WALNUTS = Within-Orbit Adaptive Leapfrog No-U-Turn Sampler

Nawaf Bou-Rabee (Rutgers & Flatiron)

joint work with Bob Carpenter (Flatiron), Tore Kleppe (Norway), & Sifan Liu (Flatiron).

arxiv:2506.18746 https://github.com/bob-carpenter/walnuts

Markov Chain Monte Carlo

- **Goal:** Sample from a target probability measure μ on $\mathbb{R}^d$, with density also denoted by μ.

Markov Chain Monte Carlo

- **Goal:** Sample from a target probability measure μ on $\mathbb{R}^d$, with density also denoted by μ.
- **Approach:** Construct a Markov chain with transition kernel π such that $\mu = \mu\pi$.

Markov Chain Monte Carlo

- **Goal:** Sample from a target probability measure μ on $\mathbb{R}^d$, with density also denoted by μ.
- **Approach:** Construct a Markov chain with transition kernel π such that $\mu = \mu\pi$.
- **Sufficient condition:** *Reversibility* (detailed balance):

$$\mu(dx)\,\pi(x, dx') = \mu(dx')\,\pi(x', dx)$$

guarantees that μ is invariant.

Markov Chain Monte Carlo

- **Goal:** Sample from a target probability measure μ on $\mathbb{R}^d$, with density also denoted by μ.
- **Approach:** Construct a Markov chain with transition kernel π such that $\mu = \mu\pi$.
- **Sufficient condition:** *Reversibility* (detailed balance):

$$\mu(dx)\,\pi(x, dx') = \mu(dx')\,\pi(x', dx)$$

  guarantees that μ is invariant.
- **Two broad classes of MCMC methods:**
  - **Gradient-free:** transitions evaluate only μ (e.g., RWM, Goodman-Weare Sampler, NURS).

Markov Chain Monte Carlo

- **Goal:** Sample from a target probability measure μ on $\mathbb{R}^d$, with density also denoted by μ.
- **Approach:** Construct a Markov chain with transition kernel π such that $\mu = \mu\pi$.
- **Sufficient condition:** *Reversibility* (detailed balance):

$$\mu(dx)\,\pi(x, dx') = \mu(dx')\,\pi(x', dx)$$

  guarantees that μ is invariant.

- **Two broad classes of MCMC methods:**
  - **Gradient-free:** transitions evaluate only μ (e.g., RWM, Goodman-Weare Sampler, NURS).
  - **Gradient-based:** also evaluate $\nabla \log \mu$ (e.g., MALA, HMC, NUTS).

Hamiltonian Monte Carlo (HMC)[†]

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.

[†] (Duane et al (1987); Neal (2011))

Hamiltonian Monte Carlo (HMC)[†]

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.
- **Approach:** Introduce an auxiliary momentum $\rho \in \mathbb{R}^d$ define the Hamiltonian,

$$H(\theta, \rho) = U(\theta) + \frac{1}{2}\rho^\top M^{-1}\rho.$$

Then simulate corresponding Hamiltonian dynamics.

[†] (Duane et al (1987); Neal (2011))

Hamiltonian Monte Carlo (HMC)[†]

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.
- **Approach:** Introduce an auxiliary momentum $\rho \in \mathbb{R}^d$ define the Hamiltonian,

$$H(\theta, \rho) = U(\theta) + \frac{1}{2}\rho^\top M^{-1}\rho.$$

  Then simulate corresponding Hamiltonian dynamics.
- **HMC transition:**
  1. Refresh momentum: $\rho \sim \mathcal{N}(0, M)$.

[†] (Duane et al (1987); Neal (2011))

Hamiltonian Monte Carlo (HMC)[†]

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.
- **Approach:** Introduce an auxiliary momentum $\rho \in \mathbb{R}^d$ define the Hamiltonian,

$$H(\theta, \rho) = U(\theta) + \frac{1}{2}\rho^\top M^{-1}\rho.$$

  Then simulate corresponding Hamiltonian dynamics.
- **HMC transition:**
  1. Refresh momentum: $\rho \sim \mathcal{N}(0, M)$.
  2. Simulate Hamiltonian dynamics for time $t = ih$ using the leapfrog method.

[†](Duane et al (1987); Neal (2011))

Hamiltonian Monte Carlo (HMC)†

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.
- **Approach:** Introduce an auxiliary momentum $\rho \in \mathbb{R}^d$ define the Hamiltonian,

$$H(\theta, \rho) = U(\theta) + \frac{1}{2}\rho^\top M^{-1}\rho.$$

  Then simulate corresponding Hamiltonian dynamics.
- **HMC transition:**
  1. Refresh momentum: $\rho \sim \mathcal{N}(0, M)$.
  2. Simulate Hamiltonian dynamics for time $t = ih$ using the leapfrog method.
  3. Accept/reject the proposal using a Metropolis step.

† (Duane et al (1987); Neal (2011))

Hamiltonian Monte Carlo (HMC)[†]

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.
- **Approach:** Introduce an auxiliary momentum $\rho \in \mathbb{R}^d$ define the Hamiltonian,

$$H(\theta, \rho) = U(\theta) + \tfrac{1}{2}\rho^\top M^{-1}\rho.$$

  Then simulate corresponding Hamiltonian dynamics.

- **HMC transition:**
  1. Refresh momentum: $\rho \sim \mathcal{N}(0, M)$.
  2. Simulate Hamiltonian dynamics for time $t = ih$ using the leapfrog method.
  3. Accept/reject the proposal using a Metropolis step.
- **Invariant distribution:** the extended target $\widehat{\mu}(\theta, \rho) \propto e^{-H(\theta,\rho)}$.

[†] (Duane et al (1987); Neal (2011))

Hamiltonian Monte Carlo (HMC)[†]

- **Core idea:** Transforms the sampling problem into simulating Hamiltonian flows.
- **Approach:** Introduce an auxiliary momentum $\rho \in \mathbb{R}^d$ define the Hamiltonian,

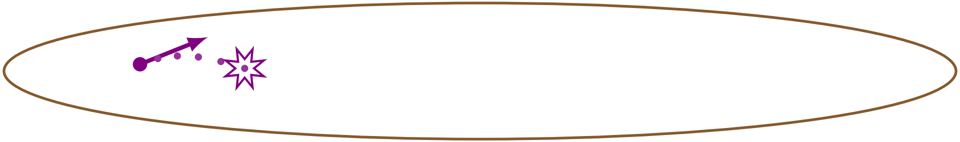
$$H(\theta, \rho) = U(\theta) + \tfrac{1}{2}\rho^\top M^{-1}\rho.$$

  Then simulate corresponding Hamiltonian dynamics.
- **HMC transition:**
  1. Refresh momentum: $\rho \sim \mathcal{N}(0, M)$.
  2. Simulate Hamiltonian dynamics for time $t = ih$ using the leapfrog method.
  3. Accept/reject the proposal using a Metropolis step.
- **Invariant distribution:** the extended target $\widehat{\mu}(\theta, \rho) \propto e^{-H(\theta,\rho)}$.
- **Tuning parameters:**

$$i \in \mathbb{Z} \quad \text{(integration time)}, \qquad h > 0 \quad \text{(step size)}, \qquad M \in \mathbb{R}^{d\times d} \quad \text{(mass matrix)}.$$

[†] (Duane et al (1987); Neal (2011))

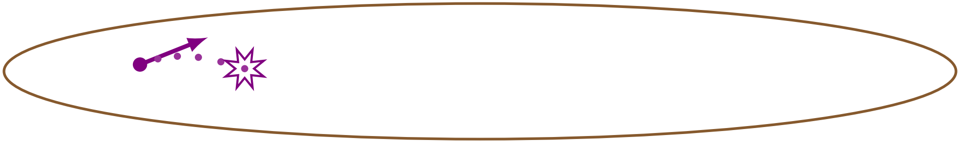
Why Integration Time Tuning is Tricky in HMC†



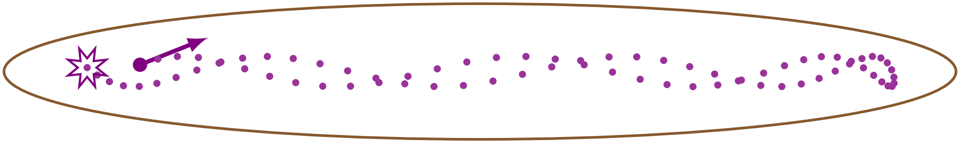
Too short: insufficient integration leads to random-walk behavior

† (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))

The No-U-Turn Sampler (NUTS)[†]

- NUTS builds on HMC by adaptively choosing integration time.

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

The No-U-Turn Sampler (NUTS)[†]

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).

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

The No-U-Turn Sampler (NUTS)[†]

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

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

The No-U-Turn Sampler (NUTS)[†]

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

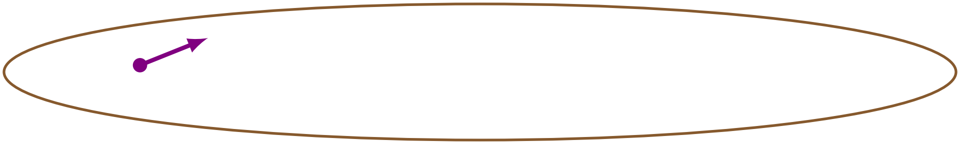


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

The No-U-Turn Sampler (NUTS)†

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

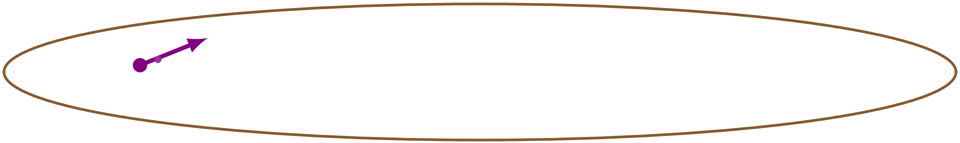


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

The No-U-Turn Sampler (NUTS)†

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

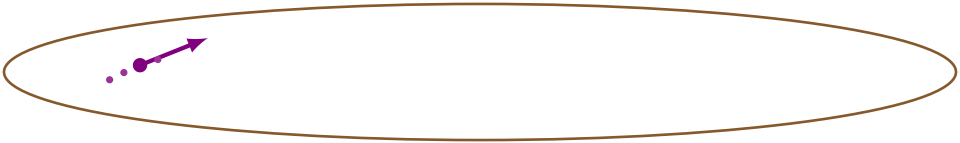


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

The No-U-Turn Sampler (NUTS)†

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

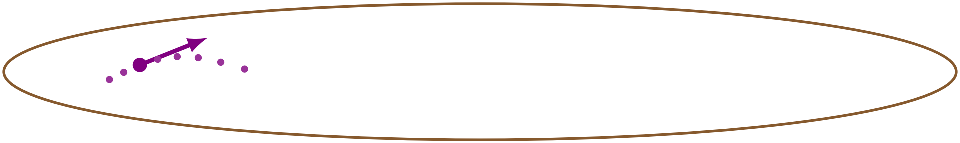


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

The No-U-Turn Sampler (NUTS)†

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

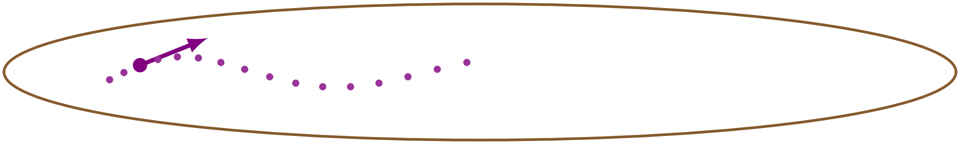


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

The No-U-Turn Sampler (NUTS)†

- NUTS builds on HMC by adaptively choosing integration time.
- **Widely used** in probabilistic programming languages:
  - `Stan`: CmdStan (2.2M), RStan (6.0M), PyStan (110M+ downloads).
  - Also used in: PyMC, NumPyro, Turing, NIMBLE.

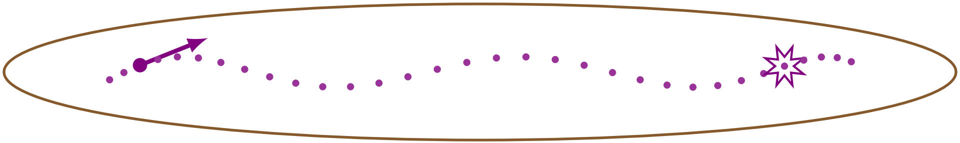


Illustration: NUTS builds a trajectory by randomly expanding forward and backward until a U-turn.

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

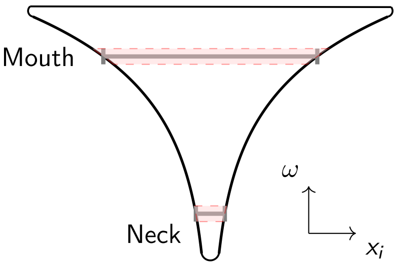
Neal's Funnel: $\mu(\omega, x) = \mathcal{N}(\omega \mid 0, 9) \prod_{i=1}^{d} \mathcal{N}(x_i \mid 0, e^{\omega})$

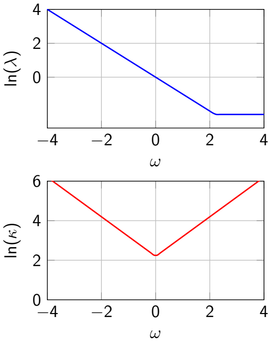
- **However:** some important targets exhibit extreme variations in scale, e.g., Neal's funnel.†

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

Neal's Funnel: $\mu(\omega, x) = \mathcal{N}(\omega \mid 0, 9) \prod_{i=1}^{d} \mathcal{N}(x_i \mid 0, e^{\omega})$

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



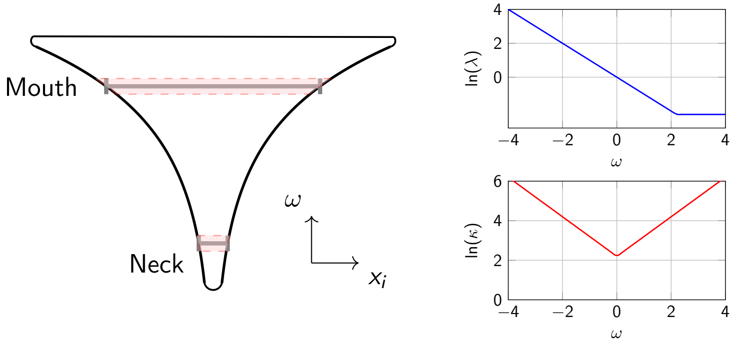


- **Left:** Funnel width scales as $e^{\omega/2}$ — shape preserved under horizontal rescaling.

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

Neal's Funnel: $\mu(\omega, x) = \mathcal{N}(\omega \mid 0, 9) \prod_{i=1}^{d} \mathcal{N}(x_i \mid 0, e^{\omega})$

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

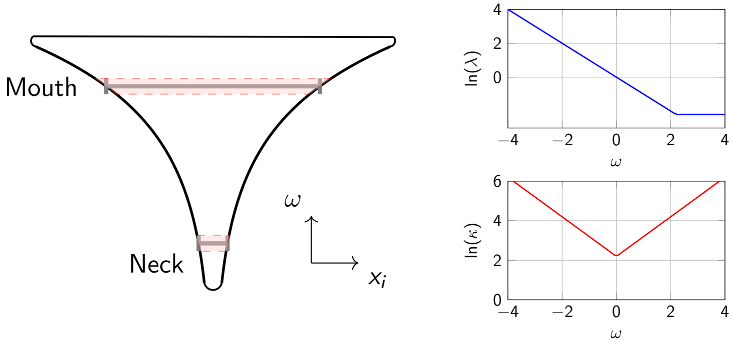


- **Left:** Funnel width scales as $e^{\omega/2}$ — shape preserved under horizontal rescaling.
- **Top right:** Spectral radius $\lambda(\omega) = \max(1/9, e^{-\omega})$ grows in the neck.

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

Neal's Funnel: $\mu(\omega, x) = \mathcal{N}(\omega \mid 0, 9) \prod_{i=1}^{d} \mathcal{N}(x_i \mid 0, e^{\omega})$

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



- **Left:** Funnel width scales as $e^{\omega/2}$ — shape preserved under horizontal rescaling.
- **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)

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.

- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.
  - Lightweight modification: simply reweights states generated during orbit expansion.

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.
  - Lightweight modification: simply reweights states generated during orbit expansion.
  - Preserves core features of NUTS: path length adaptivity & biased progressive sampling.

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.
  - Lightweight modification: simply reweights states generated during orbit expansion.
  - Preserves core features of NUTS: path length adaptivity & biased progressive sampling.
- **Benefits:**
  - Reversible and unbiased.

Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.
  - Lightweight modification: simply reweights states generated during orbit expansion.
  - Preserves core features of NUTS: path length adaptivity & biased progressive sampling.
- **Benefits:**
  - Reversible and unbiased.
  - Plugs directly into the existing NUTS implementation in Stan.

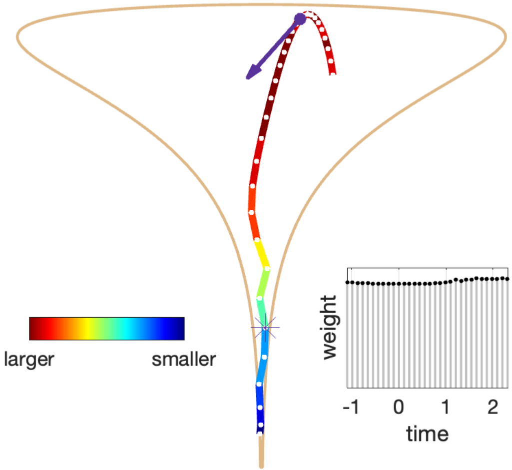
Our Solution: WALNUTS

- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.
  - Lightweight modification: simply reweights states generated during orbit expansion.
  - Preserves core features of NUTS: path length adaptivity & biased progressive sampling.
- **Benefits:**
  - Reversible and unbiased.
  - Plugs directly into the existing NUTS implementation in Stan.
  - Improves robustness through local step-size adaptivity.

Our Solution: WALNUTS

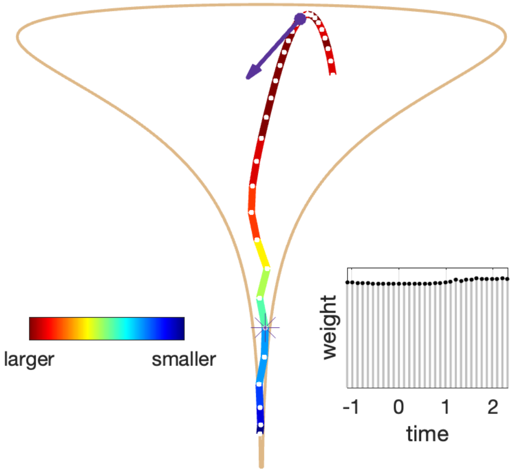
- **WALNUTS** = **W**ithin-orbit **A**daptive **L**eapfrog **N**o-**U**-**T**urn **S**ampler.
- **Key ideas:**
  - Introduces variable step sizes *within* each leapfrog orbit.
  - Adapts the step size based on local energy error.
  - Lightweight modification: simply reweights states generated during orbit expansion.
  - Preserves core features of NUTS: path length adaptivity & biased progressive sampling.
- **Benefits:**
  - Reversible and unbiased.
  - Plugs directly into the existing NUTS implementation in Stan.
  - Improves robustness through local step-size adaptivity.
  - More forgiving with respect to tuning (e.g., macro step size).

Visualization of WALNUTS Transition Step



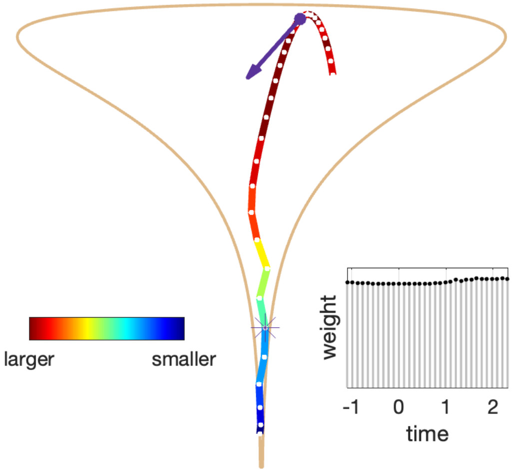
- **Macro steps:** white dots mark positions at coarse time steps.

Visualization of WALNUTS Transition Step



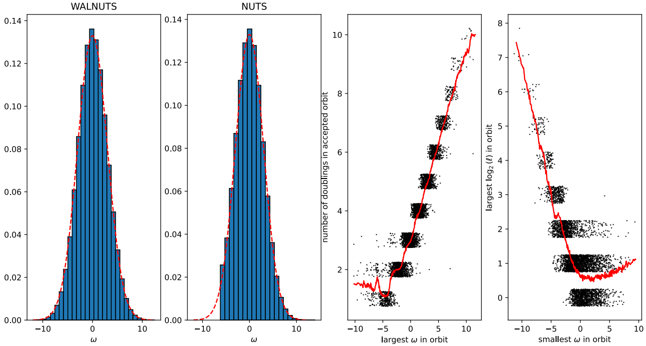
- **Macro steps:** white dots mark positions at coarse time steps.
- **Micro steps:** colors between macro points show variable-resolution integration adapted to local energy error.

Visualization of WALNUTS Transition Step



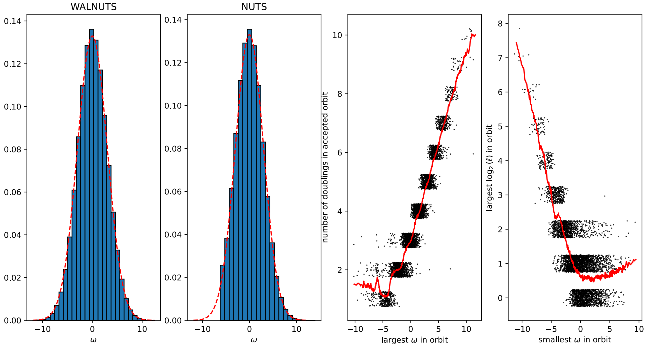
- **Macro steps:** white dots mark positions at coarse time steps.
- **Micro steps:** colors between macro points show variable-resolution integration adapted to local energy error.
- **Final state:** star indicates selection via biased progressive sampling (inset).

WALNUTS vs NUTS in Neal's Funnel



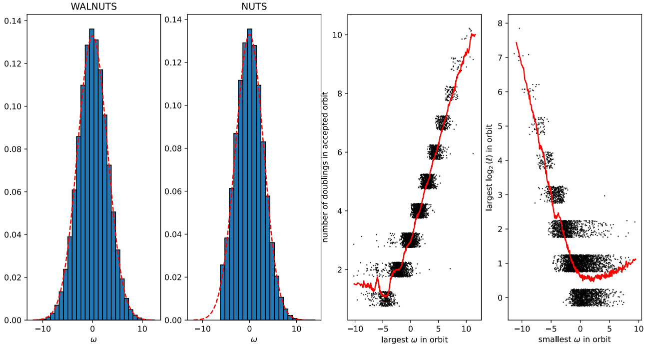
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.

WALNUTS vs NUTS in Neal's Funnel



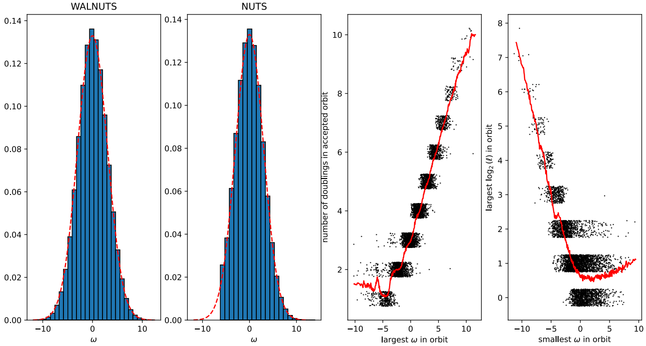
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.

WALNUTS vs NUTS in Neal's Funnel



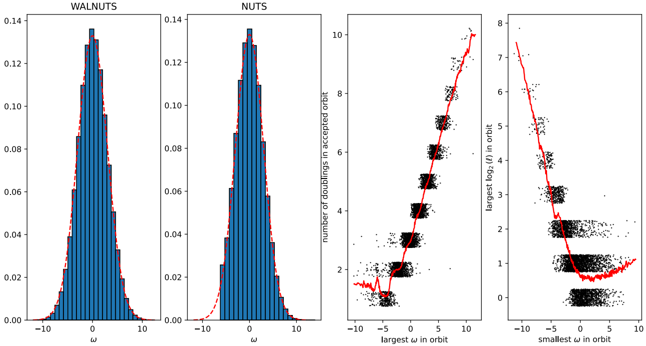
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.

WALNUTS vs NUTS in Neal's Funnel



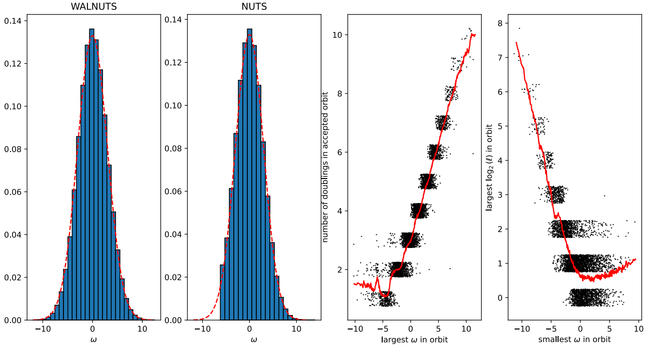
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.
- **Right two panels:** Behavior of adaptive parameters vs. location in the funnel.

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



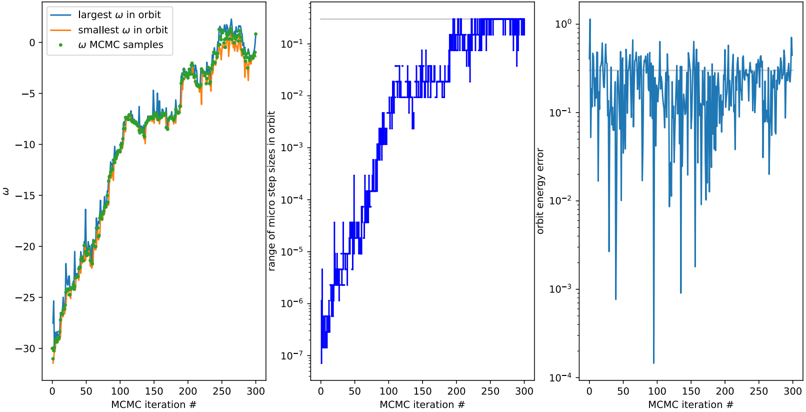
- **Left two panels:** ω-marginal histograms for WALNUTS and NUTS.
  - WALNUTS recovers the correct $\mathcal{N}(0, 9)$ distribution.
  - NUTS exhibits bias despite using 104% of WALNUTS's total gradient calls.
- **Right two panels:** Behavior of adaptive parameters vs. location in the funnel.
  - Orbit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

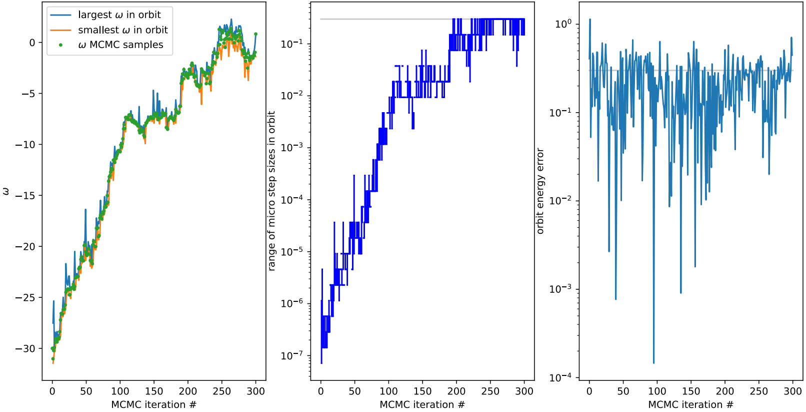
- **Setup:** Initialized deep in the neck: $\omega = -30$, $x_i = 0$ for $i = 1, \dots, 10$.



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

Cold-Start Diagnostics: WALNUTS in Neal's Funnel

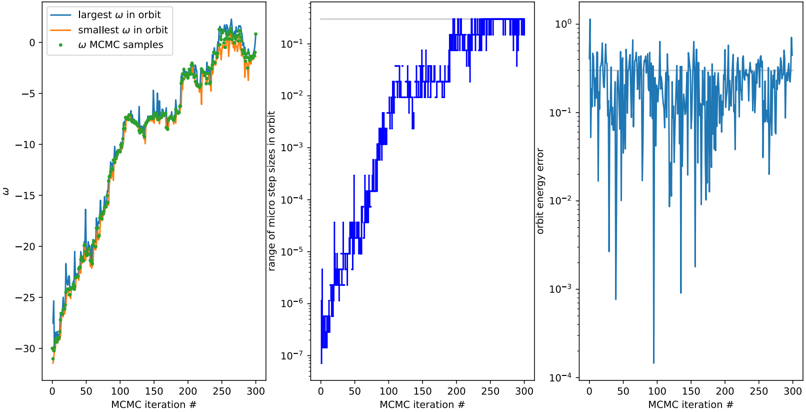
- **Setup:** Initialized deep in the neck: $\omega = -30$, $x_i = 0$ for $i = 1, \ldots, 10$.



- **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; dashed line shows $\delta = 0.3$.

WALNUTS: Transition Step

† Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].

[†] Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum),

[†] Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit),

[†]Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time),

[†] Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).

[†]Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).
- Define joint distribution over all variables: p_{joint}.

[†]Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).
- Define joint distribution over all variables: p_{joint}.
- **WALNUTS transition step:**

[†] Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].

- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).

- Define joint distribution over all variables: p_{joint}.

- **WALNUTS transition step:**
  1. Refresh auxiliary variables.

[†]Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework†.
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).
- Define joint distribution over all variables: p_{joint}.
- **WALNUTS transition step:**
  1. Refresh auxiliary variables.
  2. Apply an involution Ψ.

†Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework[†].
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).
- Define joint distribution over all variables: p_{joint}.
- **WALNUTS transition step:**
  1. Refresh auxiliary variables.
  2. Apply an involution Ψ.
  3. Return θ_i.

[†] Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

- Instance of the auxiliary-variable-and-involution framework†.
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).
- Define joint distribution over all variables: p_{joint}.
- **WALNUTS transition step:**
  1. Refresh auxiliary variables.
  2. Apply an involution Ψ.
  3. Return θ_i.
- Ψ preserves p_{joint}

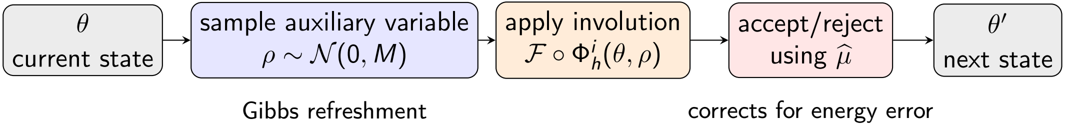
† Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

WALNUTS: Transition Step

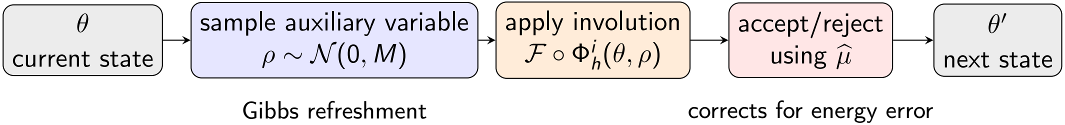
- Instance of the auxiliary-variable-and-involution framework†.
- **Auxiliary variables:** $\rho \in \mathbb{R}^d$ (momentum), $\mathcal{O} \subset \mathbb{R}^{2d}$ (orbit), $i \in \mathbb{Z}$ (integration time), $\boldsymbol{\ell} \in \mathbb{N}^{|\mathcal{O}|-1}$ (micro step counts).
- Define joint distribution over all variables: p_{joint}.
- **WALNUTS transition step:**
  1. Refresh auxiliary variables.
  2. Apply an involution Ψ.
  3. Return θ_i.
- Ψ preserves $p_{\text{joint}} \Rightarrow$ WALNUTS is reversible.

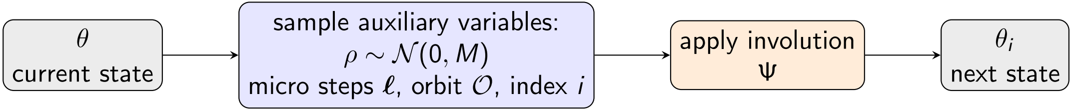
† Andrieu, Lee, & Livingstone (2020); Glatt-Holtz, Krometis, & Mondaini (2023); B.-R., Carpenter, & Marsden (2024)

Contrast with HMC



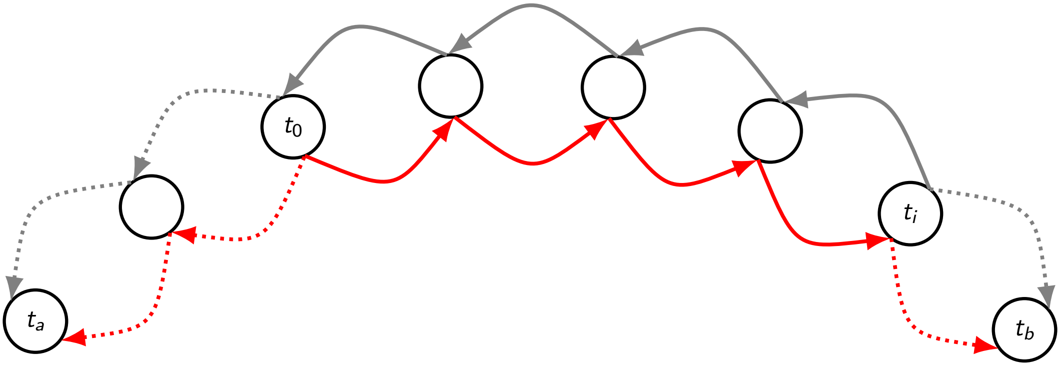
Contrast with HMC





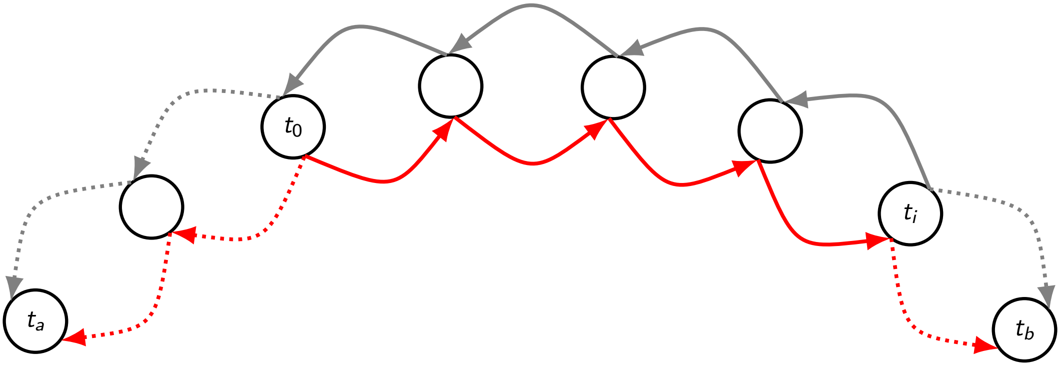
no Metropolis correction is needed since Ψ preserves the joint density

WALNUTS: Orbit Construction



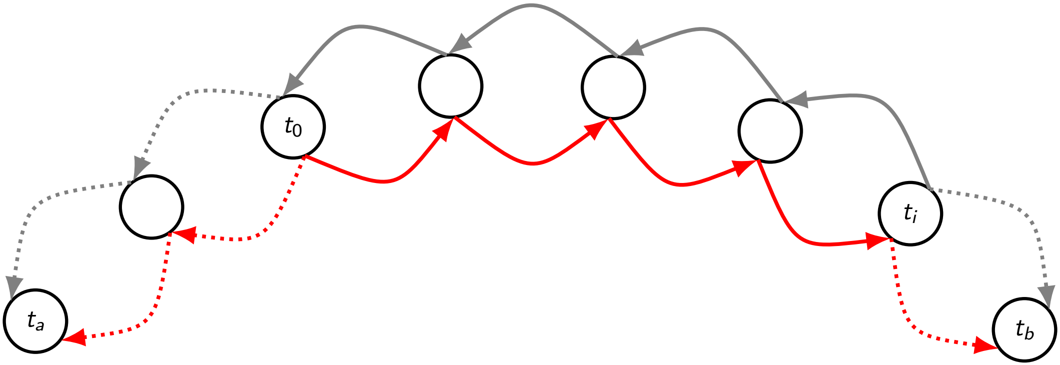
Red: forward orbit from (θ_0, ρ_0).

WALNUTS: Orbit Construction



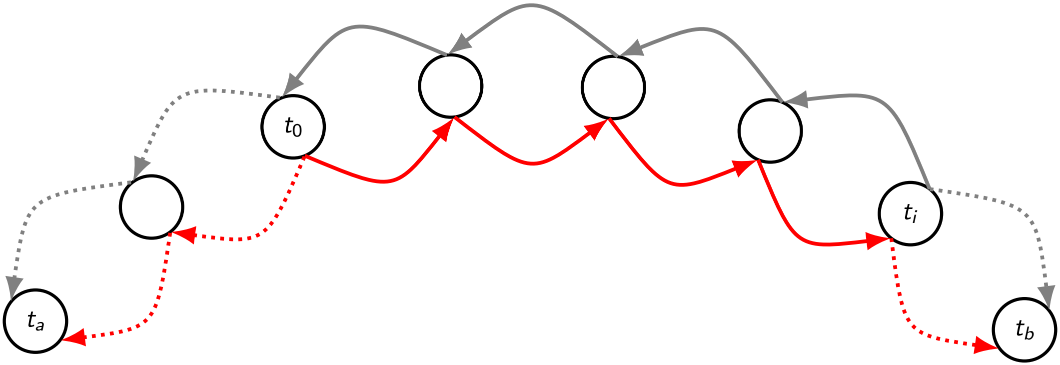
Red: forward orbit from (θ_0, ρ_0). Gray: same from (θ_i, ρ_i).

WALNUTS: Orbit Construction

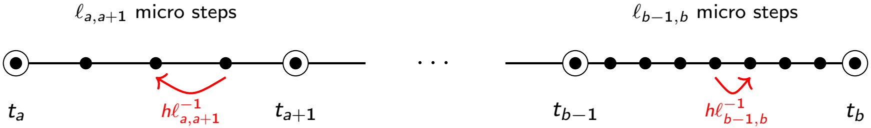


Red: forward orbit from (θ_0, ρ_0). Gray: same from (θ_i, ρ_i). Dotted segments: same step-size distribution.

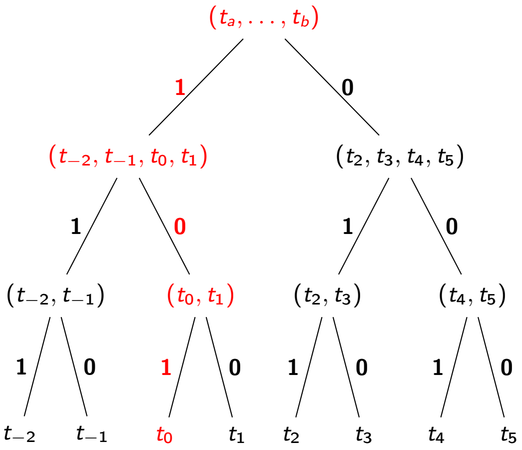
WALNUTS: Orbit Construction



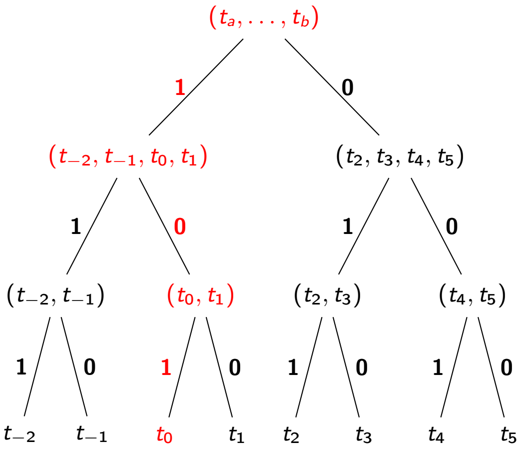
Red: forward orbit from (θ_0, ρ_0). Gray: same from (θ_i, ρ_i). Dotted segments: same step-size distribution.

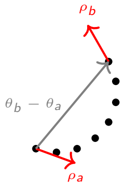


WALNUTS: Doubling Tree and U-turn Condition



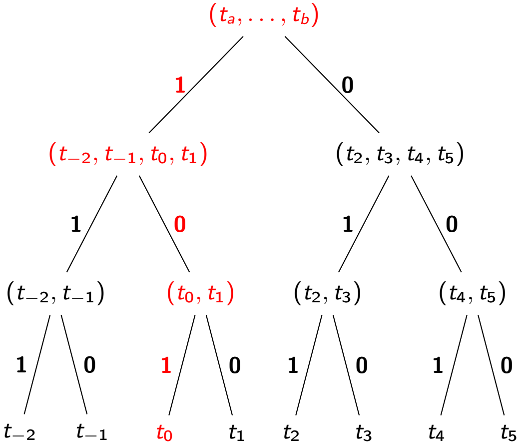
WALNUTS: Doubling Tree and U-turn Condition

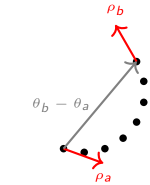




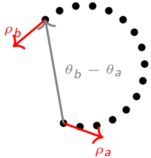
$$\min(\rho_a \cdot (\theta_b - \theta_a),\ \rho_b \cdot (\theta_b - \theta_a)) > 0$$

WALNUTS: Doubling Tree and U-turn Condition





$$\min(\rho_a \cdot (\theta_b - \theta_a),\ \rho_b \cdot (\theta_b - \theta_a)) > 0$$



$$\min(\rho_a \cdot (\theta_b - \theta_a),\ \rho_b \cdot (\theta_b - \theta_a)) < 0$$

WALNUTS: Randomized Integration Time Selection

- WALNUTS selects the next state via biased progressive sampling (BPS).

WALNUTS: Randomized Integration Time Selection

- WALNUTS selects the next state via biased progressive sampling (BPS).
- At each doubling step, sample a candidate index:

$$i_k^{\text{ext}} \sim \text{categorical}(a_k^{\text{ext}}{:}b_k^{\text{ext}}, \mathcal{W}_k^{\text{ext}}).$$

WALNUTS: Randomized Integration Time Selection

- WALNUTS selects the next state via biased progressive sampling (BPS).
- At each doubling step, sample a candidate index:

$$i_k^{\text{ext}} \sim \text{categorical}(a_k^{\text{ext}}:b_k^{\text{ext}}, \mathcal{W}_k^{\text{ext}}).$$

- Accept with probability:

$$\min\left(1, \frac{\sum \mathcal{W}_k^{\text{ext}}}{\sum \mathcal{W}_k}\right).$$

WALNUTS: Randomized Integration Time Selection

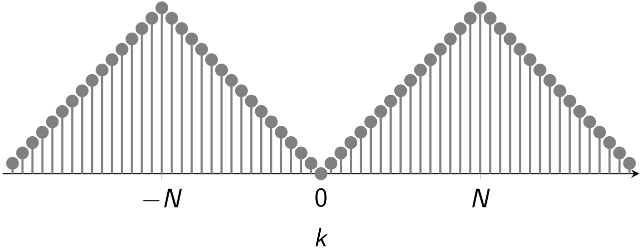
- WALNUTS selects the next state via biased progressive sampling (BPS).
- At each doubling step, sample a candidate index:

$$i_k^{\mathrm{ext}} \sim \mathrm{categorical}(a_k^{\mathrm{ext}}{:}b_k^{\mathrm{ext}},\, \mathcal{W}_k^{\mathrm{ext}}).$$

- Accept with probability:

$$\min\left(1, \frac{\sum \mathcal{W}_k^{\mathrm{ext}}}{\sum \mathcal{W}_k}\right).$$

- With uniform weights and orbit lengths, BPS produces a symmetric triangular distribution.



WALNUTS: Proof of Reversibility

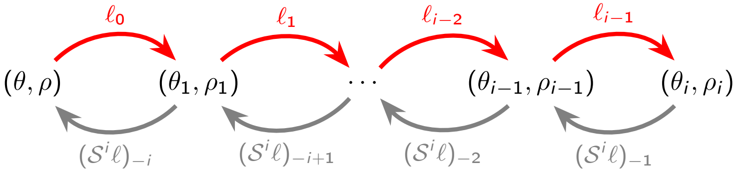
- Defines a carefully constructed involution Ψ on the augmented space: $z = (\theta, \rho, m, b, \boldsymbol{\ell}, i)$.

WALNUTS: Proof of Reversibility

- Defines a carefully constructed involution Ψ on the augmented space: $z = (\theta, \rho, m, b, \boldsymbol{\ell}, i)$.
- **In words:** Ψ recenters the orbit to index i by updating the initial point to (θ_i, ρ_i) and shifting the step size sequence accordingly $\Psi : (\theta, \rho, m, b, \boldsymbol{\ell}, i) \mapsto (\theta_i, \rho_i, m, b - i, \mathcal{S}^i \boldsymbol{\ell}, -i)$.

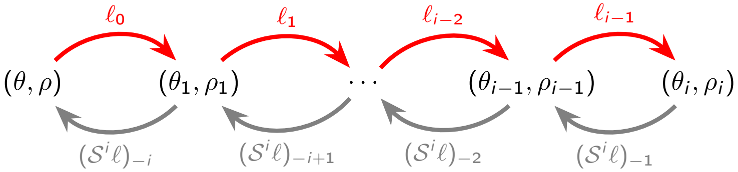
WALNUTS: Proof of Reversibility

- Defines a carefully constructed involution Ψ on the augmented space: $z = (\theta, \rho, m, b, \boldsymbol{\ell}, i)$.
- **In words:** Ψ recenters the orbit to index i by updating the initial point to (θ_i, ρ_i) and shifting the step size sequence accordingly $\Psi : (\theta, \rho, m, b, \boldsymbol{\ell}, i) \mapsto (\theta_i, \rho_i, m, b - i, \mathcal{S}^i \boldsymbol{\ell}, -i)$.



WALNUTS: Proof of Reversibility

- Defines a carefully constructed involution Ψ on the augmented space: $z = (\theta, \rho, m, b, \boldsymbol{\ell}, i)$.
- **In words:** Ψ recenters the orbit to index i by updating the initial point to (θ_i, ρ_i) and shifting the step size sequence accordingly $\Psi : (\theta, \rho, m, b, \boldsymbol{\ell}, i) \mapsto (\theta_i, \rho_i, m, b - i, \mathcal{S}^i \boldsymbol{\ell}, -i)$.

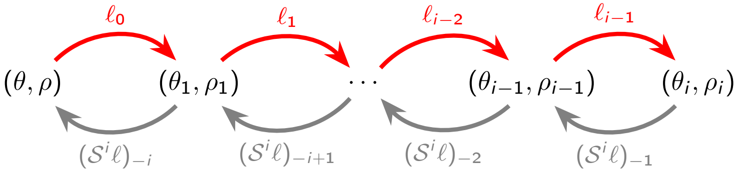


- The **extended target distribution** is exactly **invariant** under Ψ:

$$p_{\text{joint}}(z) \propto e^{-H(\theta,\rho)} \cdot p_{\text{orbit}}(m, b, \boldsymbol{\ell} \mid \theta, \rho) \cdot p_{\text{index}}(i \mid \cdot \theta, \rho, m, b, \boldsymbol{\ell})$$

WALNUTS: Proof of Reversibility

- Defines a carefully constructed involution Ψ on the augmented space: $z = (\theta, \rho, m, b, \boldsymbol{\ell}, i)$.
- **In words:** Ψ recenters the orbit to index i by updating the initial point to (θ_i, ρ_i) and shifting the step size sequence accordingly $\Psi : (\theta, \rho, m, b, \boldsymbol{\ell}, i) \mapsto (\theta_i, \rho_i, m, b - i, \mathcal{S}^i \boldsymbol{\ell}, -i)$.



- The **extended target distribution** is exactly **invariant** under Ψ:

$$p_{\text{joint}}(z) \propto e^{-H(\theta,\rho)} \cdot p_{\text{orbit}}(m, b, \boldsymbol{\ell} \mid \theta, \rho) \cdot p_{\text{index}}(i \mid \cdot \theta, \rho, m, b, \boldsymbol{\ell})$$

- **Bottom line:** WALNUTS transition kernel is reversible with respect to the target μ.

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.
- **Plug-and-play.** Easily integrates into frameworks like Stan, PyMC, and NumPyro.

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.
- **Plug-and-play.** Easily integrates into frameworks like Stan, PyMC, and NumPyro.
- **Beyond NUTS.** The core adaptive scheme extends to other HMC-type methods.

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.
- **Plug-and-play.** Easily integrates into frameworks like Stan, PyMC, and NumPyro.
- **Beyond NUTS.** The core adaptive scheme extends to other HMC-type methods.

Future Directions

- **Mass matrix.** Build on Riemannian HMC (Girolami & Calderhead (2011); Hird & Livingstone (2023); Whalley, Paulin & Leimkuhler (2024); Tran & Kleppe (2024)).

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.
- **Plug-and-play.** Easily integrates into frameworks like Stan, PyMC, and NumPyro.
- **Beyond NUTS.** The core adaptive scheme extends to other HMC-type methods.

Future Directions

- **Mass matrix.** Build on Riemannian HMC (Girolami & Calderhead (2011); Hird & Livingstone (2023); Whalley, Paulin & Leimkuhler (2024); Tran & Kleppe (2024)).
- **Ensemble methods.** Leverage ensembles (Goodman & Weare (2010); Chen (2025)).

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.
- **Plug-and-play.** Easily integrates into frameworks like Stan, PyMC, and NumPyro.
- **Beyond NUTS.** The core adaptive scheme extends to other HMC-type methods.

Future Directions

- **Mass matrix.** Build on Riemannian HMC (Girolami & Calderhead (2011); Hird & Livingstone (2023); Whalley, Paulin & Leimkuhler (2024); Tran & Kleppe (2024)).
- **Ensemble methods.** Leverage ensembles (Goodman & Weare (2010); Chen (2025)).
- **Adam-like Step-sizes.** See Ben's talk (Leimkuhler, Lohman & Whalley (2025)).

Conclusion & Outlook

Conclusions

- **Summary.** WALNUTS improves performance with minimal overhead.
- **Plug-and-play.** Easily integrates into frameworks like Stan, PyMC, and NumPyro.
- **Beyond NUTS.** The core adaptive scheme extends to other HMC-type methods.

Future Directions

- **Mass matrix.** Build on Riemannian HMC (Girolami & Calderhead (2011); Hird & Livingstone (2023); Whalley, Paulin & Leimkuhler (2024); Tran & Kleppe (2024)).
- **Ensemble methods.** Leverage ensembles (Goodman & Weare (2010); Chen (2025)).
- **Adam-like Step-sizes.** See Ben's talk (Leimkuhler, Lohman & Whalley (2025)).

Outlook. Points toward a new class of locally adaptive HMC methods for anisotropic targets.

WALNUTS: Paper and Code

- **Paper:** `arXiv:2506.18746`

- **Code Repository:** `github.com/bob-carpenter/walnuts`

Questions, feedback, or contributions are welcome!