Score-based Metropolis-Hastings Algorithms abd Their Applications to Fractional Langevin Algorithms

Vahid Tarokh
Duke University

Joint work with: Ahmed Aloui, Juncheng Dong , Ali Hasan, Junyi Liao, Zihao Wu and Jose Blanchet

Outline

- Introduction and Motivation
- Background
- Score Balance Matching (Gaussian Proposals)
- Empirical Results — Gaussian Setting
- Extension to Fractional Langevin
- MAFLA Algorithm
- Empirical Results — Fractional Setting
- Summary & Future Directions

Introduction

- Key to generative modelling is sampling from a distribution that estimates the true distribution of a finite (in some cases very large) set of data points.
  - In most cases, we intentionally designed the approximating distribution such that sampling from it is easy, e.g.
    - GANs,
    - Normalizing Flows,
    - RBMs,
    - VAEs,
    - Energy Based Models
    - Score Bases Models, etc.
- In some cases, we may also make some further simplifying approximations to make the training made simpler.

Introduction

- In general, sampling from a distribution with target density $\pi(x)$ is an important in many fields with many applications, such as:
  - Estimate expectations, risk metrics (VaR, Expected Shortfall)
  - Bayesian inference and posterior uncertainty
  - Generative modeling (images, financial time series)
  - Combinatorial optimization using relaxed Gibbs distributions
- In modern applications, targets are often multimodal, high-dimensional, and heavy-tailed.
- Various challenges may be encountered such as:
  - Potential of slow convergence to the target distribution
  - Target density may be unknown and only samples of it may be presented
  - Potential of misrepresentation of tails of target distributions

Introduction

- A beautiful sampling idea is Markov Chain Monte Carlo **(MCMC).** Some important examples include:
  - Metropolis-Hastings Algorithm
  - Random-Walk Metropolis
  - Gibbs Sampling
  - Hamiltonian Monte Carlo (HMC)
  - Metropolis Adjusted Langevin Algorithm (MALA) a.k.a Langevin Monte Carlo (LMC), etc.
- But first, we need to quickly review some technicalities

Quick Overview of Some Technicalities

- An infinite-state discrete-time Markov chain with a countable state space s possesses a unique stationary distribution if and only if it satisfies two core conditions: irreducibility and positive recurrence.

Definitions:

- **Irreducibility**: Every state must be accessible from every other state,
- **Positive Recurrence:** The expected return time to any state i starting from i must be strictly finite ($m_i < \infty$).

Disclosure: This Slide was Generated by Prompting Gemini (by Google).

Quick Overview of Some Technicalities

- For a Markov chain with a continuous, uncountable state space like $\mathbb{R}^n$, the transition probabilities shift from matrices to transition kernels $P(x, A)$. A unique stationary probability distribution π exists under measure-theoretic conditions that generalize discrete concepts:
- **ϕ-Irreducibility :** There exists a maximal irreducibility measure ϕ on $\mathbb{R}^n$ such that for any measurable set A where $\phi(A) > 0$, the probability of eventually hitting A from any starting point $x \in \mathbb{R}^n$ is strictly greater than zero.
- **Positive Harris Recurrence:** The chain must be Harris recurrent—meaning it visits any set of positive ϕ-measure infinitely often with probability 1—and possess a finite invariant measure, making it positive.
- **Foster-Lyapunov Drift Conditions:** Because direct computation of recurrence times is nearly impossible in $\mathbb{R}^n$, stability is proved using a **Lyapunov function** $V: \mathbb{R}^n \to [1, \infty)$ that satisfies a drift inequality of the form $PV(x) \leq \lambda V(x) + b$ for some $\lambda < 1$ and $b < \infty$, coupled with a "small set" condition , where $PV(x) = \mathbb{E}[V(X_{t+1}) \mid X_t = x] = \int_{\mathbb{R}^n} V(y) P(x, dy)$

Disclosure: This Slide was Generated by Prompting Gemini (by Google).

Quick Overview of some Technicalities

- Detailed balance requires that the probability flux from any measurable set A to another set B equals the flux from B to A:

$$\int_A P\,(x, B)\pi(dx) = \int_B P\,(y, A)\pi(dy)$$

- When the transition kernel can be expressed via a transition probability density $p(x, y)$ (such that $P(x, dy) = p(x, y)dy$) and the stationary distribution has a density function $\pi(x)$, the abstract integral balance conditions simplify into standard calculus:

$$\pi(x)p(x, y) = \pi(y)p(y, x)$$

- **The Detailed Balance Theorem** (sometimes called the Reversibility Theorem) formally guarantees that any measure satisfying detailed balance is automatically a stationary measure for the Markov chain.

Notes: This Slide was Generated by Prompting Gemini (by Google)

Ingenious and Remarkably Simple Idea of Metropolis

- Make a Markov chain whose stationary distribution is the target distribution p(x) by guaranteeing detailed balance equation.
- Start with a proposal $q(x'|x)$ for moving from x to x'
- Need to make sure that it satisfies the balance equation. Let us accept the proposed move with acceptance probability $a(x', x)$. Total transition probability is $q(x'|x)\ a(x', x)$.
- Choose $a(x', x)$ such that the detailed balance holds

$$a(x', x)\ q(x'|x)\ p(x) = a(x, x')\ q(x|x')\ p(x')$$

Ingenious and Remarkably Simple Idea of Metropolis

- Let

$$a(x', x) = \min\left\{1, \frac{p(x')\, q(x|x')}{p(x)\, q(x'|x)}\right\}$$

  - Then either $a(x', x) = 1$ or $a(x, x') = 1$.
  - In either case, it is trivial to see that detailed balance holds.
- We refer to accepting the move from x to x' with probability $a(x', x)$ as Langevin Adjustment.

Engineering Issues With Metropolis idea Algorithm

- Choice of proposal function: If the proposal is chosen poorly then the accepted moves from x to x' may be minuscule and it takes forever to converge to the target distribution (huge computational complexity).
- The choice becomes critical for generation of natural high dimensional data.
- What if the proposal function $q(x'|x)$ is approximated by an expressive DNN?
  - In this case, how we should train the DNN? What should be the objective function for the training?
  - A potential approach is proposed in [1] which nearly achieves optimal scaling laws of G.S. Roberts and J.S. Rosenthal [2,3].

[1] Chris Cannella and Vahid Tarokh, SEMI-EMPIRICAL OBJECTIVE FUNCTIONS FOR MCMC PROPOSAL OPTIMIZATION, 26th International Conference on Pattern Recognition (ICPR), 2022.

[2] Gareth O. Roberts and Jeffrey S. Rosenthal, Optimal scaling of discrete approximations to Langevin diffusions, *Journal of the Royal Statistical Society Series B: Statistical Methodology*, Volume 60, Issue 1, January 1998.

[3] Gareth O. Roberts and Jeffrey S. Rosenthal, Optimal scaling for various Metropolis-Hasting Algorithms, Statistical Sciences, 2001.

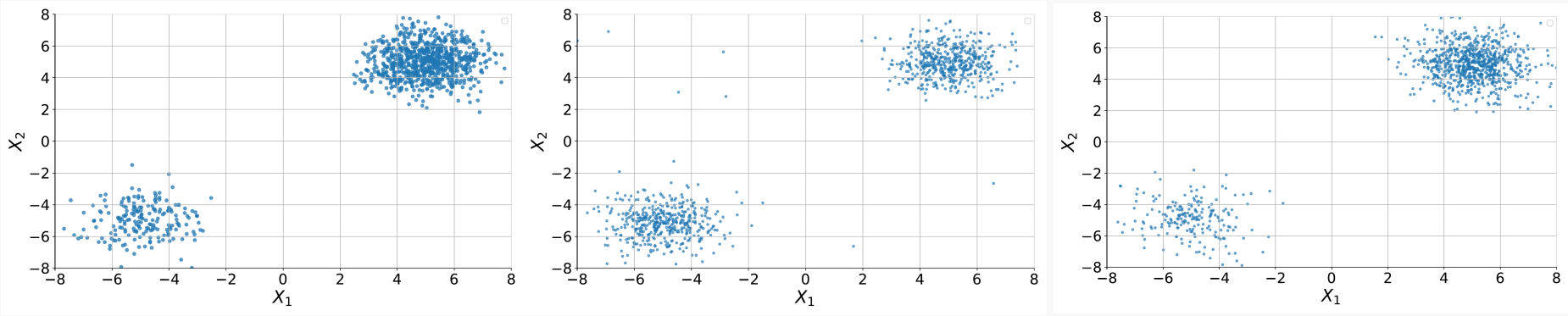
Examples of Some Classical Proposal Functions

Random Walk (RW) *Symmetric Gaussian steps*	**MALA** *Metropolis-Adjusted Langevin*	**Preconditioned Crank-Nicolson (pCN)** *Scaled mix of current state and noise*
$q(x' \mid x) = \mathcal{N}(x,\, \sigma^2 I)$	$q(x' \mid x) = \mathcal{N}\left(x + \frac{\varepsilon^2}{2}\nabla \log p(x),\, \varepsilon^2 I\right)$	$x' = \sqrt{1-\beta^2}\, x + \beta\, \xi, \quad \xi \sim \mathcal{N}(0, I)$

- Some MCMC methods can be applied without Metropolis adjustment if the acceptance function $a(x', x) \simeq 1$.
- An example is Langevin Dynamics presented with the Metropolis adjustment in the above.
  - If $\epsilon \simeq 0$, then $a(x', x) \simeq 1$ and it may work (although miserably slowly) without Langevin adjustment.
  - We referred to this as Unadjusted Langevin Algorithm (ULA).

Numerical Experiment

Sampling a Bimodal Gaussian mixture with weights (0.8, 0.2)



True (0.8, 0.2) **ULA (0.52, 0.48)** **MALA (0.79, 0.21)**

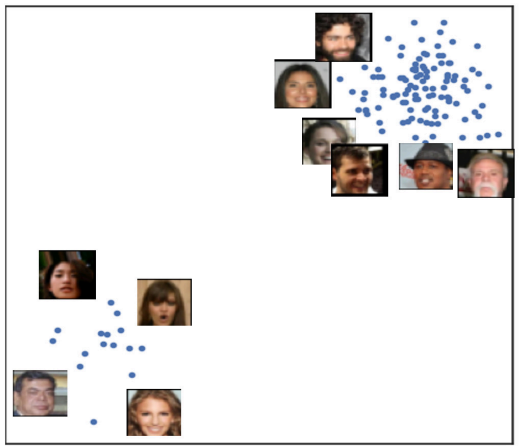
Engineering Issues With Metropolis idea Algorithm

- Not clear that the acceptance function of Metropolis is optimal in any sense?
  - What if we replace the acceptance function with an expressive deep neural network? The last layer of the DNN can be a sigmoid with range in [0,1].
    - **Question: How to train this neural network?**
- Target distribution may not be explicitly available and must be learned from limited data:
  - Mismatch may happen if dataset is not large enough to learn the target distribution well, or if the density estimation method does not match the nature of data.
  - There may not exist enough tail/extreme data so the learned distribution may address the bulk and does not do well in generation of the tail of data.
  - Is it even clear that estimating the density is the right thing to do?

Emergence of Score Matching and Diffusion Models

- In recent years, there has been an emergence of interest in score matching and diffusion models for high dimensional data.
- These models have shown the SOTA performance for image generation (and also other data types).
- In this approach rather than modeling of the pdf $p(x)$ of data directly, $\nabla_x \ln p(x)$ is modeled.
- New images then can be generated using Unadjusted Langevin Algorithm.
- This introduces bias so a lot of heuristics and fine-tuning variants of ULA are used to generate the SOTA performance.
- This approach enables remarkable generative quality (DALL·E 2, Stable Diffusion) without explicit densities.

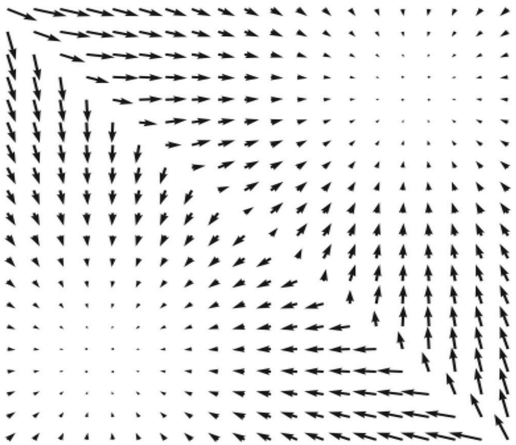
Emergence of Score Matching and Diffusion Models



Data samples

$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$

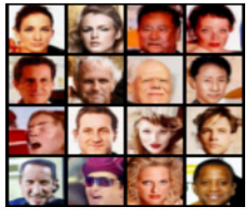
score matching



Scores

$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$

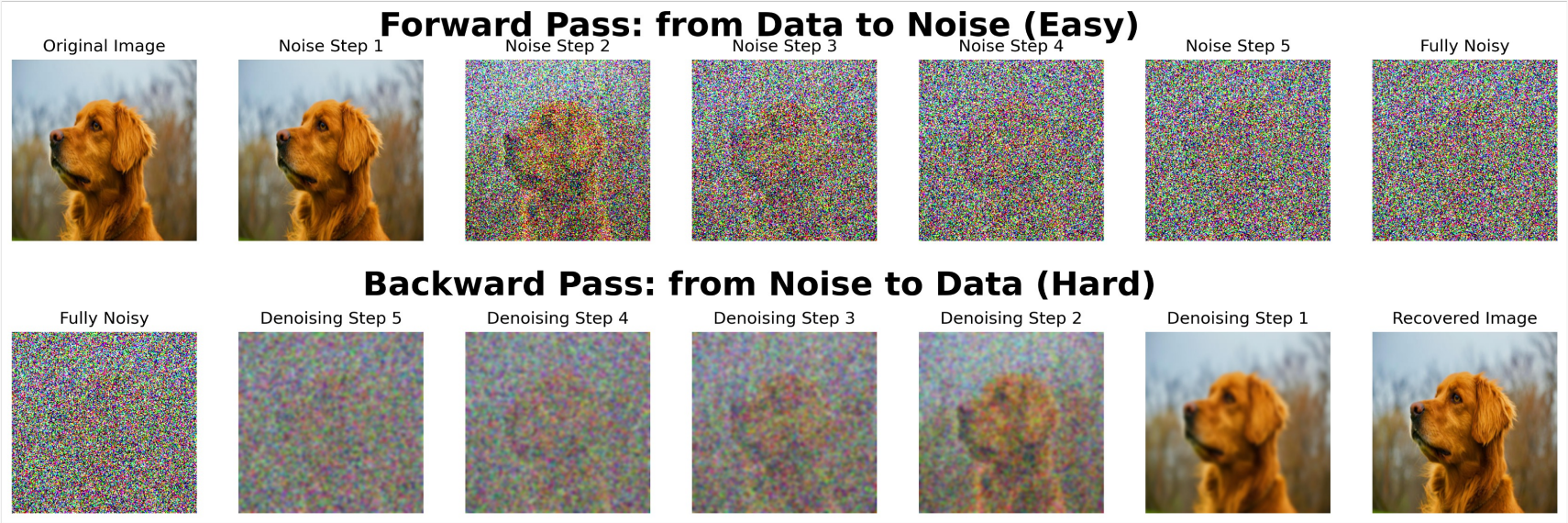
Langevin dynamics



New samples

Diffusion Models via Score Matching

- Forward process: gradually add noise to data $p_0(x) \rightarrow p_t(x) \rightarrow p_T(x) \sim N(0, I)$
- Train a time-dependent (conditional) score network (DNN): $s_\theta(x, t) \approx \nabla_x \log p_t(x)$
- Reverse process: start from noise, generate samples using the learned score



Unadjusted Langevin Dynamics

- Discretize the overdamped Langevin SDE whose stationary distribution is p(x):

$$\mathrm{d}X_t = \nabla_x \log p(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}W_t$$

$$x_{t+1} = x_t + \eta\,\nabla_x \log p(x_t) + \sqrt{2\eta}\,\xi_t, \quad \xi_t \sim \mathcal{N}(0, I_d)$$

- ULA uses only the score $\nabla_x \log p(x_t)$ — no explicit density needed
- Discretization error → stationary distribution ≠ p(x) (bias)
- Slow mixing: gets trapped in individual modes (recall slow mixing demo)
- **Question:** Can Metropolis Adjustment step be directly applied?

A Simple Calculation (Building on Metropolis Ingenuity)

- Take the balance equation

$$a(x', x)\, q(x'|x)\; p(x) \;=\; a(x, x')\, q(x|x')\; p(x')$$

- This holds if and only if:

$$\log a(x', x) + \log q(x'|x) + \log p(x) \;-\; \log a(x, x') - \log q(x|x') - \log p(x') = 0.$$

- Which holds if and only if

$$\nabla_x \log a(x', x) + \nabla_x \log q(x'|x) + \nabla_x \log p(x) \;-\; \nabla_x \log a(x, x') - \nabla_x \log q(x|x') - \nabla_x \log p(x') = 0. \quad (*)$$

- One direction in the above is obvious. For the other direction, rough idea:
- If (*) holds then

$$a(x', x)\, q(x'|x)\; p(x) \;= C(x')\, a(x, x')\, q(x|x')\; p(x'),$$

for all x for some function $C(x')$. Now letting $x = x'$ we can immediately observe that $C(x') = 1$, and detailed balance condition holds.

Training of Acceptance Function

- Suppose that a DNN with last layer a sigmoid function is given.
- Suppose that the score of a dataset $\nabla_x \log p(x)$ is estimated (using classical score estimation techniques)
- Model the Acceptance function $\boldsymbol{a(x', x)}$ with a deep neural Network DNN.
  - The DNN last layer must guarantee that its output is in [0,1]. ‘
    - For instance, the sigmoid function can be a choice for last layer.
- Using the training data and the proposal function create batches of pairs of $(\boldsymbol{x}, \boldsymbol{x'})$ (for each data point $\boldsymbol{x}$ create many proposed points $\boldsymbol{x'}$.
- Minimize $\boldsymbol{L\big(a(\cdot,\cdot)\big)}$ **defined as**

$$||\nabla_x \log a(x', x) + \nabla_x \log q(x'|x) + \nabla_x \log p(x) - \nabla_x \log a(x, x') - \nabla_x \log q(x|x') - \nabla_x \log p(x')||^2$$

using mini-batches of data using standard DNN optimization techniques.

- Note that Autograd calculates $\nabla_x \log a(x', x)$ and $\nabla_{x'} \log a(x', x)$.

Entropy Regularization

- The above minimization problem has infinitely many solutions for acceptance DNN for which the objective function $\boldsymbol{L}\big(\boldsymbol{a}(\cdot,\cdot)\big)$ attains the value of zero for overparameterized solutions (e.g. Deep Neural Networks).
- This is obvious since if any $M \geq 1$ is given then

$$\boldsymbol{L}\big(\boldsymbol{a}(\cdot,\cdot)\big) = \mathbf{0} \rightarrow \boldsymbol{L}\left(\frac{\boldsymbol{a}(\cdot,\cdot)}{\boldsymbol{M}}\right) = 0.$$

- In other words, if $\boldsymbol{a}(\cdot,\cdot)$ is a valid acceptance function, then so is $\frac{\boldsymbol{a}(\cdot,\cdot)}{\boldsymbol{M}}$.
- We need to make sure that we avoid degenerate solutions in the training.

Entropy Regularization

- Consider the neg-entropy function

$$H\big(a(x',x)\big) = a(x',x)\log a(x',x) + \big(1-a(x',x)\big)\log\big(1-a(x',x)\big)$$

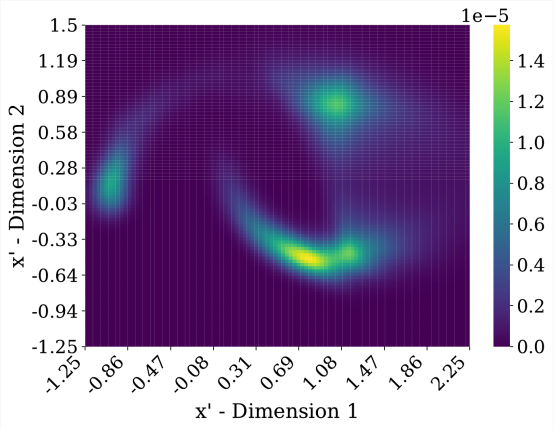
- Define the neg-entropy regularized objective function

$$\boldsymbol{L_r\big(a(\cdot,\cdot)\big) = L\big(a(\cdot,\cdot)\big) + \lambda\, H\big(a(\cdot,\cdot)\big)}$$

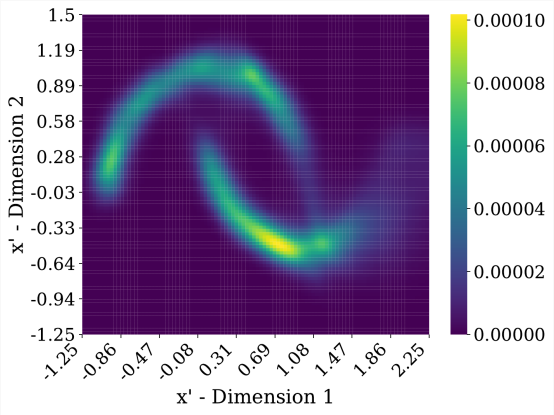
  for $\lambda > 0$.
- By minimizing this regularized objective function, we push the acceptance function away from the degenerate solutions.
- The strength of $\boldsymbol{\lambda}$ determines the trade off that we can optimize using standard DNN training techniques.

Simulation Result

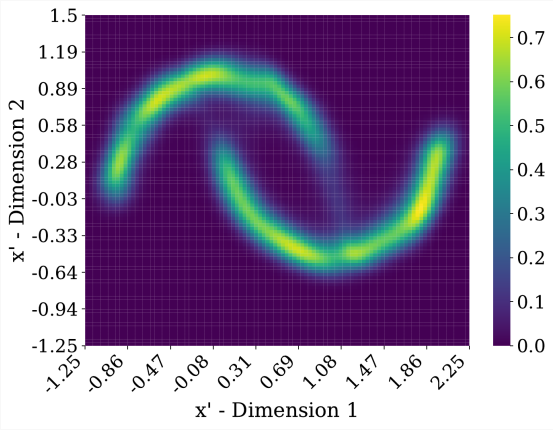
- Acceptance probabilities for initial state x = (0, 0)
  - Without regularization (λ=0): proposals almost always rejected → sampler stalls. Entropy penalty restores practical acceptance probabilities.



λ = 0 (degenerate)



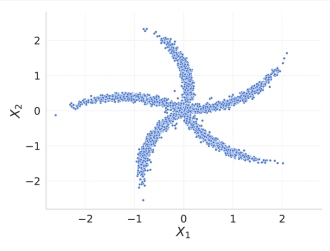
λ = 0.1 (balanced)



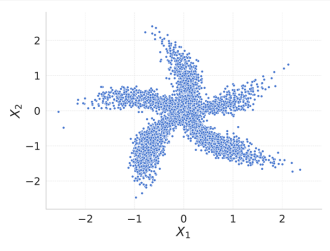
λ = 1.0 (high regularization)

2D Results: Pinwheel Dataset

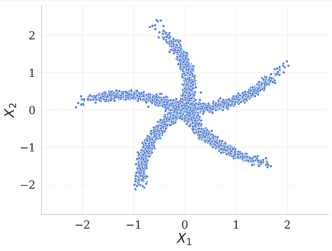
- ULA suffers from discretization bias — assigns uneven weights to the 5 blades
- Score RW and Score MALA correctly balance all modes of the pinwheel distribution
- Score pCN also achieves good coverage — all Score-MH variants outperform ULA



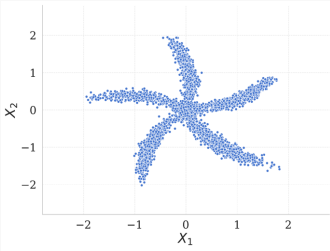
Original



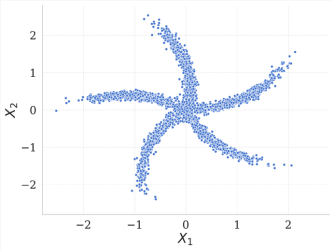
ULA



Score RW



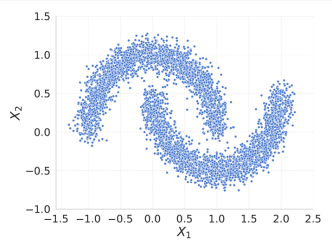
Score MALA



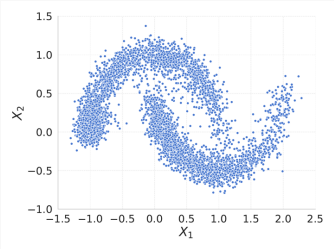
Score pCN

2D Results: Two Moons Dataset

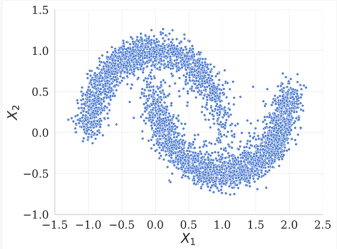
- ULA distorts the crescent shapes and underrepresents the lower moon
- Score MALA achieves best performance $(\mathrm{W}_1 = 0.018,\ W_2 = 0.004,\ MMD = 0.002)$
- Score Metropolis adjustment consistently fixes the mode weight imbalance caused by ULA



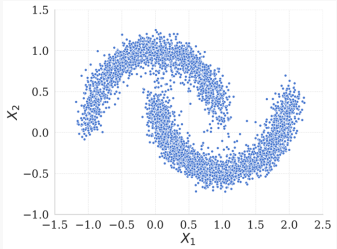
Original



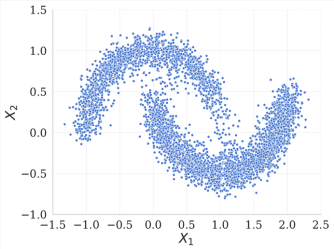
ULA



Score RW

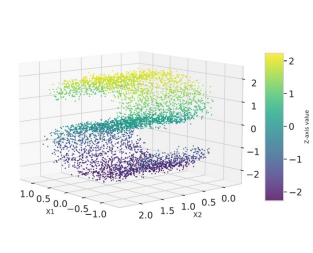


Score MALA

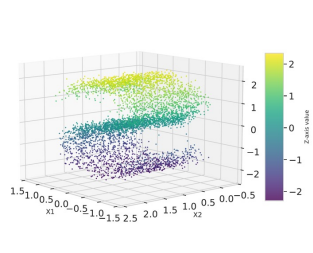


Score pCN

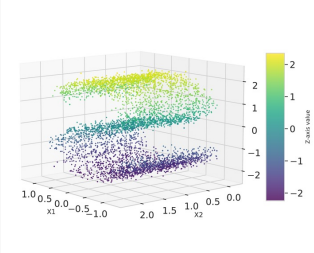
3D S-Curve & Quantitative Results



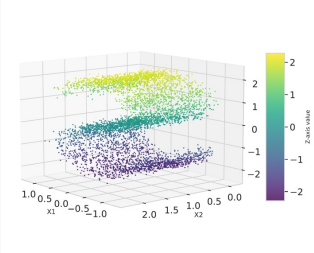
Original



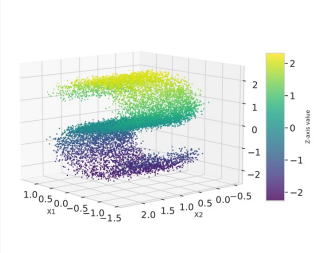
ULA



Score RW



Score MALA



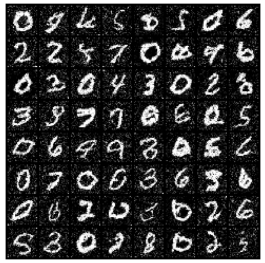
Score pCN

Dataset	ULA			Score RW			Score MALA		
	W1	W2	MMD	W1	W2	MMD	W1	W2	MMD
Moons	0.143	0.096	0.054	0.019	0.007	0.004	**0.018**	**0.004**	**0.002**
Pinwheel	0.106	0.062	0.022	**0.012**	**0.001**	**0.001**	0.086	0.064	0.027
S-curve	0.082	0.055	0.016	0.050	0.015	**0.004**	**0.046**	**0.014**	**0.004**
Swiss Roll	2.881	14.409	0.811	2.270	8.470	0.400	**1.399**	**4.575**	**0.245**

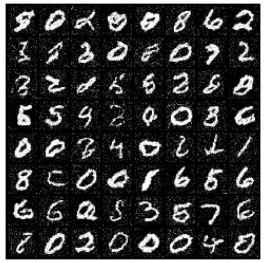
MNIST Generation

- MALA (top) produces cleaner digits across all step sizes
  - ULA (bottom) degrades at larger τ — artifacts appear at $\tau = 1.1$. τ is a fixed parameter that controls the adaptive step size.
- MH correction makes generation more robust to step size choice

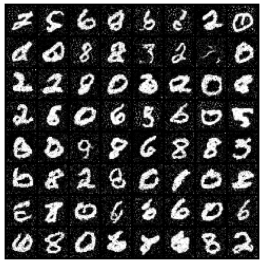
MALA (top):



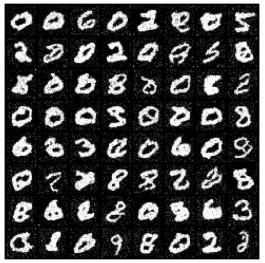
τ=0.5



τ=0.7

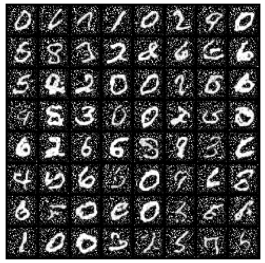


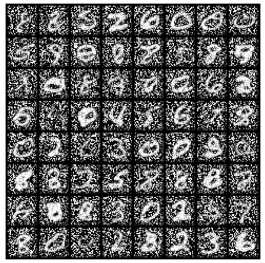
τ=0.9

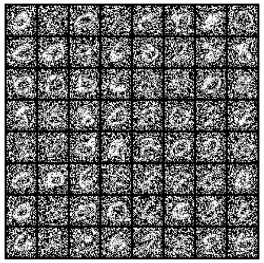


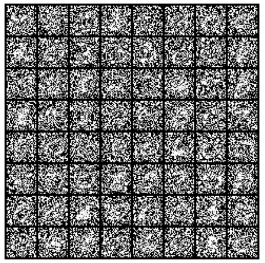
τ=1.1

ULA (bottom):









Extension to Heavy Tails

Fractional Langevin Algorithms

- Sampling from complex heavy-tailed multimodal distributions remains a challenge in Bayesian inference and generative modeling, especially when classical MCMC algorithms struggle to explore complex geometries.
  - If the sampler underexplores tails → downstream decisions become unreliable
- Fractional Langevin samplers driven by symmetric α-stable Levy processes have recently emerged as a powerful alternative, offering long jumps and improved exploration in heavy-tailed regimes.
- The dynamics is analogous to Langevin dynamics, but various changes are needed
- Wiener processes are replaced by Levy processes.
  - Gaussian noise is replaced by heavy-tailed α-stable Levy noise
  - The drift term is based on fractional score instead of score

Exact Fractional Undjusted Langevin Algorithms (FULA)

- Let $b(x) = \frac{D^{\alpha-2}(p(x)\,\nabla \log p(x))}{p(x)}$ for $0 < \alpha \leq 2$.
- Then exact Fractional Unadjusted Langevin Algorithm (FULA) takes the form

$$x^{(n+1)} = x^{(n)} + \tau\, b\left(x^{(n)}\right) + \tau^{\frac{1}{\alpha}}\, S\alpha S(1)$$

- where $S\alpha S(1)$ denotes symmetric α-stable distribution with zero location and unit scale factor.
- In general, the Fourier Transform (characteristic function) of a symmetric α-stable distribution with location parameter μ and scale parameter γ is known to be

$$\varphi(t) = \exp(i\mu t - \gamma^{\alpha}|t|\hat{}\alpha)$$

Why Fractional Langevin?

- Classical Langevin uses Gaussian noise — effective locally but problematic on:
  - Multimodal targets: hard to jump between distant modes
  - Heavy-tailed targets: tails are severely underexplored
- For fractional Langevin
  - Most moves are still moderate (same as Gaussian)
  - But occasional long jumps help exploration across modes and distant regions
  - Especially attractive when the target itself has heavy-tail behavior
  - However: discretization bias of FULA distorts mode weights and tails → MH correction needed

Fractional Undjusted Langevin Algorithms (FULA)

- FULA is achieved by a practical approximation of exact FULA using fractional calculus for $\alpha \in (1, 2]$.
- In this case, we approximate $b(x) = \frac{D^{\alpha-2}(p(x)\nabla \log p(x))}{p(x)}$ by

$$\tilde{b}(x) = c_\alpha \nabla \log p(x), \qquad c_\alpha = \frac{\Gamma(\alpha-1)}{\Gamma(\alpha/2)^2}$$

- Then Fractional Unadjusted Langevin Algorithm (FULA) step then takes the form

$$x^{(n+1)} = x^{(n)} + \tau\, \tilde{b}\left(x^{(n)}\right) + \tau^{\frac{1}{\alpha}}\, S\alpha S(1)$$

Fractional Langevin Algorithms

- Implementing a classical density-based Metropolis--Hastings (MH) correction for these samplers seems not possible in practice, since neither the target density nor the α-stable proposal density admits a tractable closed form.
  - Note that except for a few special cases (e.g., Cauchy, Gaussian) when $q(\cdot)$ is an α-stable distribution, its density does not admit a closed-form expression and is typically characterized only via its Fourier transform.
    - The proposal function does not have a closed form → Metropolis adjustment cannot be applied (at least in a straightforward manner).
- In this light, existing Fractional Langevin algorithms can only operate in an unadjusted regime.

Deriving Scores Needed For Score=Based Langevin Adjustment

- q(x′|x) is α-stable: no closed-form density —but has a known exact fractional score:

$$q(x'|x) = \tau^{-d/\alpha} f_\alpha\left(\frac{x' - x - \tau\tilde{b}(x)}{\tau^{1/\alpha}}\right), \qquad r = x' - x - \tau\tilde{b}(x)$$

- Define residuals for forward and reverse proposals:

$$r = x' - x - \tau\tilde{b}(x), \quad r_{\text{rev}} = x - x' - \tau\tilde{b}(x')$$

- Fractional score identity (exact closed form for isotropic α-SαS):

$$S_q^{(\alpha)}(x'|x) = \frac{D^{\alpha-2}(q(x'|x)\,\nabla_{x'}\log q(x'|x))}{q(x'|x)} = -\frac{r}{\alpha\,\tau}$$

- Approximation bridge via c_α (same relation used to derive the drift $\tilde{b}(x) \approx c_\alpha\,\nabla\, log\, p(x)$):

$$S_f^{(\alpha)}(z) \approx c_\alpha \nabla_z \log f_\alpha(z) \quad \Rightarrow \quad \nabla_{x'}\log q(x'|x) \approx \frac{S_q^{(\alpha)}(x'|x)}{c_\alpha} = -\kappa r, \quad \kappa = \frac{1}{\alpha c_\alpha \tau}$$

Deriving Scores Needed For Score=Based Langevin Adjustment

- Chain rule through the mean map $m(x) = x + \tau\tilde{b}(x)$ gives the x-gradient

$$\frac{\partial r}{\partial x} = -(I + \tau J_{\tilde{b}}(x)) \quad \Rightarrow \quad \nabla_x \log q(x'|x) \approx \kappa\big(r + \tau J_{\tilde{b}}(x)^\top r\big)$$

- By symmetry we can obtain the two remaining gradients for the reverse proposal $q(x|x')$.

$$(1)\ \nabla_{x'} \log q(x'|x) \approx -\kappa\, r \qquad (2)\ \nabla_x \log q(x'|x) \approx \kappa\big(r + \tau J_{\tilde{b}}(x)^\top r\big)$$

$$(3)\ \nabla_x \log q(x|x') \approx -\kappa\, r_{\text{rev}} \qquad (4)\ \nabla_{x'} \log q(x|x') \approx \kappa\big(r_{\text{rev}} + \tau J_{\tilde{b}}(x')^\top r_{\text{rev}}\big)$$

- These four gradients are fully computable from (x, x'), the score network $\tilde{b}(\cdot)$, and its Jacobian $J_{\{\tilde{b}\}}(\cdot)$ —no density evaluation is needed.
- They can be plugged directly into the Gradient Detailed Balance loss to train the acceptance network.

MAFLA: Extended SBM Objectives

- MAFLA extends Score Based Metropolis Adjustment with an α-norm loss term to better match the α-stable geometry

$$\Delta_p = \nabla \log p(x') - \nabla \log p(x), \quad \Delta_q = \nabla \log q(x|x') - \nabla \log q(x'|x)$$

$$\mathcal{L}_2(a) = \mathbb{E}[\|\nabla \log a(x', x) - \nabla \log a(x, x') - \Delta_p - \Delta_q\|_2^2]$$

$$\mathcal{L}_\alpha(a) = \mathbb{E}[\|\nabla \log a(x', x) - \nabla \log a(x, x') - \Delta_p - \Delta_q\|_\alpha^\alpha]$$

- Combined regularized objective (used in MAFLA training):

$$\mathcal{L}_r(a) = \mathcal{L}_2(a) + \lambda_\alpha \mathcal{L}_\alpha(a) + \lambda \mathbb{E}[H(a(x', x))]$$

Training of The Acceptance Network

- Consider an acceptance network with the last layer given by the sigmoid function

$$a_\phi(x', x) = \sigma(g_\phi(x', x)), \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

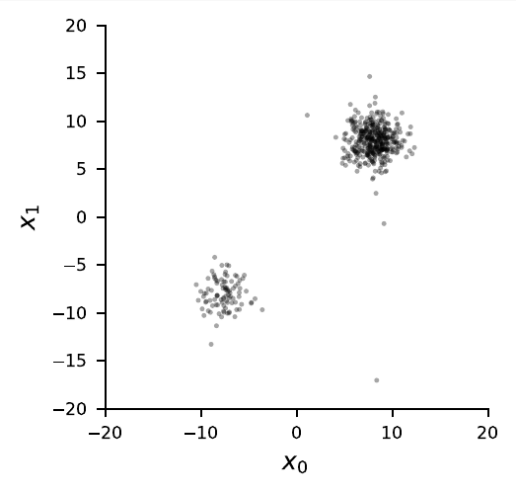
- In order to stabilize training do Curriculum Interpolation:

$$x' = \eta\, v' + (1 - \eta)\, \tilde{x}, \quad \tilde{x} \sim p, \quad v' \sim q(\cdot|\tilde{x}), \quad \eta \uparrow 1$$

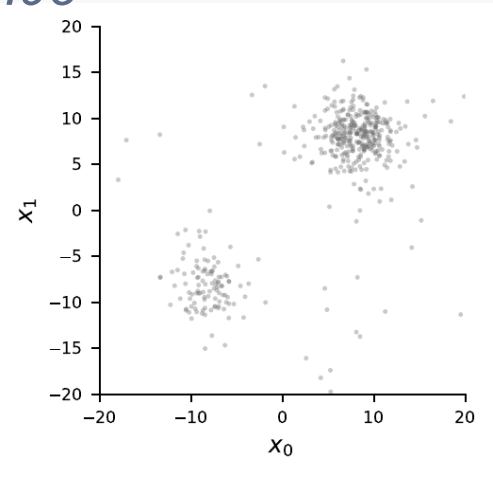
- Start with $\eta \approx 0$: interpolated points x' stay near data region $\tilde{x} \sim p$
- Gradually increase $\eta \to 1$: training transitions toward actual proposals from $q(\cdot|\tilde{x})$
- This avoids early training instability when the acceptance network is randomly initialized
- Final training uses $\eta = 1$: full fractional Levy proposal.

Heavy-Tailed Mixture: Qualitative Results

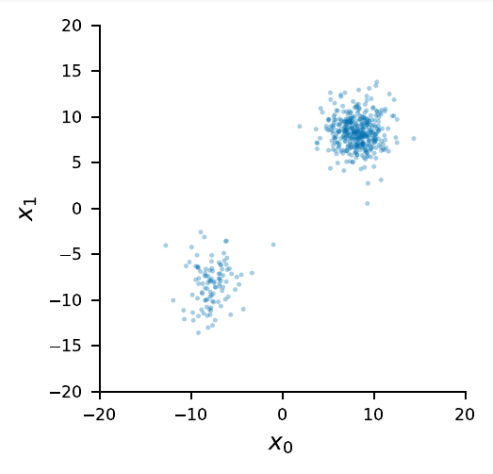
2D α-stable mixture, α = 1.95



Target
(0.2, 0.8)



FULA
(0.26, 0.74)



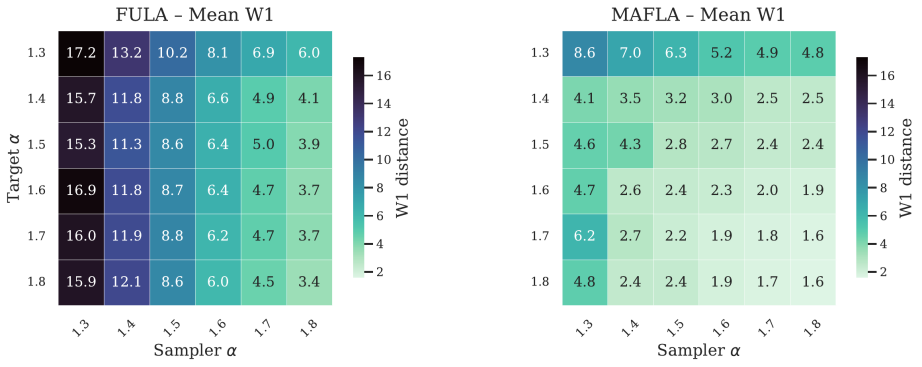
MAFLA
(0.21, 0.79)

- FULA distorts mixture weights: 0.26 vs. true 0.20 for minority mode
- MAFLA recovers correct weights (0.21/0.79) via MH-style correction
- MH correction dramatically reduces both W_1 distance and tail error

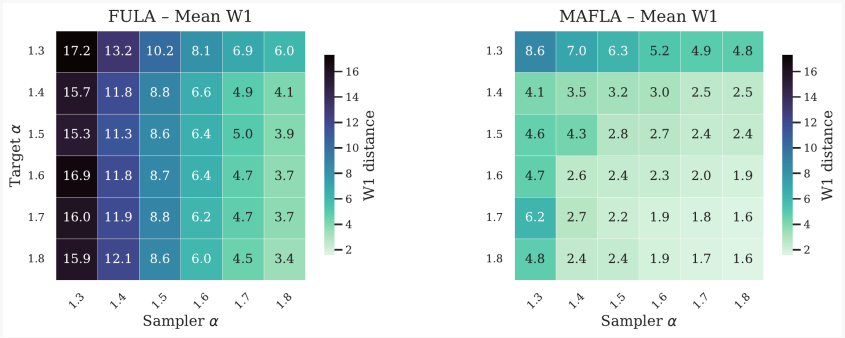
W_1: 4.87 (FULA) → 0.52 (MAFLA)

Robustness to α Misspecification

- In practice, the true α of the target is unknown — we test misspecification across an $(\alpha_{tgt}, \alpha_{prop})$ grid

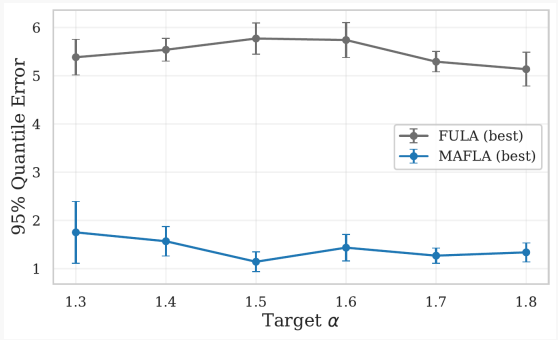


FULA: mean W_1



MAFLA: mean W_1

- MAFLA consistently lower W_1 and quantile errors than FULA across all $(\alpha_{tgt}, \alpha_{prop})$ combinations
- MAFLA less sensitive to misspecification between proposal and target α — more robust in practice

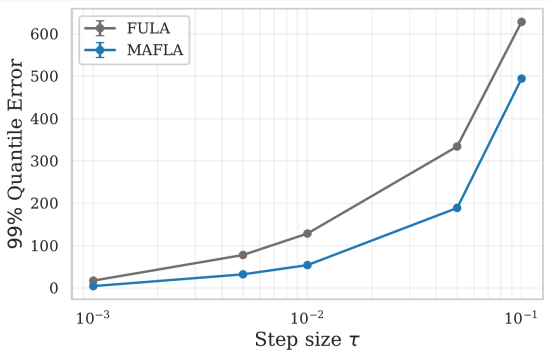


Best $q_{0.95}$ error

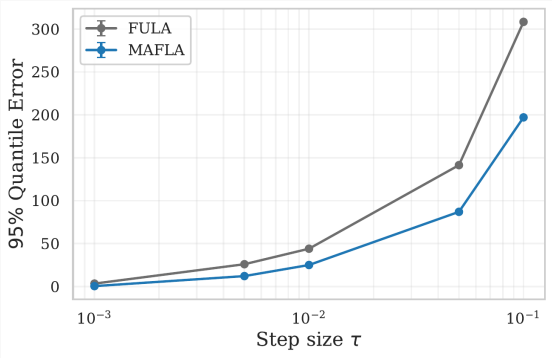
Step-Size Sensitivity

- FULA errors grow significantly with step size τ — making tuning critical
- MAFLA remains consistently better across all τ values (correction absorbs discretization error)

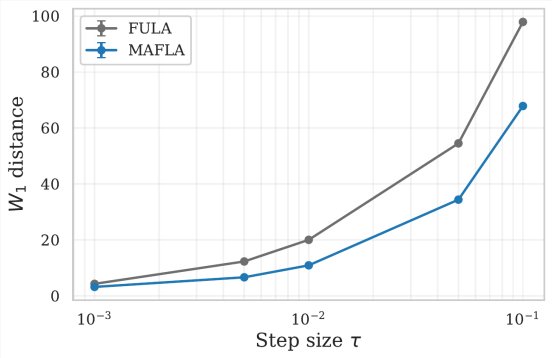
4D α-stable mixture, α = 1.5



$q_{0.99}$ error vs. τ



$q_{0.95}$ error vs. τ

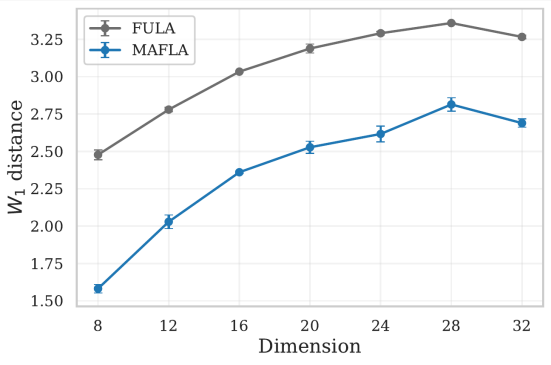


W_1 vs. τ

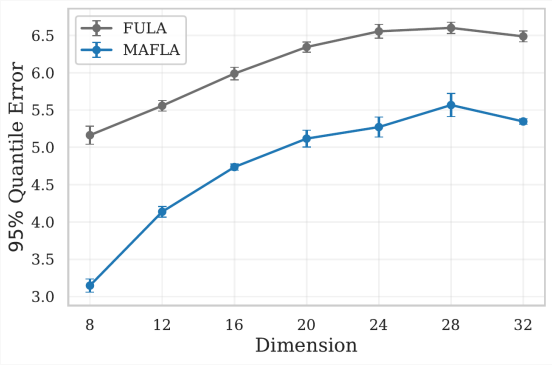
Scaling with Dimension

- MAFLA improvements remain as dimension increases.

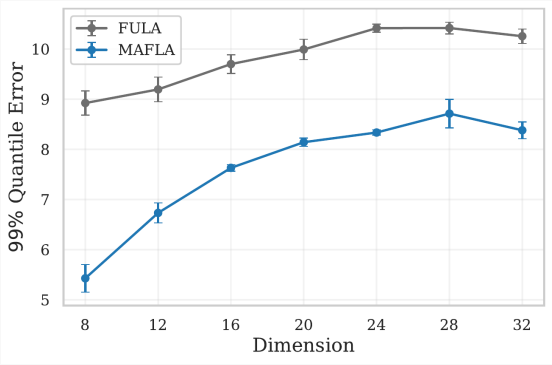
Bimodal α-stable mixture, α = 1.9, d ∈ {8, …, 32}



W_1 vs. dimension



$q_{0.95}$ error vs. dimension



$q_{0.99}$ error vs. dimension

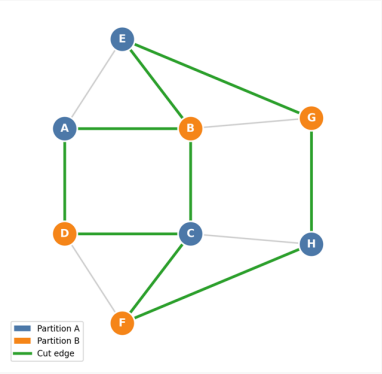
Applications to Combinatorial Optimization

- Discrete combinatorial problems → continuous energy landscape → sample from Gibbs distribution

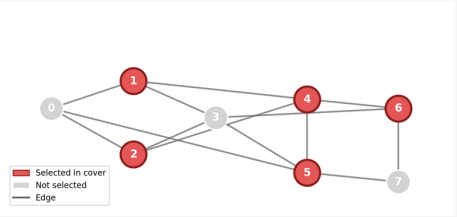
$$\pi(u) \propto \exp(-E(u)/\eta)$$

  - This tests MAFLA well: highly non-convex, many local minima, requires long jumps
    - Fractional noise helps cross energy barriers
    - MH correction stabilizes long jumps and improves final solution quality

- Two canonical NP-hard graph problems tested on Erdős-Rényi (ER) and Barabási-Albert (BA) graphs
- Continuous relaxations with oracle scores — MaxCut (tanh) and Vertex Cover (sigmoid)



MaxCut



Vertex Cover

Max-Cut Results

N = 64 and N = 256 nodes — BA and ER graphs

- MAFLA achieves the best mean cut value and best-found solution in every setting

N	Sampler	BA (m=2) Energy	BA (m=2) Cut↑	BA (m=2) Best↑	ER (p=0.1) Energy	ER (p=0.1) Cut↑	ER (p=0.1) Best↑
64	ULA	−1.20	67.97	90	−3.89	214.50	253
64	FULA	−2.93	74.46	92	−6.22	222.21	256
64	MALA	−0.99	67.23	87	−3.23	212.32	249
64	**MAFLA**	**−3.66**	**77.72**	**100**	**−6.85**	**225.03**	**264**
256	ULA	−1.22	265.48	311	−15.24	3391.87	3530
256	FULA	−4.88	295.13	335	−24.83	3455.98	3621
256	MALA	−1.02	264.25	305	−12.46	3373.21	3519
256	**MAFLA**	**−7.31**	**316.59**	**361**	**−34.35**	**3536.37**	**3728**

- MAFLA: N=64 BA: best 100/100 graphs solved | N=256 ER: best cut value 3536 vs. 3456 (FULA)
- Fractional noise enables better exploration; MH correction improves solution stability

Vertex Cover Results

- Goal: minimize cover size and uncovered edge fraction — smaller is better

ER graphs, $N \in \{64, 256, 512, 1024\}$

	FULA			MAFLA		
Graph	Cover↓	Best↓	Uncov.↓	Cover↓	Best↓	Uncov.↓
ER-64	46.17±1.96	41	0.144±0.042	**45.02±2.10**	**39**	**0.127±0.035**
ER-256	187.14±4.53	173	0.160±0.025	**181.97±4.27**	**169**	**0.139±0.020**
ER-512	385.83±6.54	364	0.179±0.019	**379.06±6.19**	**358**	**0.158±0.016**
ER-1024	780.81±8.49	753	0.196±0.015	**775.94±8.82**	**747**	**0.184±0.013**

- MAFLA consistently reduces mean cover size, best solution, and uncovered-edge fraction
- Improvements scale to ER-1024: 780.81 (FULA) → 775.94 (MAFLA) mean cover size
- Demonstrates benefit of MH correction for large-scale discrete optimization

Future Directions

- Develop Theory
- Improved Proposals
- Extend to Fractional HMC, Manifold-constrained sampling, and learned proposal distributions.
- Extend Applications: Heavy-tail Bayesian inference, rare-event simulation, large-scale combinatorial optimization, and high-resolution image generation.

Thank You!