carlo

- **Goal.** Sample $\boldsymbol{\theta} \sim \mu$ on $\mathbb{R}^d$, where μ is known up to a normalizing constant.

Markov Chain Monte Carlo

- **Goal.** Sample $\boldsymbol{\theta} \sim \mu$ on $\mathbb{R}^d$, where μ is known up to a normalizing constant.

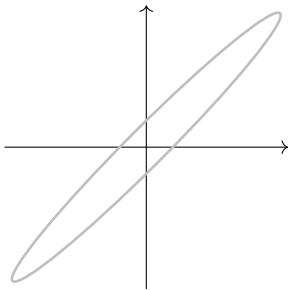
- **Key Idea.** Simulate a Markov chain whose $\boldsymbol{\theta}$-marginal converges to μ.

Markov Chain Monte Carlo

- **Goal.** Sample $\boldsymbol{\theta} \sim \mu$ on $\mathbb{R}^d$, where μ is known up to a normalizing constant.
- **Key Idea.** Simulate a Markov chain whose $\boldsymbol{\theta}$-marginal converges to μ.
- **Challenge.** In modern applications,

$$d \gg 1 \qquad \text{and} \qquad \mu \text{ is often ill-conditioned}$$

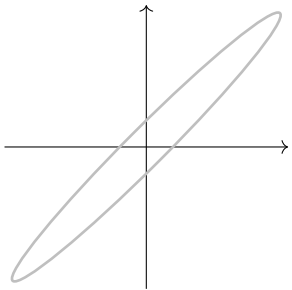
From Gibbs to Hamiltonian Monte Carlo



Gibbs sampler[1] underlies **BUGS** (1991) and **JAGS** (2007).

[1] (Geman & Geman (1984))

From Gibbs to Hamiltonian Monte Carlo

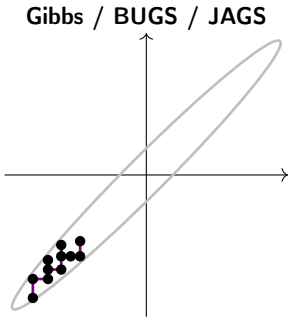


Gibbs sampler[1] underlies **BUGS** (1991) and **JAGS** (2007).

Gibbs is **diffusive**[2]: mixing time $\asymp$ condition number.

[1] (Geman & Geman (1984))
[2] (Ascolani, Lavenant & Zanella, 2024)

From Gibbs to Hamiltonian Monte Carlo



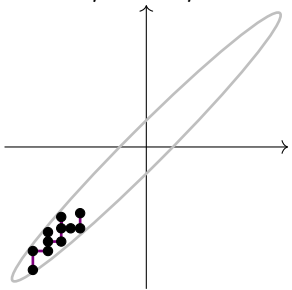
Gibbs sampler[1] underlies **BUGS** (1991) and **JAGS** (2007).

Gibbs is **diffusive**[2]: mixing time $\asymp$ condition number.

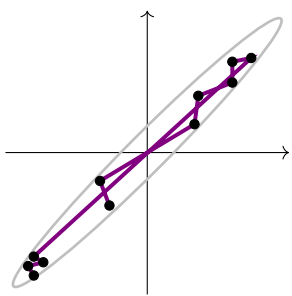
[1] (Geman & Geman (1984))
[2] (Ascolani, Lavenant & Zanella, 2024)

From Gibbs to Hamiltonian Monte Carlo

Gibbs / BUGS / JAGS



well-tuned HMC



Gibbs sampler[1] underlies **BUGS** (1991) and **JAGS** (2007).

Gibbs is **diffusive**[2]: mixing time $\asymp$ condition number.

[1] (Geman & Geman (1984))
[2] (Ascolani, Lavenant & Zanella, 2024)

Hamiltonian Monte Carlo: sampling via measure-preserving flows

- **Lift.** Introduce momentum $\boldsymbol{\rho} \in \mathbb{R}^d$ and joint distribution over $(\boldsymbol{\theta}, \boldsymbol{\rho}) \in \mathbb{R}^{2d}$

$$\boxed{\Pi = \mu \otimes \mathcal{N}(0, \boldsymbol{M})}$$

Hamiltonian Monte Carlo: sampling via measure-preserving flows

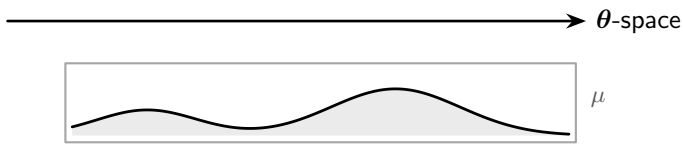
- **Lift.** Introduce momentum $\boldsymbol{\rho} \in \mathbb{R}^d$ and joint distribution over $(\boldsymbol{\theta}, \boldsymbol{\rho}) \in \mathbb{R}^{2d}$

$$\boxed{\Pi = \mu \otimes \mathcal{N}(0, \boldsymbol{M})}$$

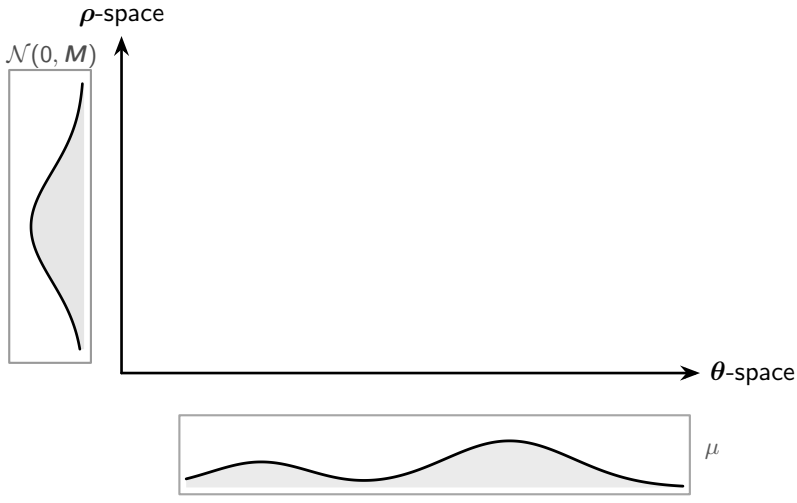
- **Flow.** Follow a deterministic flow on $\mathbb{R}^{2d}$ that preserves Π,

$$\dot{\boldsymbol{\theta}}_t = \boldsymbol{M}^{-1}\boldsymbol{\rho}_t, \qquad \dot{\boldsymbol{\rho}}_t = \nabla \log \mu(\boldsymbol{\theta}_t).$$

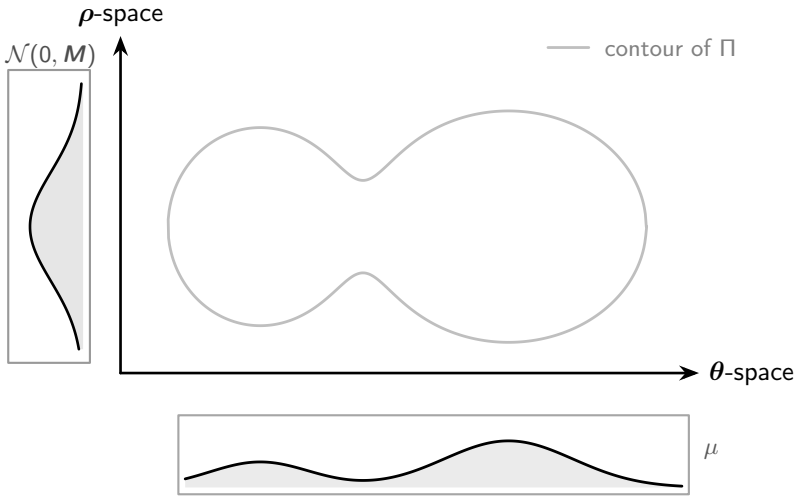
The geometry behind Hamiltonian Monte Carlo



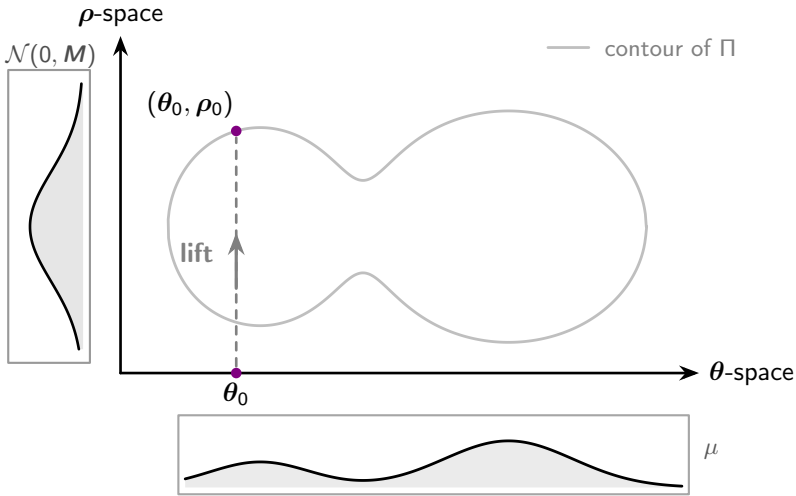
The geometry behind Hamiltonian Monte Carlo



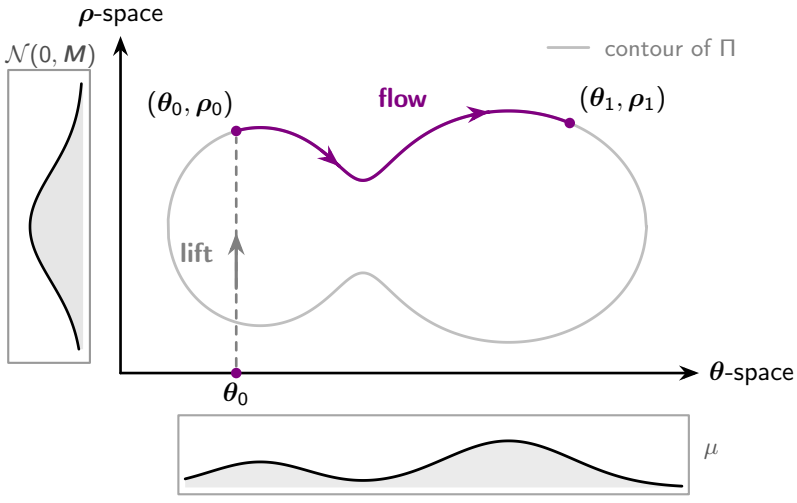
The geometry behind Hamiltonian Monte Carlo



The geometry behind Hamiltonian Monte Carlo



The geometry behind Hamiltonian Monte Carlo



Leapfrog

- Split into solvable flows:

Leapfrog

- Split into solvable flows:

$$\underbrace{\begin{cases} \dot{\boldsymbol{\theta}}_t = \boldsymbol{M}^{-1}\boldsymbol{\rho}_t, \\ \dot{\boldsymbol{\rho}}_t = 0 \end{cases}}_{\text{A–flow (drift)}}$$

Leapfrog

- Split into solvable flows:

$$\underbrace{\begin{cases} \dot{\boldsymbol{\theta}}_t = \boldsymbol{M}^{-1}\boldsymbol{\rho}_t, \\ \dot{\boldsymbol{\rho}}_t = 0 \end{cases}}_{\text{A–flow (drift)}} \quad \text{and} \quad \underbrace{\begin{cases} \dot{\boldsymbol{\theta}}_t = 0, \\ \dot{\boldsymbol{\rho}}_t = \nabla \log \mu(\boldsymbol{\theta}_t) \end{cases}}_{\text{B–flow (kick)}}$$

Leapfrog

- Split into solvable flows:

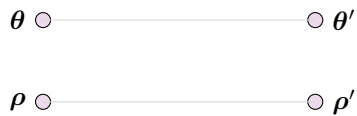
$$\underbrace{\begin{cases} \dot{\boldsymbol{\theta}}_t = \boldsymbol{M}^{-1}\boldsymbol{\rho}_t, \\ \dot{\boldsymbol{\rho}}_t = 0 \end{cases}}_{\text{A–flow (drift)}} \quad \text{and} \quad \underbrace{\begin{cases} \dot{\boldsymbol{\theta}}_t = 0, \\ \dot{\boldsymbol{\rho}}_t = \nabla \log \mu(\boldsymbol{\theta}_t) \end{cases}}_{\text{B–flow (kick)}}$$

- **Leapfrog (step size $h > 0$):**

$$\boxed{\Phi_h = B_{h/2} \circ A_h \circ B_{h/2}}$$

Leapfrog

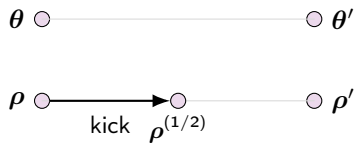
- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$



Leapfrog

- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$

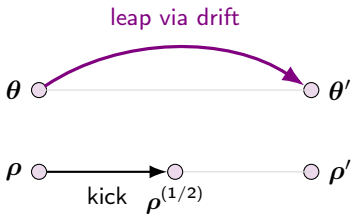
$$\rho^{(1/2)} = \rho + \tfrac{h}{2} \nabla \log \mu(\theta)$$



Leapfrog

- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$

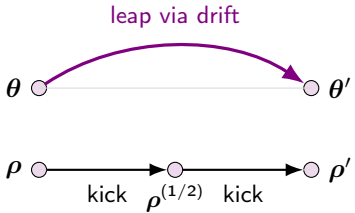
$$\rho^{(1/2)} = \rho + \tfrac{h}{2} \nabla \log \mu(\theta)$$
$$\theta' = \theta + h \boldsymbol{M}^{-1} \rho^{(1/2)}$$



Leapfrog

- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$

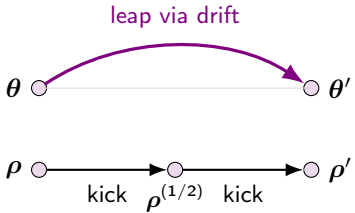
$$\rho^{(1/2)} = \rho + \tfrac{h}{2} \nabla \log \mu(\theta)$$
$$\theta' = \theta + h \boldsymbol{M}^{-1} \rho^{(1/2)}$$
$$\rho' = \rho^{(1/2)} + \tfrac{h}{2} \nabla \log \mu(\theta')$$



Leapfrog

- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$

$$\rho^{(1/2)} = \rho + \tfrac{h}{2} \nabla \log \mu(\theta)$$
$$\theta' = \theta + h \boldsymbol{M}^{-1} \rho^{(1/2)}$$
$$\rho' = \rho^{(1/2)} + \tfrac{h}{2} \nabla \log \mu(\theta')$$

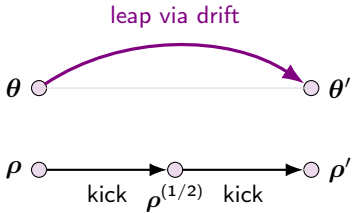


- **Key properties**
  - explicit, cheap, volume-preserving, and $\Phi_h^{-1} = \Phi_{-h}$

Leapfrog

- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$

$$\rho^{(1/2)} = \rho + \tfrac{h}{2} \nabla \log \mu(\theta)$$
$$\theta' = \theta + h \boldsymbol{M}^{-1} \rho^{(1/2)}$$
$$\rho' = \rho^{(1/2)} + \tfrac{h}{2} \nabla \log \mu(\theta')$$

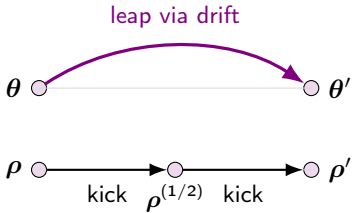


- **Key properties**
  - explicit, cheap, volume-preserving, and $\Phi_h^{-1} = \Phi_{-h}$
  - does not preserve the lifted density: $\Pi \circ \Phi_h \neq \Pi$

Leapfrog

- **Leapfrog map:** $(\theta', \rho') = \Phi_h(\theta, \rho)$

$$\rho^{(1/2)} = \rho + \tfrac{h}{2} \nabla \log \mu(\theta)$$
$$\theta' = \theta + h \boldsymbol{M}^{-1} \rho^{(1/2)}$$
$$\rho' = \rho^{(1/2)} + \tfrac{h}{2} \nabla \log \mu(\theta')$$



- **Key properties**
  - explicit, cheap, volume-preserving, and $\Phi_h^{-1} = \Phi_{-h}$
  - does not preserve the lifted density: $\Pi \circ \Phi_h \neq \Pi$
  - only conditionally stable

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

[†](Duane et al (1987); Neal (2011))

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

$L \in \mathbb{N}$ (integration time), $h > 0$ (step size), $\boldsymbol{M} \succ 0$ (mass matrix), $\alpha \in (0, 1]$ (refreshment).

[†](Duane et al (1987); Neal (2011))

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

$L \in \mathbb{N}$ (integration time), $h > 0$ (step size), $\boldsymbol{M} \succ 0$ (mass matrix), $\alpha \in (0, 1]$ (refreshment).

[†](Duane et al (1987); Neal (2011))

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

$L \in \mathbb{N}$ (integration time), $h > 0$ (step size), $\boldsymbol{M} \succ 0$ (mass matrix), $\alpha \in (0, 1]$ (refreshment).

[†](Duane et al (1987); Neal (2011))

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

$L \in \mathbb{N}$ (integration time), $h > 0$ (step size), $\boldsymbol{M} \succ 0$ (mass matrix), $\alpha \in (0, 1]$ (refreshment).

1. **Partial momentum refreshment**

$$\boldsymbol{\rho}^{(0)} = \sqrt{1 - \alpha^2}\, \boldsymbol{\rho}_n + \alpha\, \boldsymbol{\xi}_n, \qquad \boldsymbol{\xi}_n \sim \mathcal{N}(0, \boldsymbol{M}).$$

[†](Duane et al (1987); Neal (2011))

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

$L \in \mathbb{N}$ (integration time), $h > 0$ (step size), $\boldsymbol{M} \succ 0$ (mass matrix), $\alpha \in (0, 1]$ (refreshment).

1. **Partial momentum refreshment**

$$\boldsymbol{\rho}^{(0)} = \sqrt{1 - \alpha^2}\, \boldsymbol{\rho}_n + \alpha\, \boldsymbol{\xi}_n, \qquad \boldsymbol{\xi}_n \sim \mathcal{N}(0, \boldsymbol{M}).$$

2. **Leapfrog flow**

$$(\boldsymbol{\theta}^\star, \boldsymbol{\rho}^\star) = \Phi_h^L(\boldsymbol{\theta}_n, \boldsymbol{\rho}^{(0)}).$$

[†](Duane et al (1987); Neal (2011))

HMC: One Transition[†]

Requires: current state $(\boldsymbol{\theta}_n, \boldsymbol{\rho}_n)$

$L \in \mathbb{N}$ (integration time), $h > 0$ (step size), $\boldsymbol{M} \succ 0$ (mass matrix), $\alpha \in (0, 1]$ (refreshment).

1. **Partial momentum refreshment**

$$\boldsymbol{\rho}^{(0)} = \sqrt{1 - \alpha^2}\, \boldsymbol{\rho}_n + \alpha\, \boldsymbol{\xi}_n, \qquad \boldsymbol{\xi}_n \sim \mathcal{N}(0, \boldsymbol{M}).$$

2. **Leapfrog flow**

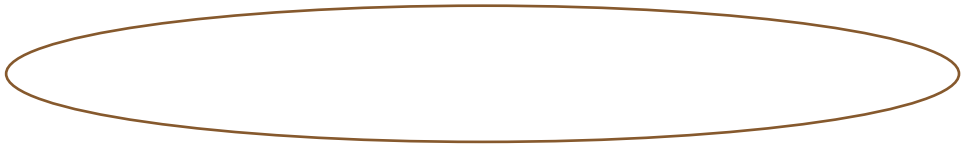
$$(\boldsymbol{\theta}^\star, \boldsymbol{\rho}^\star) = \Phi_h^L(\boldsymbol{\theta}_n, \boldsymbol{\rho}^{(0)}).$$

3. **Metropolis correction**

$$(\boldsymbol{\theta}_{n+1}, \boldsymbol{\rho}_{n+1}) = \begin{cases} (\boldsymbol{\theta}^\star, \boldsymbol{\rho}^\star) & \text{with probability } \exp\big(-[H(\boldsymbol{\theta}^\star, \boldsymbol{\rho}^\star) - H(\boldsymbol{\theta}_n, \boldsymbol{\rho}^{(0)})]^+\big), \\ (\boldsymbol{\theta}_n, -\boldsymbol{\rho}^{(0)}) & \text{otherwise.} \end{cases}$$

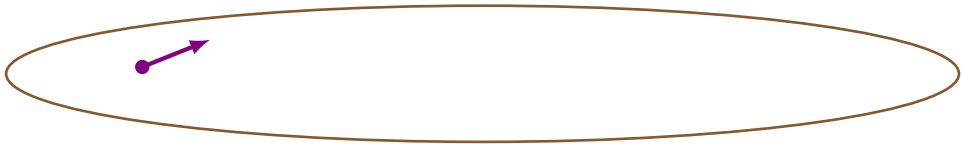
[†](Duane et al (1987); Neal (2011))

Why Integration Time Tuning is Tricky in HMC[†]



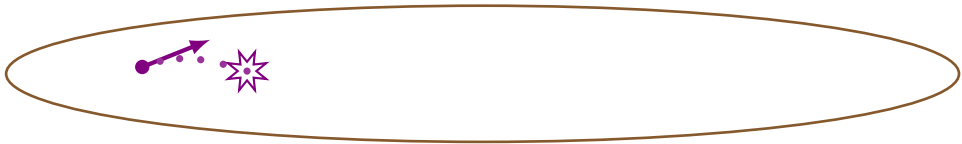
[†](Mackenzie (1989))

Why Integration Time Tuning is Tricky in HMC[†]



[†](Mackenzie (1989))

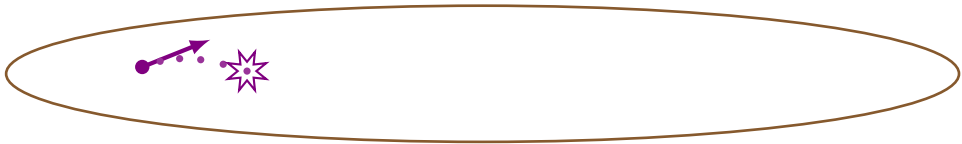
Why Integration Time Tuning is Tricky in HMC[†]



Too short: insufficient integration leads to random-walk behavior

[†](Mackenzie (1989))

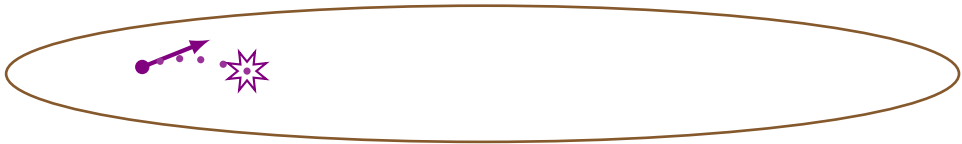
Why Integration Time Tuning is Tricky in HMC[†]



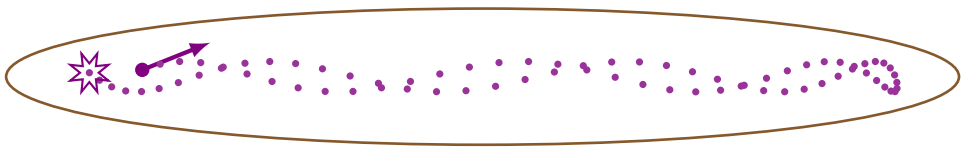
Too short: insufficient integration leads to random-walk behavior

[†](Mackenzie (1989))

Why Integration Time Tuning is Tricky in HMC[†]



Too short: insufficient integration leads to random-walk behavior



Too long: trajectory loops back, wasting computation

[†](Mackenzie (1989))

Sample from the orbit, not just the endpoint[†]

- **Key shift:** do not just propose the endpoint. Instead:

[†]Related to Multiple Proposal MCMC (Calderhead (2014))

Sample from the orbit, not just the endpoint[†]

- **Key shift:** do not just propose the endpoint. Instead:
  - sample an orbit $\mathcal{O}$ containing z_0,

[†]Related to Multiple Proposal MCMC (Calderhead (2014))

Sample from the orbit, not just the endpoint[†]

- **Key shift:** do not just propose the endpoint. Instead:
  - sample an orbit $\mathcal{O}$ containing $\boldsymbol{z}_0$,
  - then sample the next state $\boldsymbol{z}' \in \mathcal{O}$.

[†]Related to Multiple Proposal MCMC (Calderhead (2014))

Sample from the orbit, not just the endpoint[†]

- **Key shift:** do not just propose the endpoint. Instead:
  - sample an orbit $\mathcal{O}$ containing z_0,
  - then sample the next state $z' \in \mathcal{O}$.
- Starting from $z_0 = (\theta_0, \rho_0)$, the leapfrog map, $z_k = \Phi_h^k(z_0)$ for $k \in \mathbb{Z}$, generates

$$z_0$$

[†]Related to Multiple Proposal MCMC (Calderhead (2014))

Sample from the orbit, not just the endpoint[†]

- **Key shift:** do not just propose the endpoint. Instead:
  - sample an orbit $\mathcal{O}$ containing $\boldsymbol{z}_0$,
  - then sample the next state $\boldsymbol{z}' \in \mathcal{O}$.
- Starting from $\boldsymbol{z}_0 = (\boldsymbol{\theta}_0, \boldsymbol{\rho}_0)$, the leapfrog map, $\boldsymbol{z}_k = \Phi_h^k(\boldsymbol{z}_0)$ for $k \in \mathbb{Z}$, generates

$$\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots$$

[†]Related to Multiple Proposal MCMC (Calderhead (2014))

Sample from the orbit, not just the endpoint†

- **Key shift:** do not just propose the endpoint. Instead:
  - sample an orbit $\mathcal{O}$ containing $\boldsymbol{z}_0$,
  - then sample the next state $\boldsymbol{z}' \in \mathcal{O}$.
- Starting from $\boldsymbol{z}_0 = (\boldsymbol{\theta}_0, \boldsymbol{\rho}_0)$, the leapfrog map, $\boldsymbol{z}_k = \Phi_h^k(\boldsymbol{z}_0)$ for $k \in \mathbb{Z}$, generates

$$\ldots, \boldsymbol{z}_{-2}, \boldsymbol{z}_{-1}, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots$$

†Related to Multiple Proposal MCMC (Calderhead (2014))

Sample from the orbit, not just the endpoint[†]

- **Key shift:** do not just propose the endpoint. Instead:
  - sample an orbit $\mathcal{O}$ containing z_0,
  - then sample the next state $z' \in \mathcal{O}$.
- Starting from $z_0 = (\theta_0, \rho_0)$, the leapfrog map, $z_k = \Phi_h^k(z_0)$ for $k \in \mathbb{Z}$, generates

$$\ldots, z_{-2}, z_{-1}, z_0, z_1, z_2, \ldots$$

- An **orbit** is a consecutive block of this trajectory indexed by $a : b = \{a, a+1, \ldots, b\}$

$$\mathcal{O} = \{z_a, \ldots, z_b\}, \qquad |\mathcal{O}| = b - a + 1.$$

[†]Related to Multiple Proposal MCMC (Calderhead (2014))

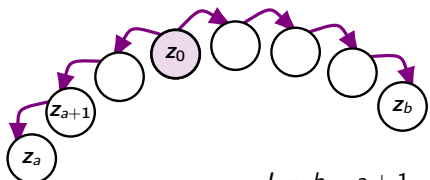
Choosing the orbit

Choosing the orbit

- Fix $L \in \mathbb{N}$.

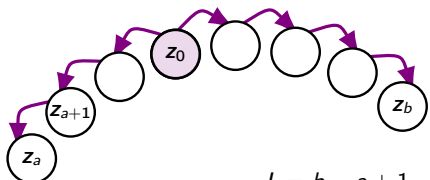
Choosing the orbit

- Fix $L \in \mathbb{N}$. Consider the set $\mathcal{C}_L(z_0)$ of all orbits of length L containing z_0.
  - **Example orbit** $\mathcal{O} \in \mathcal{C}_L(z_0)$**:**



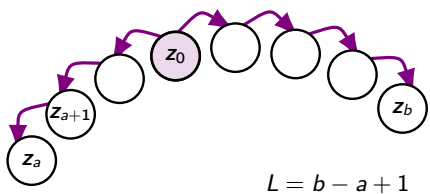
Choosing the orbit

- Fix $L \in \mathbb{N}$. Consider the set $\mathcal{C}_L(z_0)$ of all orbits of length L containing z_0.
  - **Example orbit** $\mathcal{O} \in \mathcal{C}_L(z_0)$**:**



Choosing the orbit

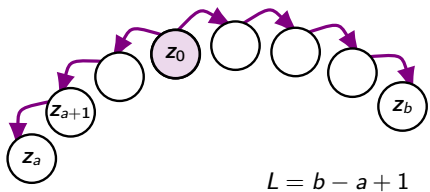
- Fix $L \in \mathbb{N}$. Consider the set $\mathcal{C}_L(\boldsymbol{z}_0)$ of all orbits of length L containing $\boldsymbol{z}_0$.
  - **Example orbit** $\mathcal{O} \in \mathcal{C}_L(\boldsymbol{z}_0)$**:**



- **Initial-point symmetry:** $\mathbb{P}(\mathcal{O} \mid \boldsymbol{z})$ is invariant w.r.t. $\boldsymbol{z} \in \mathcal{O}$.

Choosing the orbit

- Fix $L \in \mathbb{N}$. Consider the set $\mathcal{C}_L(\boldsymbol{z}_0)$ of all orbits of length L containing $\boldsymbol{z}_0$.
  - **Example orbit** $\mathcal{O} \in \mathcal{C}_L(\boldsymbol{z}_0)$:

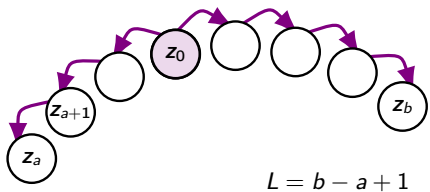


- **Initial-point symmetry:** $\mathbb{P}(\mathcal{O} \mid \boldsymbol{z})$ is invariant w.r.t. $\boldsymbol{z} \in \mathcal{O}$.

$$\boxed{\mathcal{O} \sim \mathrm{Uniform}\big(\mathcal{C}_L(\boldsymbol{z}_0)\big)}$$

Choosing the orbit

- Fix $L \in \mathbb{N}$. Consider the set $\mathcal{C}_L(\boldsymbol{z}_0)$ of all orbits of length L containing $\boldsymbol{z}_0$.
  - **Example orbit** $\mathcal{O} \in \mathcal{C}_L(\boldsymbol{z}_0)$:



- **Initial-point symmetry:** $\mathbb{P}(\mathcal{O} \mid \boldsymbol{z})$ is invariant w.r.t. $\boldsymbol{z} \in \mathcal{O}$.

$$\boxed{\mathcal{O} \sim \mathrm{Uniform}\big(\mathcal{C}_L(\boldsymbol{z}_0)\big)} \quad \Longrightarrow \quad \text{initial-point symmetry}$$

Sampling from the orbit

- To complete the transition we must sample

$$\boldsymbol{z}' \sim \mathrm{Categorical}\big(\mathcal{O};\, \Pi\big).$$

Sampling from the orbit

- To complete the transition we must sample

$$\boldsymbol{z}' \sim \mathrm{Categorical}\big(\mathcal{O};\, \Pi\big).$$

- Naive implementation: construct the entire orbit and then sample.

Requires storing L states.

Sampling from the orbit

- To complete the transition we must sample

$$\boldsymbol{z}' \sim \mathrm{Categorical}\big(\mathcal{O};\, \Pi\big).$$

- Naive implementation: construct the entire orbit and then sample.

Requires storing L states.

- **Goal:** $O(\log L)$ storage.

Sampling from the orbit

- To complete the transition we must sample

$$z' \sim \mathrm{Categorical}\big(\mathcal{O};\, \Pi\big).$$

- Naive implementation: construct the entire orbit and then sample.

Requires storing L states.

- **Goal:** $O(\log L)$ storage.

- **Strategy:** use a geometric progression.

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}$.

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \mathrm{Uniform}\{0, \ldots, L-1\}.$$

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \text{Uniform}\{0, \ldots, L-1\}.$$

$$B_1, \ldots, B_m \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1/2),$$

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \mathrm{Uniform}\{0, \ldots, L-1\}.$$

$$B_1, \ldots, B_m \overset{\text{i.i.d.}}{\sim} \mathrm{Bernoulli}(1/2),$$

$$\text{uniform } b \implies \text{uniform } \mathcal{O}$$

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \text{Uniform}\{0, \ldots, L-1\}.$$

$$B_1, \ldots, B_m \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1/2),$$

uniform $b \implies$ uniform $\mathcal{O}$

start at z_0

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

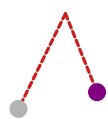
$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \text{Uniform}\{0, \ldots, L-1\}.$$

$$B_1, \ldots, B_m \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1/2),$$

$$\text{uniform } b \implies \text{uniform } \mathcal{O}$$



first doubling: $B_1 = 1$

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

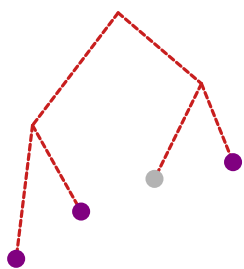
$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \text{Uniform}\{0, \ldots, L-1\}.$$

$$B_1, \ldots, B_m \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1/2),$$

uniform $b \implies$ uniform $\mathcal{O}$



second doubling: $B_2 = 0$

Uniform orbit via stochastic doubling

Orbit parametrization. Fix $L = 2^m$.

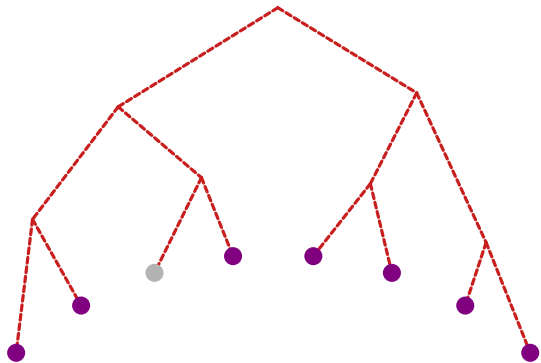
$$\mathcal{O} = (z_a, \ldots, z_b) \longleftrightarrow b \in \{0, \ldots, L-1\}.$$

Sampling scheme.

$$b = \sum_{i=1}^{m} B_i \, 2^{i-1} \sim \text{Uniform}\{0, \ldots, L-1\}.$$

$$B_1, \ldots, B_m \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1/2),$$

$$\text{uniform } b \implies \text{uniform } \mathcal{O}$$



third doubling: $B_3 = 1$

Progressive sampling along the doubling tree

Lemma (2-block merge)

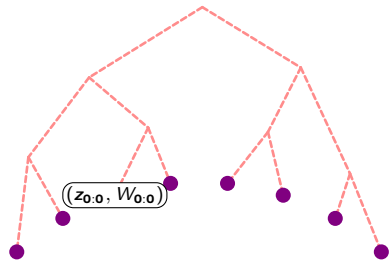
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



store (z_0, W_0)

Progressive sampling along the doubling tree

Lemma (2-block merge)

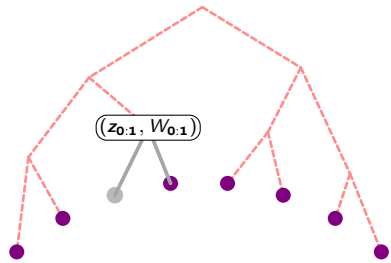
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



merge to $(z_{0:1}, W_{0:1})$

Progressive sampling along the doubling tree

Lemma (2-block merge)

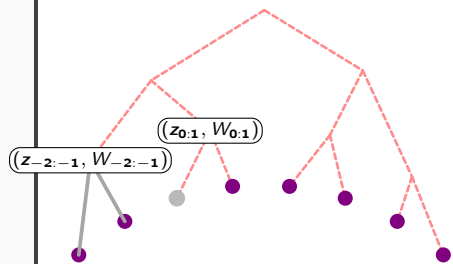
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



also store $(z_{-2:-1}, W_{-2:-1})$

Progressive sampling along the doubling tree

Lemma (2-block merge)

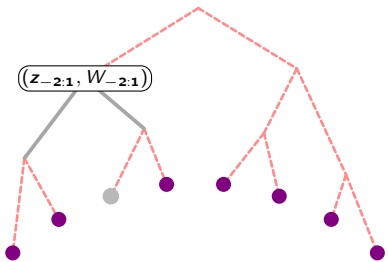
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



merge to $(z_{-2:1}, W_{-2:1})$

Progressive sampling along the doubling tree

Lemma (2-block merge)

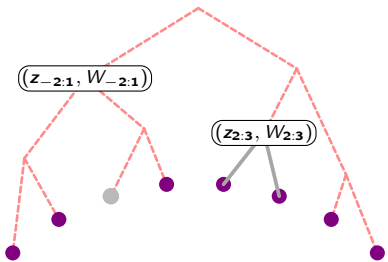
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



also store $(z_{2:3}, W_{2:3})$

Progressive sampling along the doubling tree

Lemma (2-block merge)

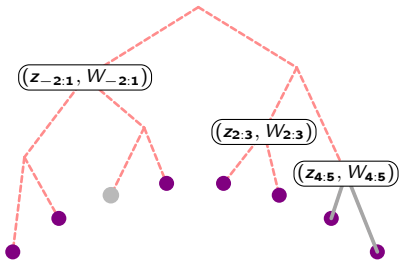
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



peak storage: 3 summaries

Progressive sampling along the doubling tree

Lemma (2-block merge)

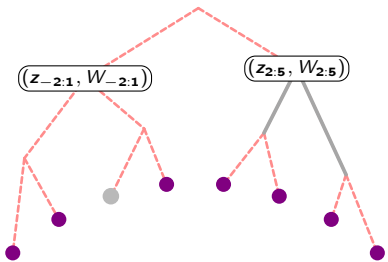
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \text{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \text{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \text{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



merge to $(z_{2:5}, W_{2:5})$

Progressive sampling along the doubling tree

Lemma (2-block merge)

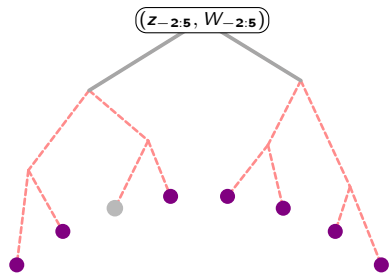
Let

$$W(A) = \sum_{z \in A} \Pi(z) \quad \textit{and} \quad W(B) = \sum_{z \in B} \Pi(z).$$

If $z_A \sim \mathrm{Cat}(A; \Pi)$ *and* $z_B \sim \mathrm{Cat}(B; \Pi)$ *independently, then*

$$z^\star = \begin{cases} z_B & \textit{with prob. } \dfrac{W(B)}{W(A) + W(B)}, \\ z_A & \textit{otherwise} \end{cases}$$

satisfies $z^\star \sim \mathrm{Cat}(A \cup B; \Pi)$.



final summary $(z_{-2:5}, W_{-2:5})$

Biased Progressive HMC (bpHMC)

- Progressive sampling yields

$$\boldsymbol{z}' \sim \mathrm{Cat}(\mathcal{O}; \Pi)$$

Biased Progressive HMC (bpHMC)

- Progressive sampling yields

$$\boldsymbol{z}' \sim \mathrm{Cat}(\mathcal{O}; \Pi)$$

- No leapfrog error $\Rightarrow$ Π is uniform on $\mathcal{O}$

$$\boldsymbol{z}' \sim \mathrm{Uniform}(\mathcal{O})$$

Biased Progressive HMC (bpHMC)

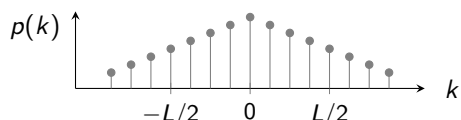
- Progressive sampling yields

$$\boldsymbol{z}' \sim \mathrm{Cat}(\mathcal{O}; \Pi)$$

- No leapfrog error $\Rightarrow$ Π is uniform on $\mathcal{O}$

$$\boldsymbol{z}' \sim \mathrm{Uniform}(\mathcal{O})$$

$$\frac{W(B)}{W(A) + W(B)} \quad \text{(Barker)}$$



Biased Progressive HMC (bpHMC)

- Progressive sampling yields

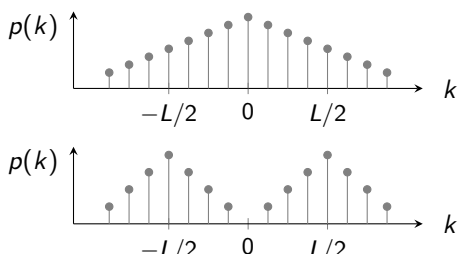
$$\boldsymbol{z}' \sim \mathrm{Cat}(\mathcal{O}; \Pi)$$

- No leapfrog error $\Rightarrow \Pi$ is uniform on $\mathcal{O}$

$$\boldsymbol{z}' \sim \mathrm{Uniform}(\mathcal{O})$$

$$\frac{W(B)}{W(A) + W(B)} \quad \text{(Barker)}$$

$$\boxed{\min\left(1, \frac{W(B)}{W(A)}\right) \quad \text{(Metropolis)}}$$



The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The No-U-Turn Sampler (NUTS)[†]

- Adds a no-U-turn rule to bpHMC to stop doubling while preserving initial-point symmetry.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

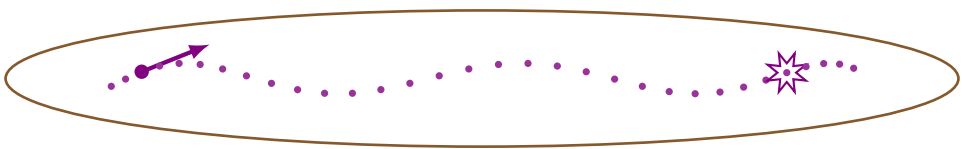
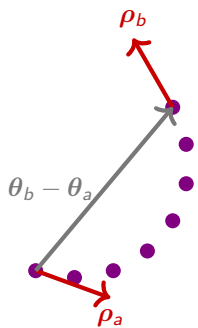


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

[†](Hoffman & Gelman, 2011; Betancourt, 2017)

The U-turn condition

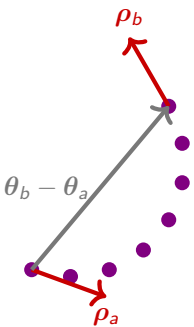
no U-turn



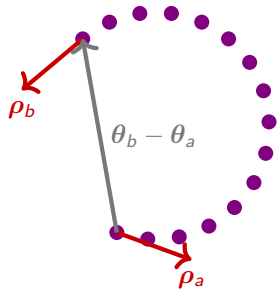
$$\rho_a \cdot (\theta_b - \theta_a) < 0 \qquad \text{or} \qquad \rho_b \cdot (\theta_b - \theta_a) < 0$$

The U-turn condition

no U-turn

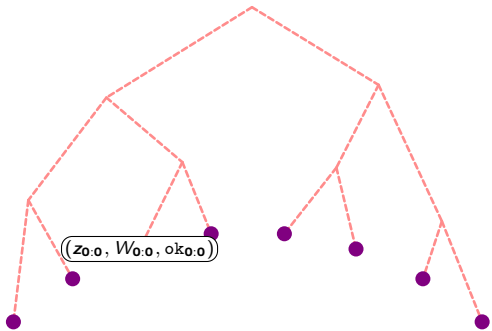


U-turn detected



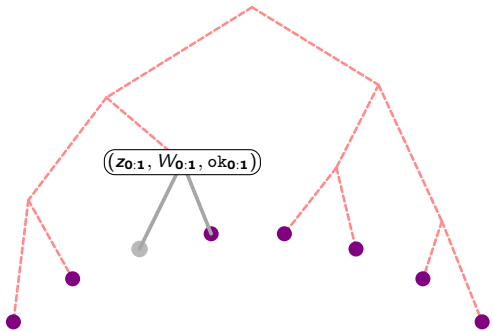
$$\rho_a \cdot (\theta_b - \theta_a) < 0 \qquad \text{or} \qquad \rho_b \cdot (\theta_b - \theta_a) < 0$$

NUTS: recursive U-turn checks (no extra memory)



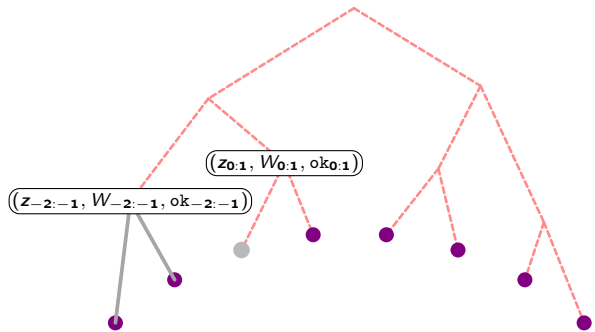
$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

NUTS: recursive U-turn checks (no extra memory)



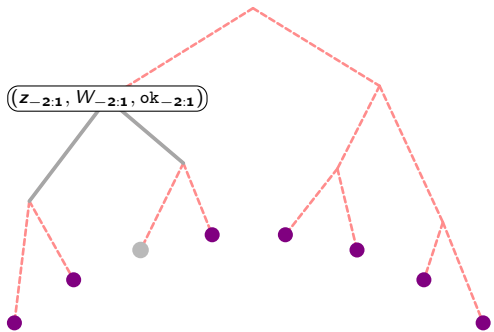
$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

NUTS: recursive U-turn checks (no extra memory)



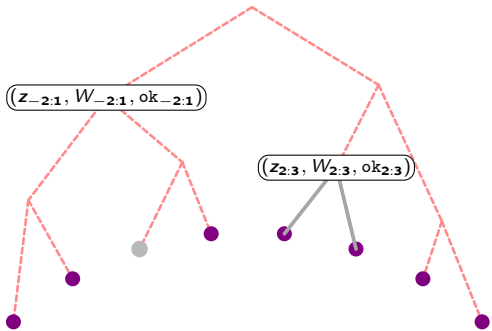
$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

NUTS: recursive U-turn checks (no extra memory)



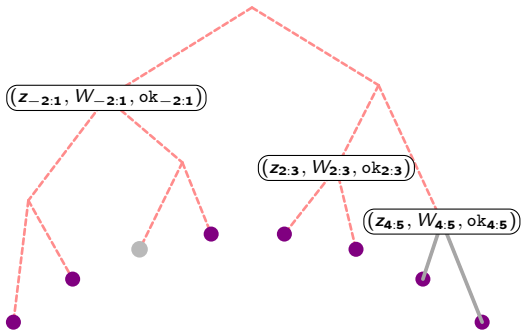
$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

NUTS: recursive U-turn checks (no extra memory)



$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

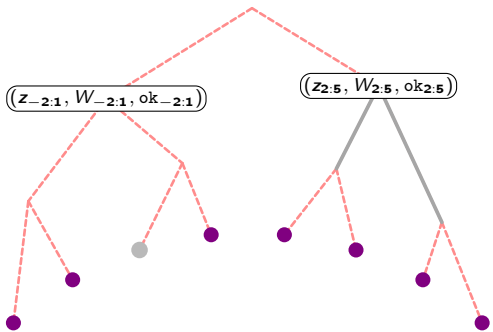
NUTS: recursive U-turn checks (no extra memory)



peak storage unchanged

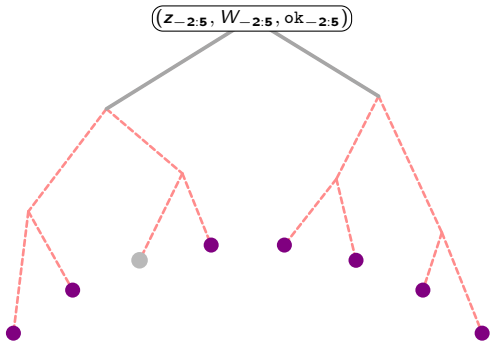
$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

NUTS: recursive U-turn checks (no extra memory)



$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

NUTS: recursive U-turn checks (no extra memory)



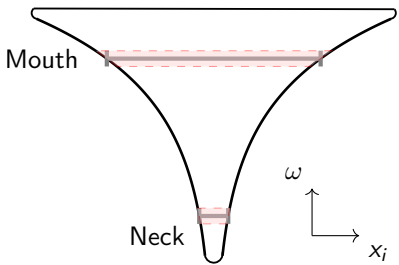
final orbit is U-turn free on every subtree

$\mathrm{ok}_{a:b} = 1$ iff no U-turn occurs in the subtree rooted at $\mathcal{O}_{a:b}$ or in any subtree below it

Neal's Funnel: $\mu(\omega, \boldsymbol{x}) = \mathcal{N}(\omega \mid 0, 9)\, \mathcal{N}(\boldsymbol{x} \mid 0, \exp(\omega)\, \boldsymbol{I}_{d \times d})$

Neal's Funnel: $\mu(\omega, \boldsymbol{x}) = \mathcal{N}(\omega \mid 0, 9)\, \mathcal{N}(\boldsymbol{x} \mid 0, \exp(\omega)\, \boldsymbol{I}_{d \times d})$

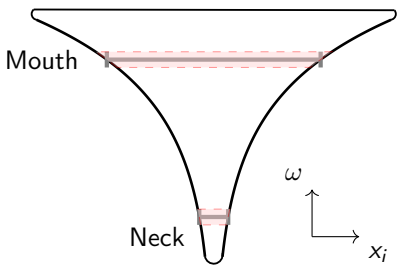
- **However:** some important targets exhibit extreme variations in scale, e.g., Neal's funnel.[†]

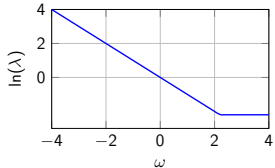


[†](Neal, 2003; Betancourt & Girolami, 2013)

Neal's Funnel: $\mu(\omega, \boldsymbol{x}) = \mathcal{N}(\omega \mid 0, 9)\, \mathcal{N}(\boldsymbol{x} \mid 0, \exp(\omega)\, \boldsymbol{I}_{d \times d})$

- **However:** some important targets exhibit extreme variations in scale, e.g., Neal's funnel.[†]



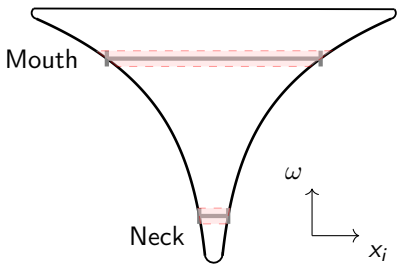


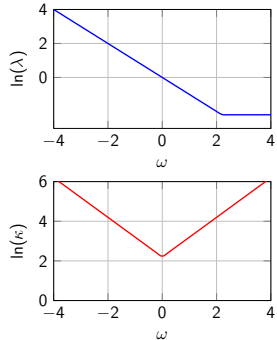
- **Top right:** Spectral radius $\lambda(\omega) = \max(1/9, e^{-\omega})$ grows in the neck.

[†](Neal, 2003; Betancourt & Girolami, 2013)

Neal's Funnel: $\mu(\omega, \boldsymbol{x}) = \mathcal{N}(\omega \mid 0, 9)\, \mathcal{N}(\boldsymbol{x} \mid 0, \exp(\omega)\, \boldsymbol{I}_{d \times d})$

- **However:** some important targets exhibit extreme variations in scale, e.g., Neal's funnel.[†]

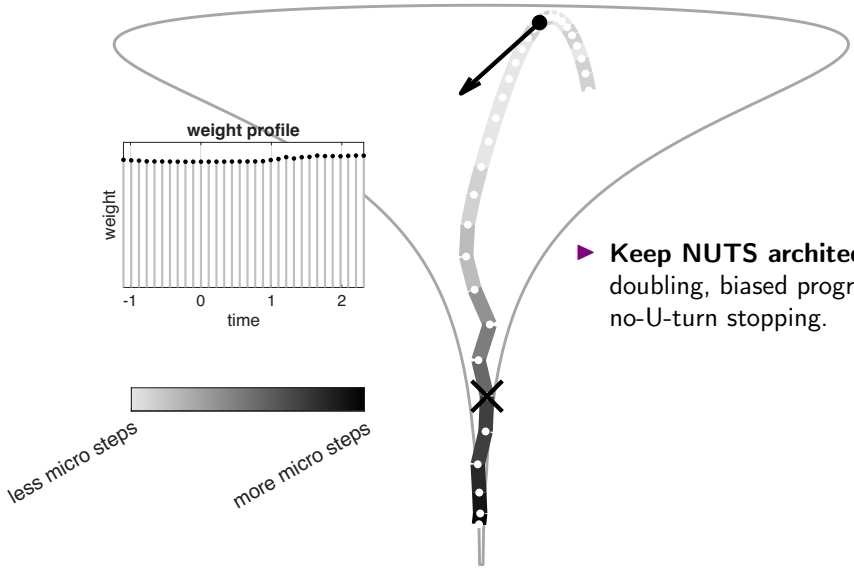




- **Top right:** Spectral radius $\lambda(\omega) = \max(1/9, e^{-\omega})$ grows in the neck.
- **Bottom right:** Condition number $\kappa(\omega) = 9 \cdot \max(e^{\omega}, e^{-\omega})$ grows sharply with $|\omega|$.

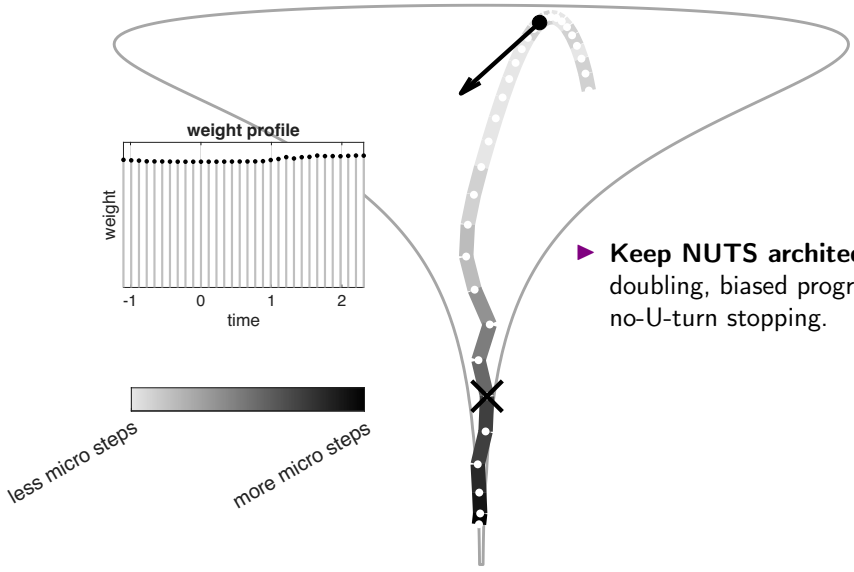
[†](Neal, 2003; Betancourt & Girolami, 2013)

Within-orbit Adaptive Leapfrog No-U-Turn Sampler



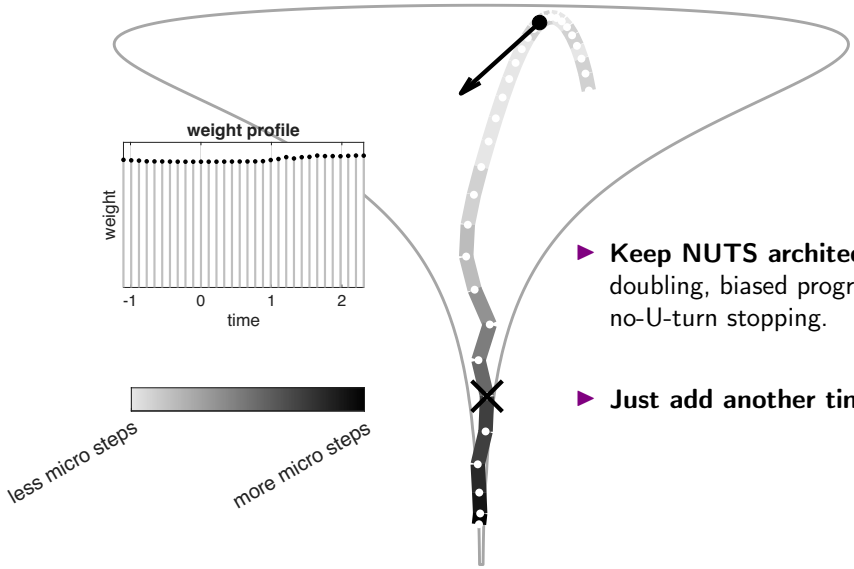
- **Keep NUTS architecture:** stochastic doubling, biased progressive sampling, & no-U-turn stopping.

Within-orbit Adaptive Leapfrog No-U-Turn Sampler



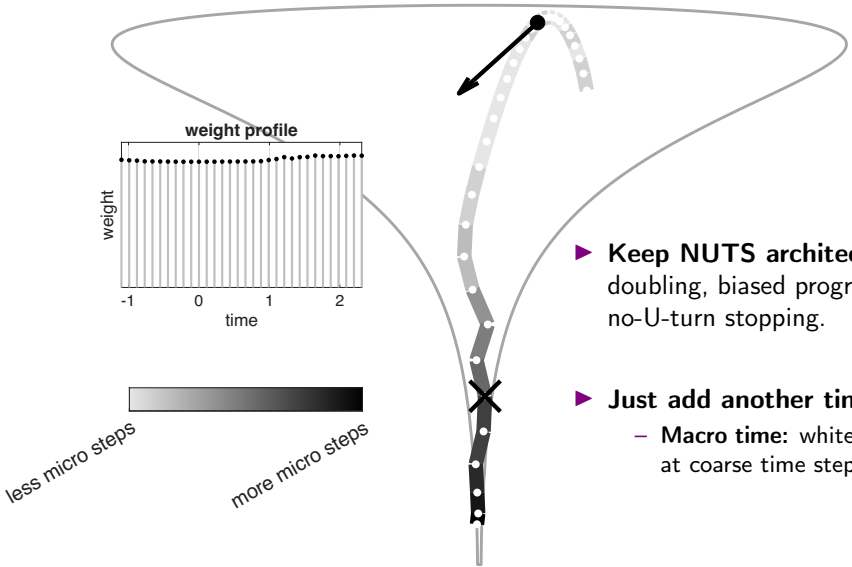
- **Keep NUTS architecture:** stochastic doubling, biased progressive sampling, & no-U-turn stopping.

Within-orbit Adaptive Leapfrog No-U-Turn Sampler



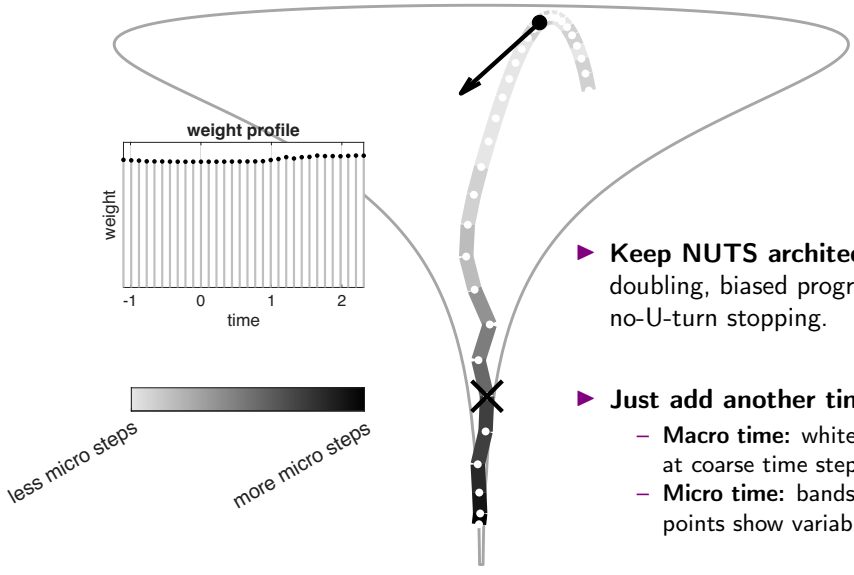
- **Keep NUTS architecture:** stochastic doubling, biased progressive sampling, & no-U-turn stopping.
- **Just add another time-scale:**

Within-orbit Adaptive Leapfrog No-U-Turn Sampler



- **Keep NUTS architecture:** stochastic doubling, biased progressive sampling, & no-U-turn stopping.
- **Just add another time-scale:**
  - **Macro time:** white dots mark positions at coarse time steps t_a, t_{a+1}, ...

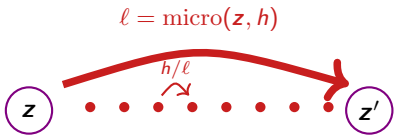
Within-orbit Adaptive Leapfrog No-U-Turn Sampler



- **Keep NUTS architecture:** stochastic doubling, biased progressive sampling, & no-U-turn stopping.
- **Just add another time-scale:**
  - **Macro time:** white dots mark positions at coarse time steps t_a, t_{a+1}, ...
  - **Micro time:** bands between macro points show variable-resolution.

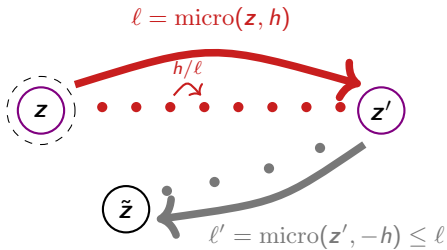
Adaptive Step Size: Breaking and Restoring Symmetry

Deterministic adaptivity



Adaptive Step Size: Breaking and Restoring Symmetry

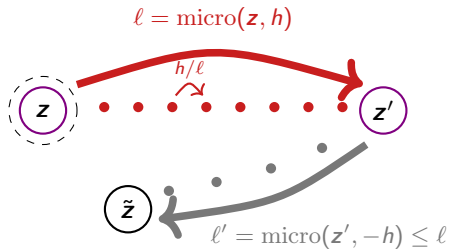
Deterministic adaptivity



backward map need not return to z

Adaptive Step Size: Breaking and Restoring Symmetry

Deterministic adaptivity

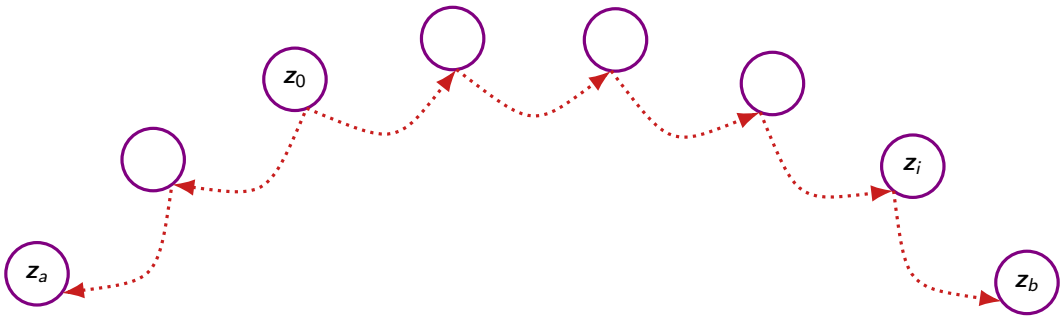


backward map need not return to z

Fix: randomize + reweight

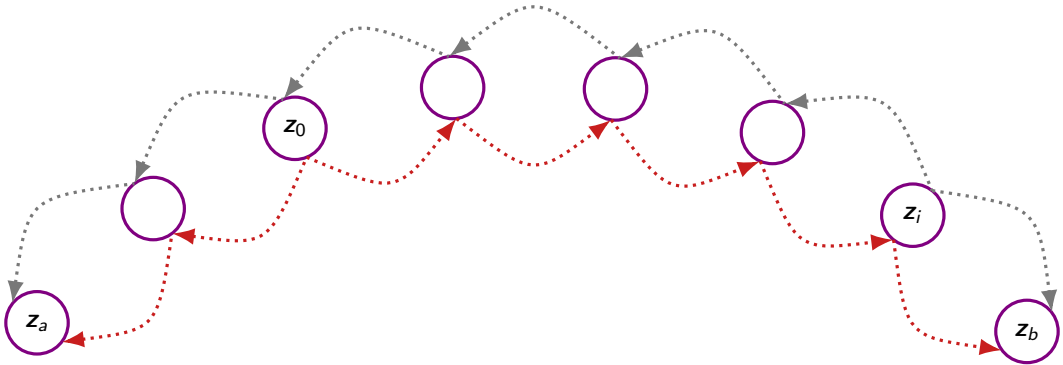
$$\ell \sim p_{\mathrm{micro}}(\cdot \mid z, h), \qquad w' = e^{-H(z')} \frac{p_{\mathrm{micro}}(\ell \mid z', -h)}{p_{\mathrm{micro}}(\ell \mid z, h)}$$

WALNUTS: Orbit Construction



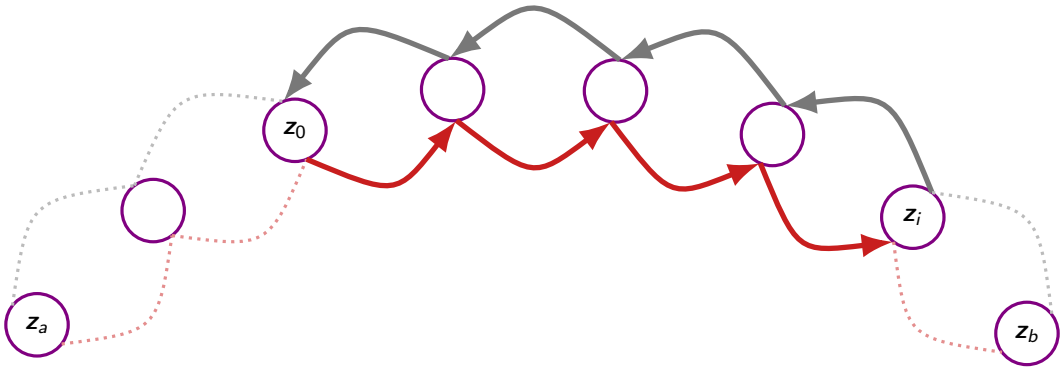
Red: orbit from z_0.

WALNUTS: Orbit Construction



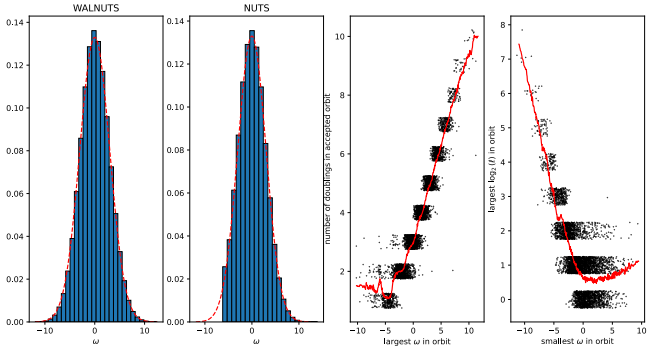
Red: orbit from z_0. **Gray:** orbit from z_i.

WALNUTS: Orbit Construction



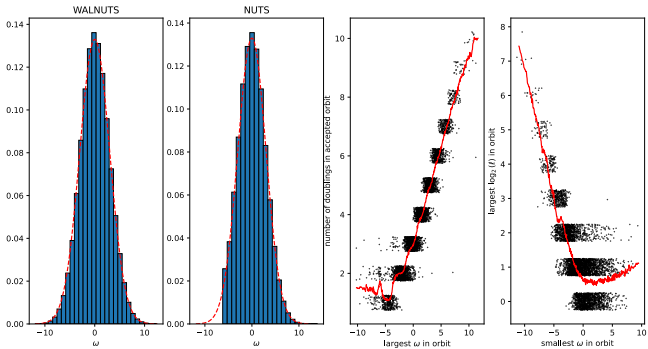
Red: orbit from z_0. **Gray:** orbit from z_i. **Only the segment between z_0 and z_i contributes.**

WALNUTS vs NUTS in Neal's Funnel



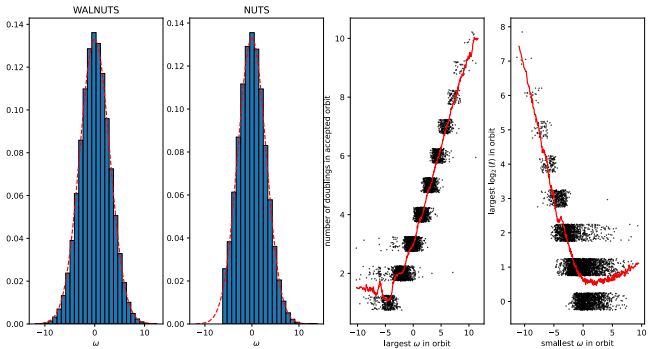
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.

WALNUTS vs NUTS in Neal's Funnel



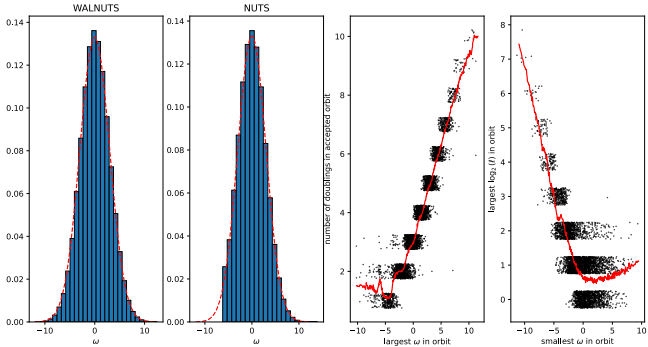
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.

WALNUTS vs NUTS in Neal's Funnel



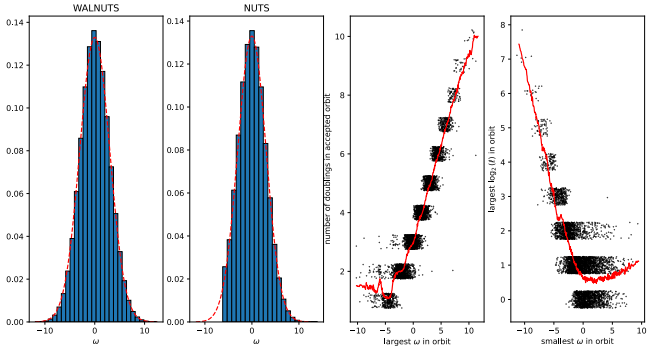
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.

WALNUTS vs NUTS in Neal's Funnel



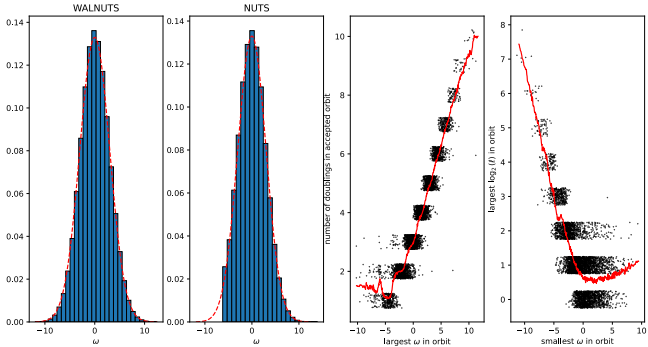
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.
- **Right two panels:** Behavior of adaptive parameters vs. location in the funnel.

WALNUTS vs NUTS in Neal's Funnel



- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.
- **Right two panels:** Behavior of adaptive parameters vs. location in the funnel.
  - Orbit length increases in wide mouth (right tail).

WALNUTS vs NUTS in Neal's Funnel



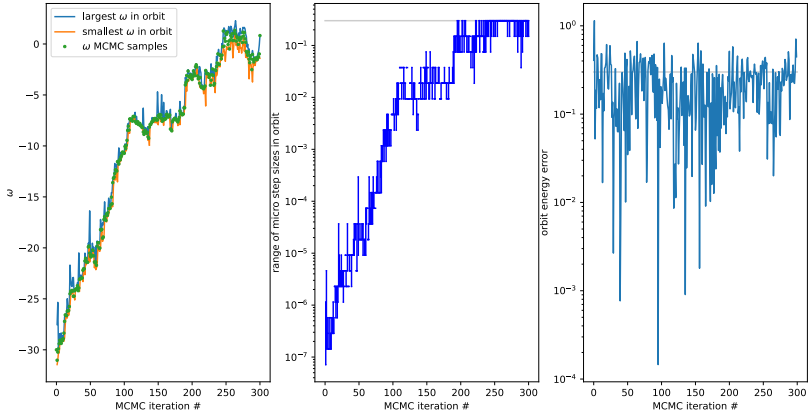
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.
- **Right two panels:** Behavior of adaptive parameters vs. location in the funnel.
  - Orbit length increases in wide mouth (right tail).
  - Micro step size $h\ell^{-1}$ decreases in narrow neck (left tail).

Cold-Start Diagnostics: WALNUTS in Neal's Funnel

- **Setup:** Initialized deep in the neck: $\omega = -30$, $x_i = 0$ for $i = 1, \ldots, 10$.

Cold-Start Diagnostics: WALNUTS in Neal's Funnel

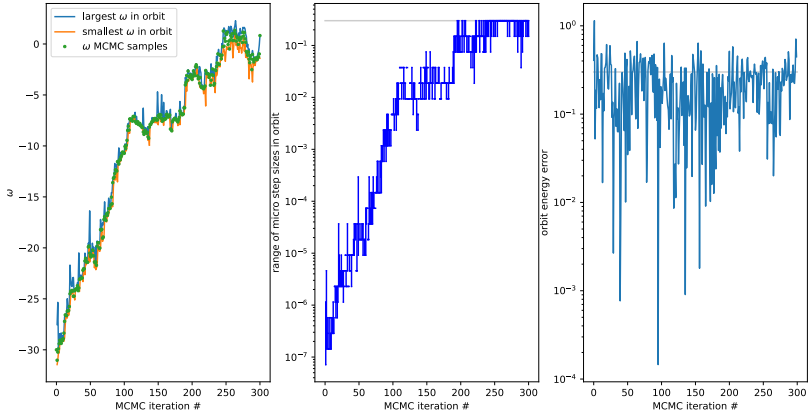
- **Setup:** Initialized deep in the neck: $\omega = -30$, $x_i = 0$ for $i = 1, \ldots, 10$.



- **Left:** WALNUTS samples of ω (green) with orbit spans (lines).

Cold-Start Diagnostics: WALNUTS in Neal's Funnel

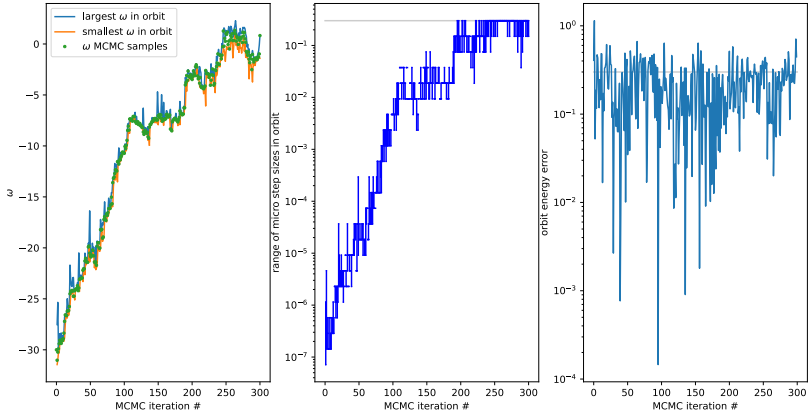
- **Setup:** Initialized deep in the neck: $\omega = -30$, $x_i = 0$ for $i = 1, \ldots, 10$.



- **Left:** WALNUTS samples of ω (green) with orbit spans (lines).
- **Center:** Adaptive micro step sizes $h\ell^{-1}$ per orbit; gray line shows macro step size $h = 0.3$.

Cold-Start Diagnostics: WALNUTS in Neal's Funnel

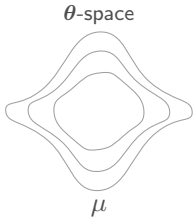
- **Setup:** Initialized deep in the neck: $\omega = -30$, $x_i = 0$ for $i = 1, \ldots, 10$.

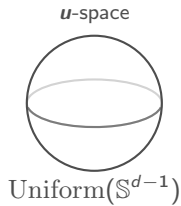


- **Left:** WALNUTS samples of ω (green) with orbit spans (lines).
- **Center:** Adaptive micro step sizes $h\ell^{-1}$ per orbit; gray line shows macro step size $h = 0.3$.
- **Right:** Energy error remains tightly controlled per orbit.

Future: Beyond Hamiltonian Flows

Use alternative flows. For example, isokinetic dynamics[†].

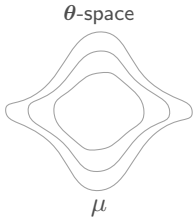


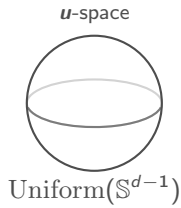


[†](Martyna, Minary & Tuckerman (2003); Fang, Sanz-Serna & Skeel (2014); Robnik, Cohn-Gordon & Seljak (2025))

Future: Beyond Hamiltonian Flows

Use alternative flows. For example, isokinetic dynamics[†].





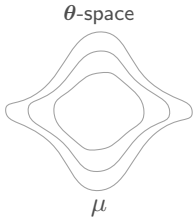
Introduce an auxiliary direction $\boldsymbol{u} \in \mathbb{S}^{d-1}$ and joint distribution over $(\boldsymbol{\theta}, \boldsymbol{u}) \in \mathbb{R}^d \times \mathbb{S}^{d-1}$

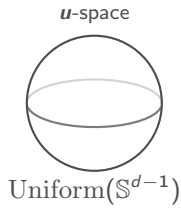
$$\boxed{\Pi = \mu \otimes \mathrm{Uniform}(\mathbb{S}^{d-1})}$$

[†](Martyna, Minary & Tuckerman (2003); Fang, Sanz-Serna & Skeel (2014); Robnik, Cohn-Gordon & Seljak (2025))

Future: Beyond Hamiltonian Flows

Use alternative flows. For example, isokinetic dynamics[†].





Introduce an auxiliary direction $\boldsymbol{u} \in \mathbb{S}^{d-1}$ and joint distribution over $(\boldsymbol{\theta}, \boldsymbol{u}) \in \mathbb{R}^d \times \mathbb{S}^{d-1}$

$$\boxed{\Pi = \mu \otimes \mathrm{Uniform}(\mathbb{S}^{d-1})}$$

Isokinetic flow on $\mathbb{R}^d \times \mathbb{S}^{d-1}$ preserves Π.

[†](Martyna, Minary & Tuckerman (2003); Fang, Sanz-Serna & Skeel (2014); Robnik, Cohn-Gordon & Seljak (2025))

Future: Partial Momentum Refreshment

- **Mechanism.**

$$\boldsymbol{\rho}' = \alpha\boldsymbol{\rho} + \sqrt{1-\alpha^2}\,\boldsymbol{\xi} \quad (\alpha \approx 1: \text{ persistent} \mid \alpha \approx 0: \text{ diffusive})$$

[†](Lu & Wang (2022); Cao, Lu, & Wang (2023))

Future: Partial Momentum Refreshment

- **Mechanism.**

$$\boldsymbol{\rho}' = \alpha\boldsymbol{\rho} + \sqrt{1-\alpha^2}\,\boldsymbol{\xi} \quad (\alpha \approx 1: \text{ persistent} \mid \alpha \approx 0: \text{ diffusive})$$

[†](Lu & Wang (2022); Cao, Lu, & Wang (2023))

Future: Partial Momentum Refreshment

- **Mechanism.**

$$\boldsymbol{\rho}' = \alpha\boldsymbol{\rho} + \sqrt{1-\alpha^2}\,\boldsymbol{\xi} \quad (\alpha \approx 1 : \text{ persistent} \mid \alpha \approx 0 : \text{ diffusive})$$

- **Acceleration.** Underdamped (persistent) dynamics achieve faster mixing[†]:

$$\text{mixing time} \asymp (\text{condition number})^{1/2}$$

[†](Lu & Wang (2022); Cao, Lu, & Wang (2023))

Future: Partial Momentum Refreshment

- **Mechanism.**

$$\boldsymbol{\rho}' = \alpha\boldsymbol{\rho} + \sqrt{1-\alpha^2}\,\boldsymbol{\xi} \quad (\alpha \approx 1 : \text{ persistent} \mid \alpha \approx 0 : \text{ diffusive})$$

- **Acceleration.** Underdamped (persistent) dynamics achieve faster mixing[†]:

$$\text{mixing time} \asymp (\text{condition number})^{1/2}$$

- **Challenge.** A fixed α imposes a single persistence scale:
  - optimal persistence is state-dependent
  - motivates adaptive or randomized α

[†](Lu & Wang (2022); Cao, Lu, & Wang (2023))

Future: Dynamic Preconditioners

- **Challenge.** No natural ordering for matrices $\Rightarrow$ no threshold rule.

.

Future: Dynamic Preconditioners

- **Challenge.** No natural ordering for matrices $\Rightarrow$ no threshold rule.
- **Mass matrix.** Position-dependent preconditioning $M(\theta)$.[1]
  - Capture local curvature (e.g. $\nabla^2 U(\theta)$)
  - Must remain tractable (avoid full Hessians)

.

[1] (Girolami & Calderhead (2011); Hird & Livingstone (2023); Whalley, Paulin & Leimkuhler (2024); Tran & Kleppe (2024))

Future: Dynamic Preconditioners

- **Challenge.** No natural ordering for matrices $\Rightarrow$ no threshold rule.
- **Mass matrix.** Position-dependent preconditioning $M(\theta)$.[1]
  - Capture local curvature (e.g. $\nabla^2 U(\theta)$)
  - Must remain tractable (avoid full Hessians)
- **Ensembles.** Use parallel chains.[2]
  - Estimate covariance
  - Extract dominant directions
  - Precondition along these directions

.

[1] (Girolami & Calderhead (2011); Hird & Livingstone (2023); Whalley, Paulin & Leimkuhler (2024); Tran & Kleppe (2024))

[2] (Goodman & Weare (2010); Chen (2025))

Questions?

- **arXiv:** 2506.18746
- **GitHub:** flatironinstitute/walnuts

Adaptive WALNUTS in C++

This is a C++ implementation of the following three Hamiltonian Monte Carlo (HMC) samplers.

- NUTS
- WALNUTS
- Adaptive WALNUTS (continuous form of Nutpie-style adaptation)

Thanks to Bob Carpenter, Andreas Eberle, Tore Selland Kleppe, Sifan Liu, Milo Marsden, Stefan Oberdörster, and all of my colleagues in the MCMC community!