Deep Learning Hamiltonian Monte Carlo[1]

Building topological samplers for lattice gauge theories

Sam Foreman[1], Xiao-Yong Jin[2], James Osborn[3]
Argonne National Laboratory

bit.ly/l2hmc-qcd

Abstract

- We generalize the Hamiltonian Monte Carlo (HMC) algorithm with a stack of trainable neural network (NN) layers and evaluate its ability to sample from different topologies in a two-dimensional lattice gauge theory. We demonstrate that our model is able to successfully mix between modes of different topologies, significantly reducing the computational cost required to generate independent gauge field configurations.

Motivation

- Markov Chain Monte Carlo (MCMC) methods are pervasive throughout science and are used in applications ranging from epidemiological modeling to election forecasting.
- Recent developments in ML, together with ever-more-capable hardware has led to a resurgence in developing faster, more efficient simulation techniques.
- In particular, the development of invertible NN architectures has opened the flood-gates for new approaches that are capable of outperforming traditional techniques on particularly challenging distributions.
- Simulations in lattice gauge theory / lattice QCD are limited by our ability to generate independent configurations, making it a prime target for testing novel approaches.
- We propose a generalized version of the L2HMC algorithm [2], and look at applying it to generate configurations for a two-dimensional $U(1)$ lattice gauge theory.

Method

- We begin by introducing distinct neural networks, called ***leapfrog layers*** for each leapfrog step of the HMC update, as shown in **Figure (a).**
  - Denote the leapfrog step (layer) by a discrete index $k = \{0,1,\ldots,N_{\mathrm{LF}}\} \in \mathbb{N}$ where N_{LF} is the total number of leapfrogs.
- Introduce $d \sim \mathcal{U}(+,-)$ (direction—forward/backward) and denote the complete state $\xi = (x, v, d)$, then the ***target distribution*** is given by $p(\xi) = p(x)\cdot p(v)\cdot p(d)$.
- Each leapfrog step transforms $\xi_k \equiv (x_k, v_k, d_k) \rightarrow (x''_k, v''_k, d_k) = \xi''_k$ by passing it through k^{th} leapfrog layer (note the direction, d_k is persistent).
- Consider the forward $d = +1$ direction[2] and introduce the notation:

$$\boxed{v'_k \equiv \Gamma^+_k(v_k;\zeta_{v_k}) = v_k \odot \exp\left(\frac{\varepsilon^k_v}{2}s^k_v(\zeta_{v_k})\right) - \frac{\varepsilon^k_v}{2}\left[\partial_x S_\beta(x_k)\odot\exp\left(\varepsilon^k_v q^k_v(\zeta_{v_k})\right) + t^k_v(\zeta_{v_k})\right]}$$

$$\boxed{x'_k \equiv \Lambda^+_k(x_k;\zeta_{x_k}) = x_k \odot \exp\left(\varepsilon^k_x s^k_x(\zeta_{x_k})\right) + \varepsilon^k_x\left[v'_K \odot \exp\left(\varepsilon^k_x q^k_x(\zeta_{x_k})\right) + t^k_x(\zeta_{x_k})\right]}$$

- We can write a complete leapfrog update as: (compare with **HMC** (i.))

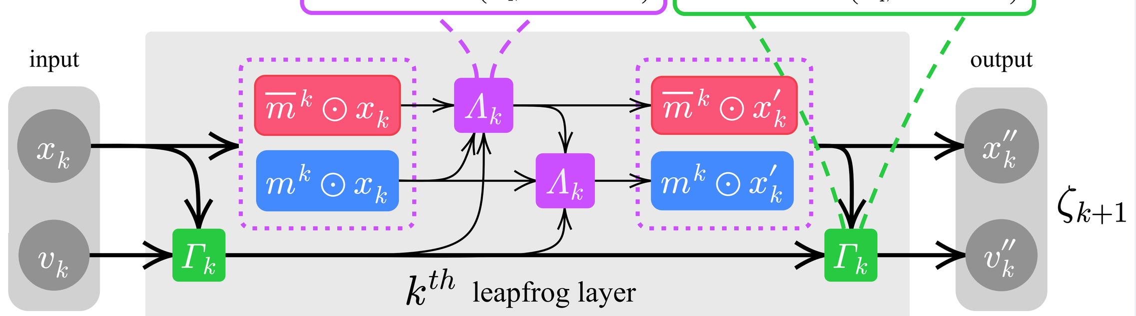
Generalized leapfrog update

1. Half-step (v): $v'_k = \Gamma^{\pm}_k(v_k;\ \zeta_{v_k})$
2. Full-step (1st half of x): $x'_k = \bar{m}^k \odot x_k + m^k \odot \Lambda^{\pm}_k(x_k;\ \zeta_{x_k})$
3. Full-step (2nd half of x): $x'' = \bar{m}^k \odot \Lambda^{\pm}_k(x'_k;\ \zeta_{x'_k}) + m^k \odot x'_k$
4. Half-step (v): $v''_k = \Gamma^{\pm}_k(v'_k;\ \zeta_{v'_k})$

Figure (a.) Illustration of the k^{th} leapfrog layer.

$$\begin{aligned} s^k_x &= \alpha_{s,x}\tanh\left(w^T_{s,x}\cdot z_d + b_{s,x}\right) \\ t^k_x &= w^T_{t,x}\cdot z_d + b_{t,x} \\ q^k_x &= \alpha_{q,x}\tanh\left(w^T_{q,x}\cdot z_d + b_{q,x}\right)\end{aligned}$$

$$\begin{aligned} s^k_v &= \alpha_{s,v}\tanh\left(w^T_{s,v}\cdot z_d + b_{s,v}\right) \\ t^k_v &= w^T_{t,v}\cdot z_d + b_{t,v} \\ q^k_v &= \alpha_{q,v}\tanh\left(w^T_{q,v}\cdot z_d + b_{q,v}\right)\end{aligned}$$



- For reversibility we split the x update into two sub-updates using a binary mask, $m^k = 1 - \bar{m}^k$ and update each half of x sequentially.
- We've introduced the shorthand notation for the networks' inputs: $\zeta_{v_k} \equiv \left(x_k, \partial_x S_\beta(x)\right),\ \zeta_{x_k} \equiv \left(m_k \odot x_k, v_k\right)$
- the *Jacobian factor* of the update $\xi \rightarrow \xi'$, can be easily computed to give: $\left|\frac{\partial v'_k}{\partial v_k}\right| = \exp\left(\frac{1}{2}\varepsilon^k_v s^k_v(\zeta_{v_k})\right),\ \left|\frac{\partial x'_k}{\partial x_k}\right| = \exp\left(\varepsilon^k_x s^k_x(x_k)\right)$.

Method (contd.)

- For target distributions in Euclidean space, $x \in \mathbb{R}^n$, we define a loss function that encourages our sampler to move large distances in the phase space.
- To do this, we can maximize the *expected squared jump distance* (ESJD):

$$\mathscr{L}(\theta) \equiv \mathbb{E}_{p(\xi)}\left[\delta(\xi,\xi')\cdot A(\xi'|\xi)\right],\quad \delta(\xi,\xi') \equiv \|x - x'\|^2.$$

where θ are (collectively) the weights in the NN

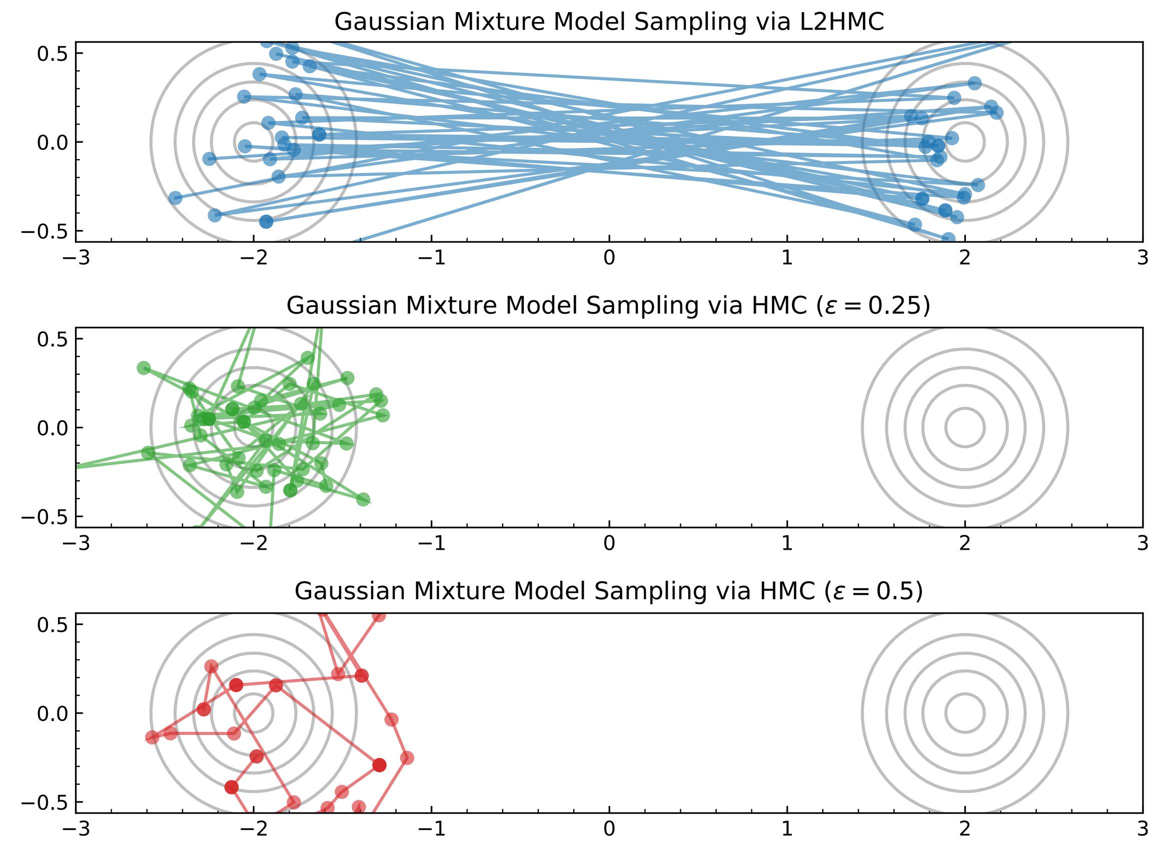


Figure (b.) Illustration of contours from the true target distribution vs samples obtained from both the trained model and generic HMC. We can see that HMC fails to mix between the two modes of the distribution whereas the trained model efficiently jumps between them.

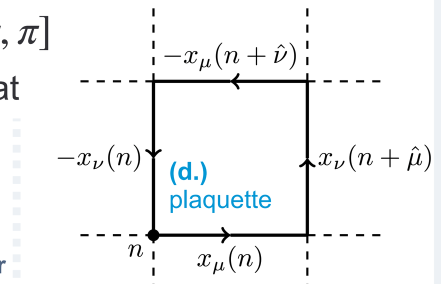
Algorithm 1: Training Procedure

```
input:
  1. Loss function, L_θ(ξ', ξ, A(ξ'|ξ))
  2. Batch of initial states, x
  3. Learning rate schedule, {α_t}_{t=0}^{N_train}
  4. Annealing schedule, {γ_t}_{t=0}^{N_train}
  5. Target distribution, p_t(x) ∝ e^{-γ_t S_β(x)}

Initialize weights θ
for 0 ≤ t < N_train :
  | resample   v ~ N(0, 𝟙)
  | resample   d ~ U(+, −)
  | construct  ξ_0 ≡ (x_0, v_0, d_0)
  | for 0 ≤ k < N_LF :
  |   | propose (leapfrog layer)  ξ'_k ← ξ_k
  | compute   A(ξ'|ξ) = min{1, p(ξ')/p(ξ) |∂ξ'/∂ξ^T|}
  | update    L ← L_θ(ξ', ξ, A(ξ'|ξ))
  | backprop  θ ← θ − α_t ∇_θ L
  | assign    x_{t+1} ← { x'  with probability A(ξ'|ξ)
  |                     { x   with probability (1 − A(ξ'|ξ)).
```

Application to Lattice Gauge Theory

- Let $U_\mu(n) = e^{ix_\mu(n)} \in U(1)$, with $x_\mu(n) \in [-\pi,\pi]$ denote the *link variables*, where $x_\mu(n)$ is a link at the site n oriented in direction $\hat{\mu}$.



$$x_P = \left[x_\mu(n) + x_\nu(n+\hat{\mu}) - x_\mu(n+\hat{\nu}) - x_\nu(n)\right]$$

- **Target distribution:**

$$p_t(x) \propto e^{-\gamma_t\cdot S_\beta(x)}$$

  - Introduce **annealing factor** γ_t, with $\|\gamma_t\| \leq 1$
- **Wilson action:**

$$S_\beta(x) = \beta\sum_P 1 - \cos x_P$$

- **Topological Charge:** Each lattice has a *charge* $\mathcal{Q}_{\mathbb{Z}} \in \mathbb{Z}$

$$\mathcal{Q}_{\mathbb{Z}} = \frac{1}{2\pi}\sum_P \lfloor x_P \rceil \in \mathbb{Z},$$

✗ discrete, **hard to work with**

$$\mathcal{Q}_{\mathbb{R}} = \frac{1}{2\pi}\sum_P \sin x_P \in \mathbb{R}$$

✓ **continuous, differentiable**

where: $\lfloor x_P \rceil = x_P - 2\pi\left\lfloor\frac{x_P+\pi}{2\pi}\right\rfloor$

- **Loss function:** We maximize the expected squared charge difference

$$\mathscr{L}(\theta) = \mathbb{E}_{p(\xi)}\left[-\delta(\xi',\xi)\cdot A(\xi'|\xi)\right]$$

$$\delta(\xi',\xi) = \left[\mathcal{Q}_{\mathbb{R}}(x') - \mathcal{Q}_{\mathbb{R}}(x)\right]^2$$

where $A(\xi'|\xi)$ is given in **HMC** (j.)

Results

- To measure the computational cost of our approach we use the ***integrated autocorrelation time*** $\tau^{\mathcal{Q}_{\mathbb{Z}}}_{\mathrm{int}}$, which can be interpreted as the number of trajectories before an *independent sample* is drawn.
- We can see in **Figure (e.)** that the trained model consistently outperforms generic HMC across $\beta = 2,3,\ldots,7$.

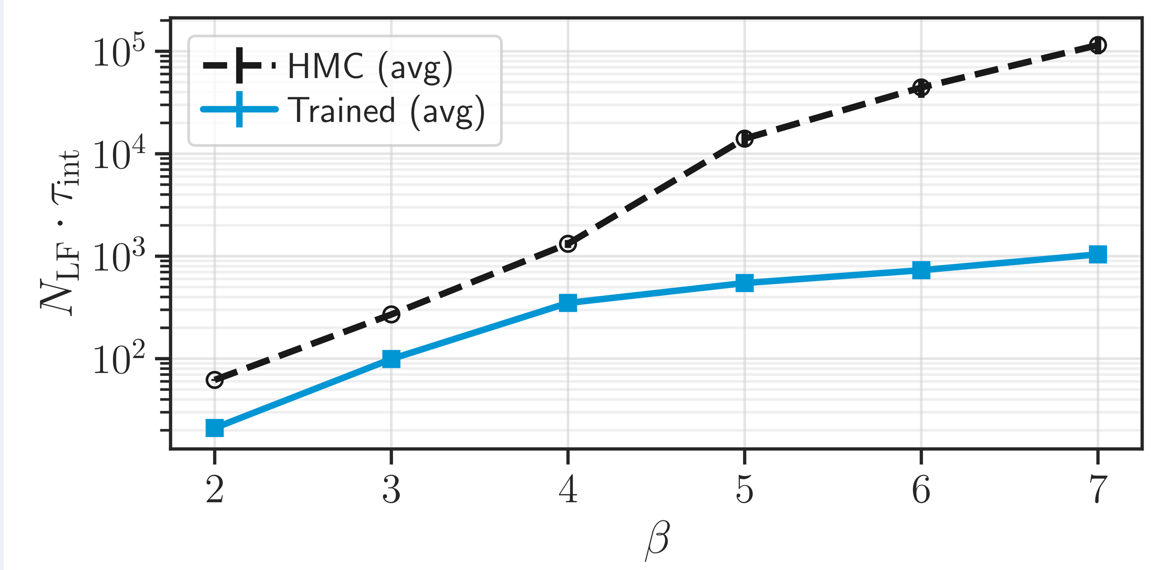


Figure (e.) Estimate of the integrated autocorrelation time $N_{\mathrm{LF}}\cdot\tau^{\mathcal{Q}_{\mathbb{Z}}}_{\mathrm{int}}$ vs β, scaled by N_{LF} to account for simulation cost.

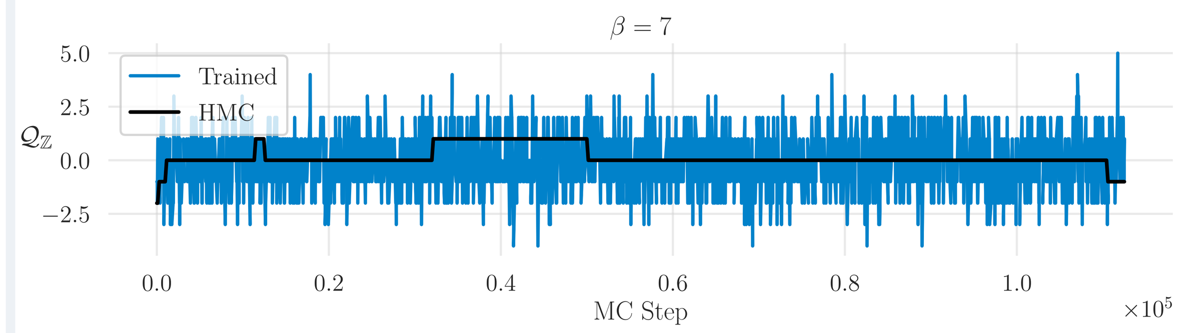


Figure (f.) Illustration of the integer valued topological charge $\mathcal{Q}_{\mathbb{Z}}$ vs MC step for both HMC (black) and the trained model (blue).

- In order to understand the mechanism driving this improved behavior, we looked at how different physical quantities evolve during a single trajectory in the trained model as shown in **Figures (g.1,2,3).**
- We see that our sampler artificially *increased the energy* of the physical system during the first half of the trajectory, before returning back to its original physical value.
- We believe that this ability to vary the energy during the trajectory helps the sampler to overcome energy barriers between topological sectors whereas HMC remains stuck.

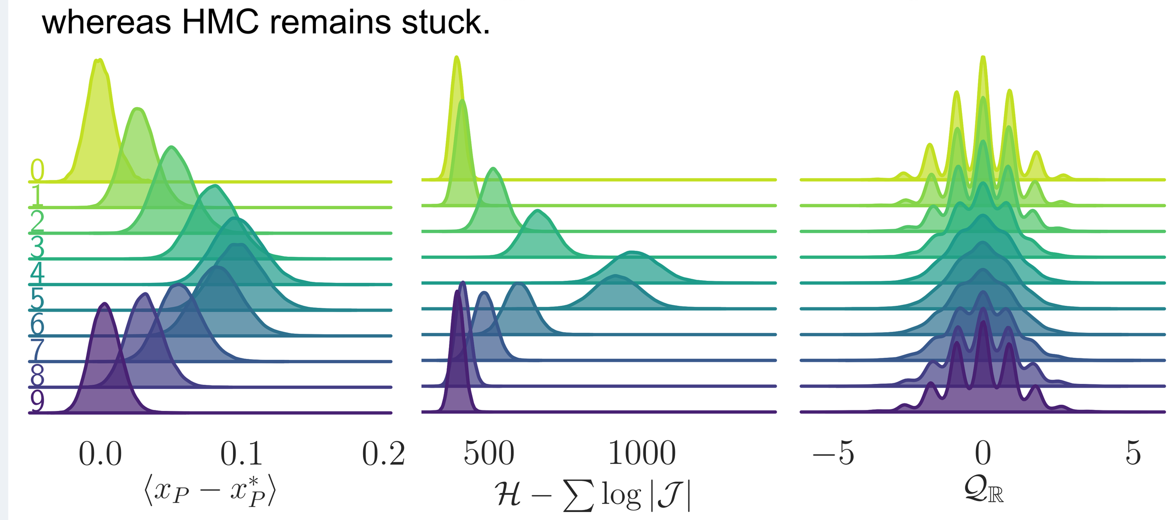


Figure (g.1) The variation in the average plaquette $\langle x_P - x^*_P\rangle$ at intermediate leapfrog layers.

Figure (g.2) The adjusted energy $H - \log|\mathcal{J}|$ at intermediate leapfrog layers.

Figure (g.3) The real-valued topological charge $\mathcal{Q}_{\mathbb{R}}$ at intermediate leapfrog layers.

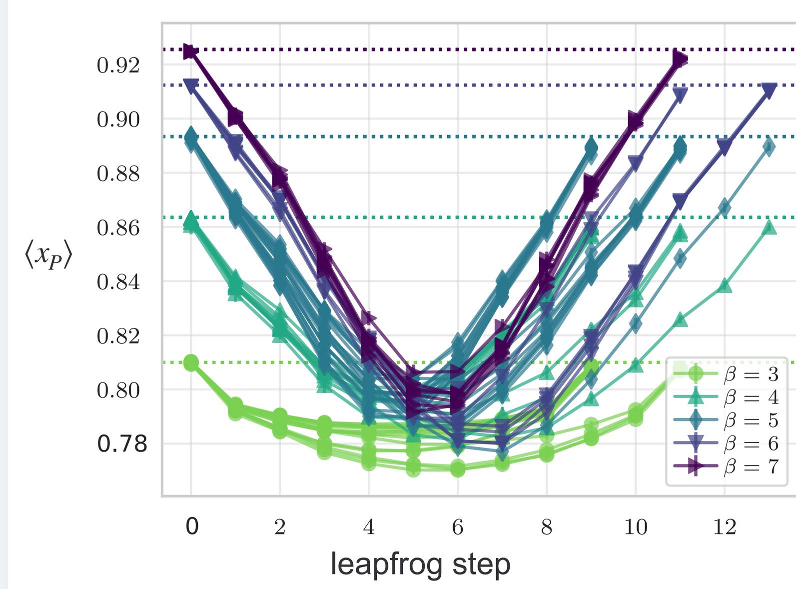


Figure (k.) Illustration of the average plaquette, $\langle x_P\rangle$ vs leapfrog step for different values of β. Note that at $\beta = 7$, the energy is shifted down past the expected value at $\beta = 3$.

Next Steps

- Going forward we plan to continue development of this approach towards more complex theories in higher space-time dimensions (e.g. 2D, 4D SU(3)).

Training Costs

- Our models were trained using Horovod on the ThetaGPU supercomputer at the Argonne Leadership Computing Facility (ALCF). A typical training run on 1 node ($8\times$ NVIDIA A100 GPUs) using a batch size $M = 2048$, hidden layer shapes $[256,256,256]$ for each of the $N_{\mathrm{LF}} = 10$ leapfrog layers, on a 16×16 lattice for 5×10^5 training steps takes roughly 24 hours to complete.

Conclusion

- Presented a generalized version of the L2HMC algorithm—consisting of a stack of leapfrog layers—that improves the existing approaches' flexibility while remaining statistically exact.
- Shown that our trained model successfully mixes between modes of a two-dimensional Gaussian Mixture Model while HMC remains stuck in a local mode.
- Looked at applying the described approach to a two-dimensional U(1) lattice gauge theory.
- Saw that for this lattice gauge model, our trained sampler is capable of significantly outperforming traditional HMC across a range of coupling constants.

References

1. Sam Foreman, Xiao-Yong Jin, James Osborn. Deep Learning Hamiltonian Monte Carlo. **(code: github.com/saforem2/l2hmc-qcd)**
2. Daniel Lévy, M. Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with Neural Networks. abs/1711.09268, 2018.
3. Michael S Albergo, Denis Boyda, Daniel C Hackett, Gurtej Kanwar, Kyle Cranmer, Sébastien Racaniére, Danilo Jimenez Rezende, and Phiala E Shanahan. Introduction to Normalizing Flows for Lattice Field Theory arXiv:2101.08176, 2021.
4. MS Albergo, G Kanwar, and PE Shanahan. Flow-based generative models for Markov Chain Monte Carlo in Lattice Field Theory. Physical Review D, 100(3):034515, 2019.
5. Denis Boyda, Gurtej Kanwar, Sébastien Racaniére, Danilo Jimenez Rezende, Michael S Albergo, Kyle Cranmer, Daniel C Hackett, and Phiala E Shanahan. Sampling using SU(n) Gauge Equivariant Flows. arXiv:2008.05456, 2020.
6. Gurtej Kanwar, Michael S Albergo, Denis Boyda, Kyle Cranmer, Daniel C Hackett, Sébastien Racaniére, Danilo Jimenez Rezende, and Phiala E Shanahan. Equivariant Flow Based Sampling for Lattice Gauge Theory Physical Review Letters, 125(12):121601, 2020

Hamiltonian Monte Carlo (HMC)

- **Goal**: Sample from (difficult) target distribution: $p(x) \propto e^{-S(x)}$
- **Method:**
  1. For $x \in \mathrm{U}(1)^n$, build chain $x_0 \rightarrow x_1 \rightarrow, \ldots, \rightarrow x_N$ such that $x_N \sim p(x)$ as $N \rightarrow \infty$
  2. Introduce $v \sim \mathcal{N}\left(0, I_n\right) \in \mathbb{R}^n$, write joint distribution: $p(x,v) = p(x)p(v) \propto e^{-S(x)}e^{-\frac{1}{2}v^Tv} = e^{-H(x,v)}$
  3. Evolve the system of equations $\dot{x} = \frac{\partial H}{\partial v}$, $\dot{v} = -\frac{\partial H}{\partial x}$ using the **leapfrog integrator (i.)** along $H =$ const: $\xi \equiv (x,v) \rightarrow (x',v') = \xi'$
  4. Accept or reject proposal configuration ξ' using **Metropolis-Hastings** test **(j.)**

(i.) Leapfrog update

1. **Half-step** (v): $\tilde{v} = v - \frac{\varepsilon}{2}\partial_x S(x)$
2. **Full-step** (x): $x' = x + \varepsilon\tilde{v}$
3. **Half-step** (v): $v' = \tilde{v} - \frac{\varepsilon}{2}\partial_x S(x')$

(j.) Metropolis-Hastings

$$x_{i+1} = \begin{cases} x' & \text{w/ prob} \quad A(\xi'|\xi) \\ x & \text{w/ prob} \quad 1 - A(\xi'|\xi)\end{cases}$$

where $A(\xi'|\xi) \equiv \min\left\{1, \frac{p(\xi')}{p(\xi)}\left|\frac{\partial\xi'}{\xi^T}\right|\right\}$

[1] foremans@anl.gov [2] xjin@anl.gov [3] osborn@alcf.anl.gov

[2]To obtain the expression for the reverse direction, we can invert each of the $\Gamma^- \equiv \left(\Gamma^+\right)^{-1}, \Lambda^- \equiv \left(\Lambda^+\right)^{-1}$ functions and perform the updates in the opposite order