LQCD @ PASC23

Sam Foreman

2023-06-05

Generative Modeling & Efficient Sampling

2023-06-29

@ PASC23

Sam Foreman

saforem2/lqcd-pasc23

Argonne NATIONAL LABORATORY

Standard Model

- ⚡ Electricity & Magnetism 🧲
- ⚛️ Quantum Field Theory
  - Nuclear interactions
    - Strong + Weak Force
    - Observed particles
  - Quantum Chromodynamics (**QCD**):
    - Quark / gluon interactions in the nuclueus
    - Analytic progress is *difficult*...[1]
      - Lattice QCD to the rescue! 🚀
- **known to be incomplete!**

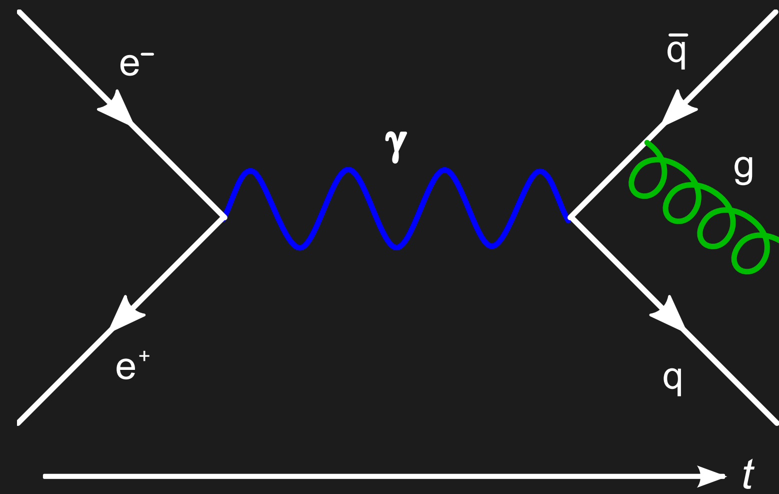


Figure 1: Feynman diagram for

$$e^{+} + e^{-} \to 2\gamma$$

1. Completely stalled ?

Magnetic Moment of the Muon

$$\boxed{a_\mu = \frac{(g_\mu - 2)}{2}}$$

$$a_\mu^{\text{exp}} - a_\mu^{\text{SM}} = (25.1 \pm 5.9) \cdot 10^{-10}$$

can Lattice QCD resolve this?

Fermilab: Muon g-2

The Ring | In transit | Almost Home | Arrival

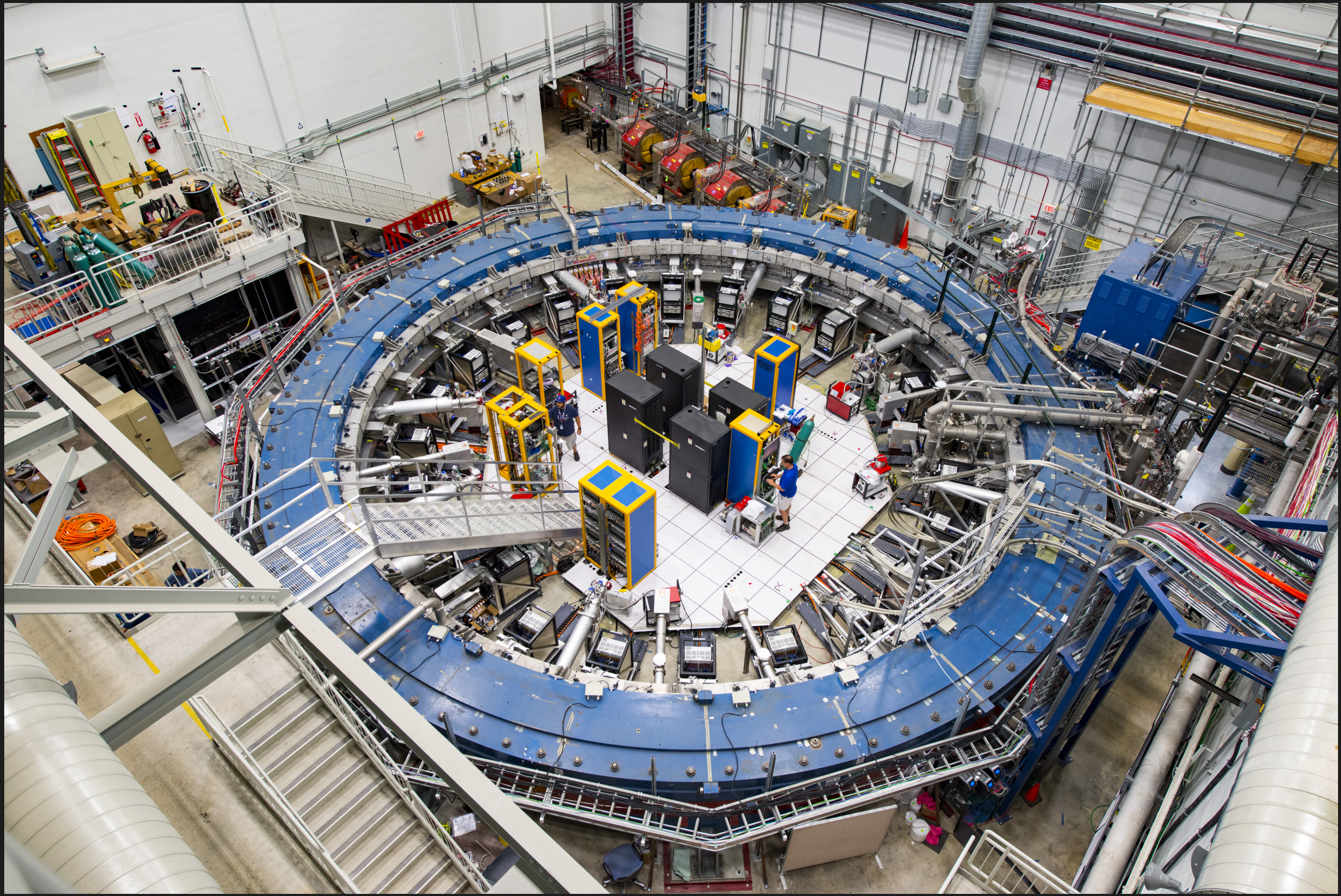


Figure 2: The Muon g-2 ring sits in its detector hall amidst electronics racks, the muon beamline, and other equipment. This impressive experiment operates at negative 450 degrees Fahrenheit and studies the precession (or wobble) of muons as they travel through the magnetic field.

Tension

> *Obviously, an independent cross-check of the BMW lattice result for $a_\mu^{\mathrm{hvp,LO}}$ with sub-percent precision is badly needed.*
> — (*Wittig 2023*)

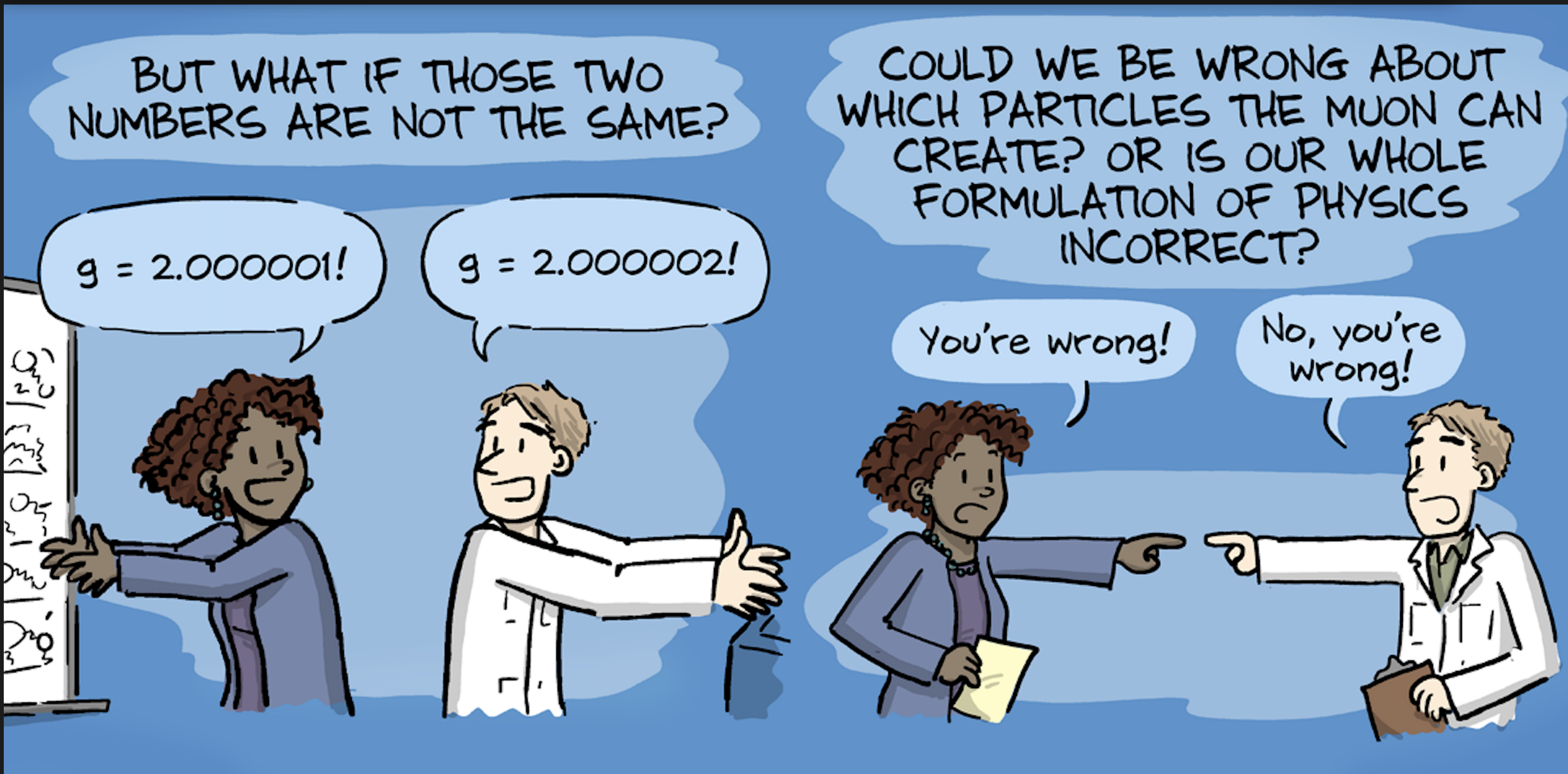


Figure 3: Full cartoon

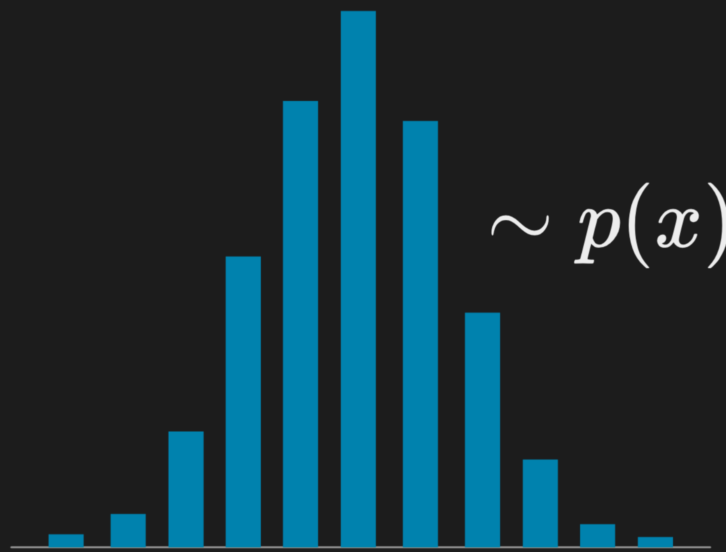
Markov Chain Monte Carlo (MCMC)

Goal

Generate **independent** samples $\{x_i\}$, s.t.[1]

$$\{x_i\} \sim p(x) \propto e^{-S(x)}$$

- Want to calculate observables $\mathcal{O}$:
  $\langle\mathcal{O}\rangle \propto \int [\mathcal{D}x] \;\; \mathcal{O}(x) e^{-S[x]}$



If configurations were independent, we could approximate[2]

$$\langle\mathcal{O}\rangle \simeq \frac{1}{N}\sum_{n=1}^{N}\mathcal{O}(x_n) \quad \Longrightarrow \quad \sigma_{\mathcal{O}}^2 = \frac{1}{N}\mathrm{Var}[\mathcal{O}(x)]$$

1. Here, $\sim$ means "is distributed according to"
2. Our variance on this estimator is then $\sigma_{\mathcal{O}}^2$

Hamiltonian Monte Carlo (HMC)[1]

- Want to (sequentially) construct a chain of states:

$$x_0 \rightarrow x_1 \rightarrow x_i \rightarrow \cdots \rightarrow x_N$$

such that, as $N \rightarrow \infty$:

$$\{x_i, x_{i+1}, x_{i+2}, \cdots, x_N\} \xrightarrow{N\rightarrow\infty} p(x) \propto e^{-S(x)}$$

Trick

- Introduce fictitious momentum $v \sim \mathcal{N}(0,1)$
  - Normally distributed **independent** of x, i.e.

$$p(x, v) = p(x)p(v) \propto e^{-S(x)} e^{-\frac{1}{2}v^T v} = e^{-\left[S(x) + \frac{1}{2}v^T v\right]} = e^{-H(x,v)}$$

1. Fun fact: HMC was *originally* invented for LQCD! (Duane et al. 1987)

Argonne NATIONAL LABORATORY

Hamiltonian Monte Carlo (HMC)

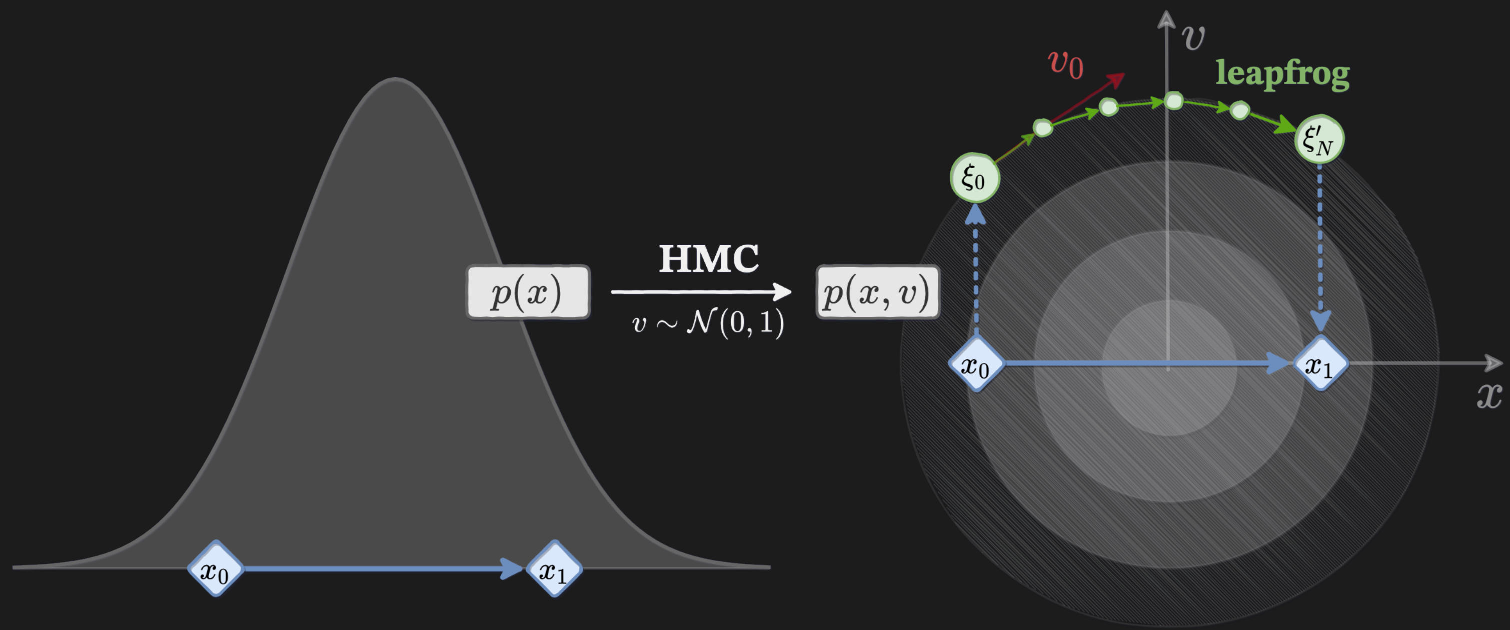
- **Idea**: Evolve the $(\dot{x}, \dot{v})$ system to get new states $\{x_i\}$!
- Write the **joint distribution** $p(x, v)$:

$$p(x, v) \propto e^{-S[x]} e^{-\frac{1}{2}v^T v} = e^{-H(x,v)}$$

Hamiltonian Dynamics

$$H = S[x] + \frac{1}{2}v^T v \Longrightarrow$$

$$\dot{x} = +\partial_v H, \ \dot{v} = -\partial_x H$$



Leapfrog Integrator (HMC)

Hamiltonian Dynamics

$$(\dot{x}, \dot{v}) = (\partial_v H, -\partial_x H)$$

Leapfrog Step

`input` $(x, v) \rightarrow (x', v')$ `output`

$$\tilde{v} := \Gamma(x; v) \ = v - \frac{\varepsilon}{2}\partial_x S(x)$$

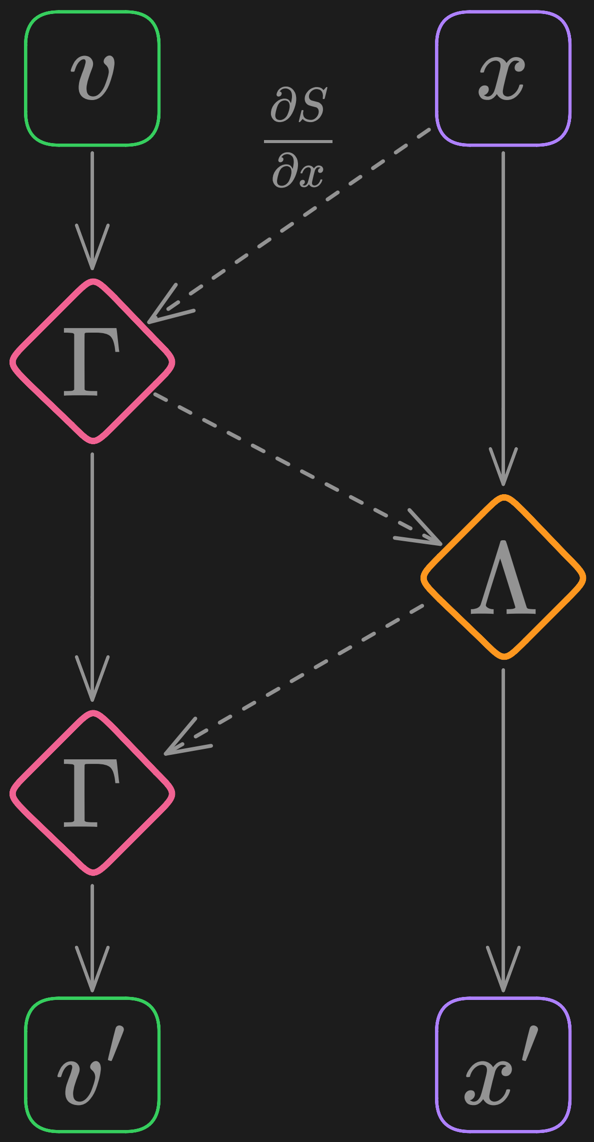
$$x' := \Lambda(x; \tilde{v}) \ = x + \varepsilon\,\tilde{v}$$

$$v' := \Gamma(x'; \tilde{v}) = \tilde{v} - \frac{\varepsilon}{2}\partial_x S(x')$$

Warning!

Resample $v_0 \sim \mathcal{N}(0, 1)$
at the beginning of each trajectory

Note: $\partial_x S(x)$ is the *force*



HMC Update

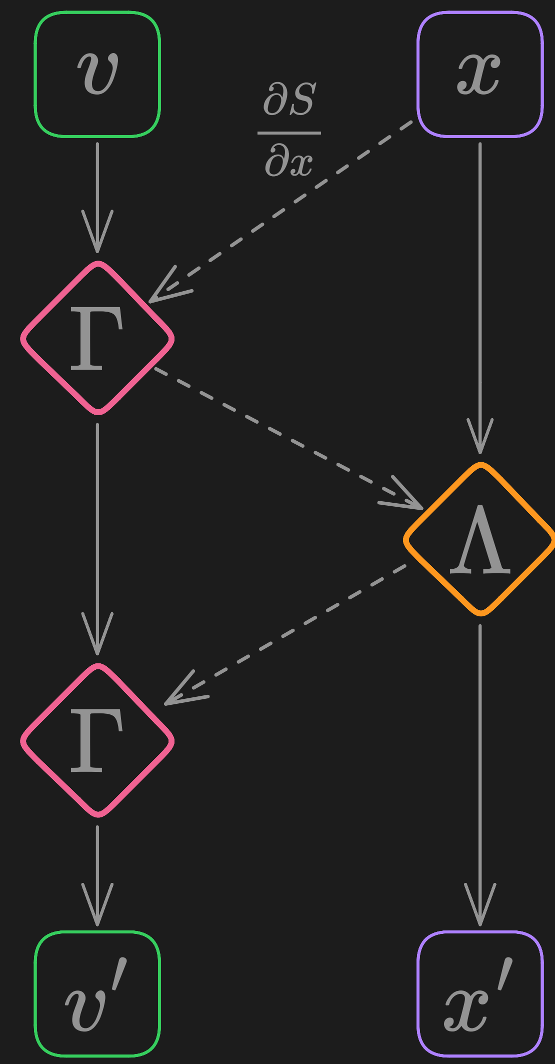
- We build a trajectory of N_{LF} **leapfrog steps**[1]

$$(x_0, v_0) \to (x_1, v_1) \to \cdots \to (x', v')$$

- And propose x' as the next state in our chain

$$\Gamma : (x, v) \to v' := v - \frac{\varepsilon}{2}\partial_x S(x)$$
$$\Lambda : (x, v) \to x' := x + \varepsilon v$$

- We then accept / reject x' using Metropolis-Hastings criteria,
$A(x'|x) = \min\left\{1, \frac{p(x')}{p(x)}\left|\frac{\partial x'}{\partial x}\right|\right\}$



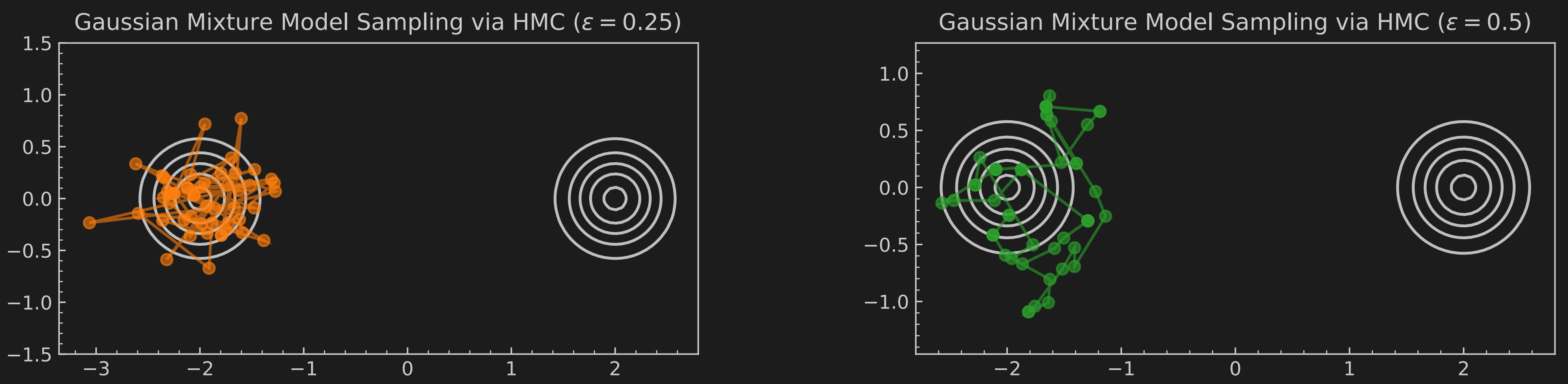
1. We **always** start by resampling the momentum, $v_0 \sim \mathcal{N}(0, 1)$

HMC Demo

Figure 4: HMC Demo

Issues with HMC

- What do we want in a good sampler?
  - **Fast mixing** (small autocorrelations)
  - **Fast burn-in** (quick convergence)
- Problems with HMC:
  - Energy levels selected randomly $\longrightarrow$ **slow mixing**
  - Cannot easily traverse low-density zones $\longrightarrow$ **slow convergence**



HMC Samples with $\varepsilon = 0.25$

HMC Samples with $\varepsilon = 0.5$

Figure 5: HMC Samples generated with varying step sizes ε

L2HMC: Leapfrog Layer

Invertible NN

1. Update $\mathbf{v}$:

$$\mathbf{v}' = \Gamma^{\pm}[\mathbf{v};\zeta_{\mathbf{v}}]$$

2. Update **half** of $\mathbf{x}$ via $\bar{m}_k \odot \mathbf{x}_k$:

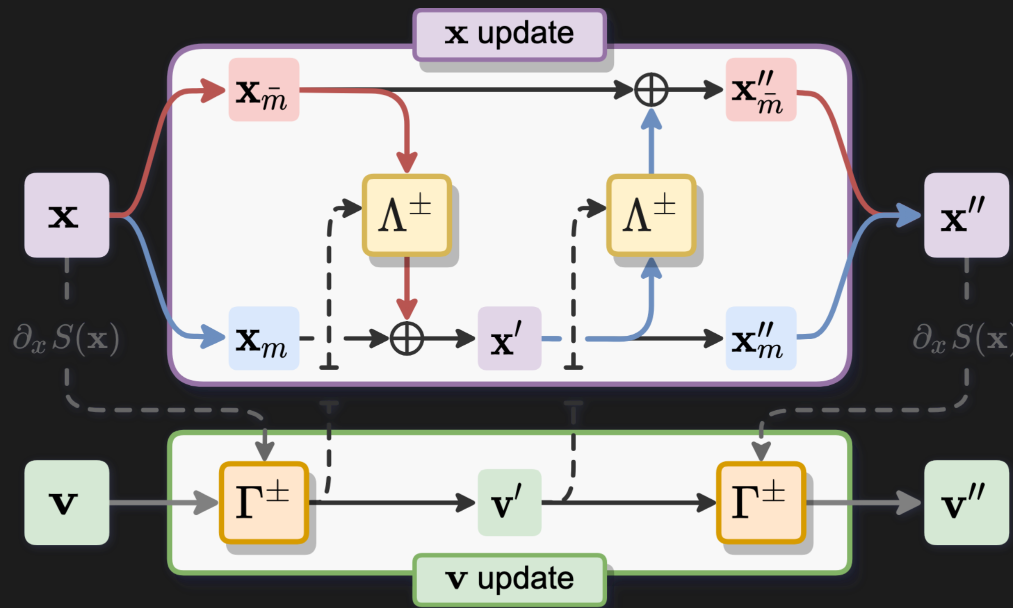
$$\mathbf{x}' = \mathbf{x}_m + \bar{m} \odot \Lambda^{\pm}\left[\mathbf{x}_{\bar{m}};\zeta_{\bar{\mathbf{x}}_k}\right]$$

3. Update (other) **half** via $m^k \odot \mathbf{x}'_k$:

$$\mathbf{x}'' = \mathbf{x}'_m + \bar{m} \odot \Lambda^{\pm}\left[\mathbf{x}'_m;\zeta_{\mathbf{x}'}\right]$$

4. Half-step full $\mathbf{v}$ update:

$$\mathbf{v}'' = \Gamma^{\pm}[\mathbf{v}';\zeta_{\mathbf{v}'}]$$



$$\Gamma^{+}[\mathbf{v}_k;\,\zeta_{\mathbf{v}}] \equiv \overbrace{\mathbf{v}_k \odot \exp\left(\frac{\varepsilon^k_{\mathbf{v}}}{2}\, s^k_{\mathbf{v}}(\zeta_{\mathbf{v}_k})\right)}^{\mathbf{v}\text{ scaling}} - \frac{\varepsilon^k_{\mathbf{v}}}{2}\Big[\overbrace{\partial_x S(x_k) \odot \exp\left(\varepsilon^k_{\mathbf{v}}\, q^k_{\mathbf{v}}(\zeta_{\mathbf{v}_k})\right)}^{\text{force scaling}} + \overbrace{t^k_{\mathbf{v}}(\zeta_{\mathbf{v}_k})}^{\text{translation}}\Big]$$

trainable step sizes

$$\Lambda^{+}[\bar{\mathbf{x}}_k;\,\zeta_{\bar{\mathbf{x}}_k}] \equiv \overbrace{\bar{\mathbf{x}}_k \odot \exp\left(\varepsilon^k_{\bar{\mathbf{x}}}\, s^k_{\bar{\mathbf{x}}}(\zeta_{\bar{\mathbf{x}}_k})\right)}^{\mathbf{x}\text{ scaling}} + \varepsilon^k_{\mathbf{x}}\Big[\overbrace{v'_k \odot \exp\left(\varepsilon^k_{\mathbf{x}}\, q^k_{\mathbf{x}}(\zeta_{\bar{\mathbf{x}}_k})\right)}^{\mathbf{v}\text{ scaling}} + \overbrace{t^k_{\mathbf{x}}(\zeta_{\bar{\mathbf{x}}_k})}^{\text{translation}}\Big]$$

👾 Algorithm: L2HMC Update

1. `input`: $\mathbf{x}$
   - Resample $\mathbf{v} \sim \mathcal{N}(0,1)$, $d \sim \mathcal{U}(+,-)$, and construct $\xi = (\mathbf{x}, \mathbf{v}, \pm)$
2. `forward`: Generate proposal ξ^* by passing initial ξ through N_{LF} leapfrog layers

$$\xi \xrightarrow{\text{LF layer}} \xi_1 \longrightarrow \cdots \longrightarrow \xi_{N_{\mathrm{LF}}} = \xi^*$$

   - Compute the Metropolis-Hastings (MH) acceptance (with Jacobian $\mathcal{J}$)

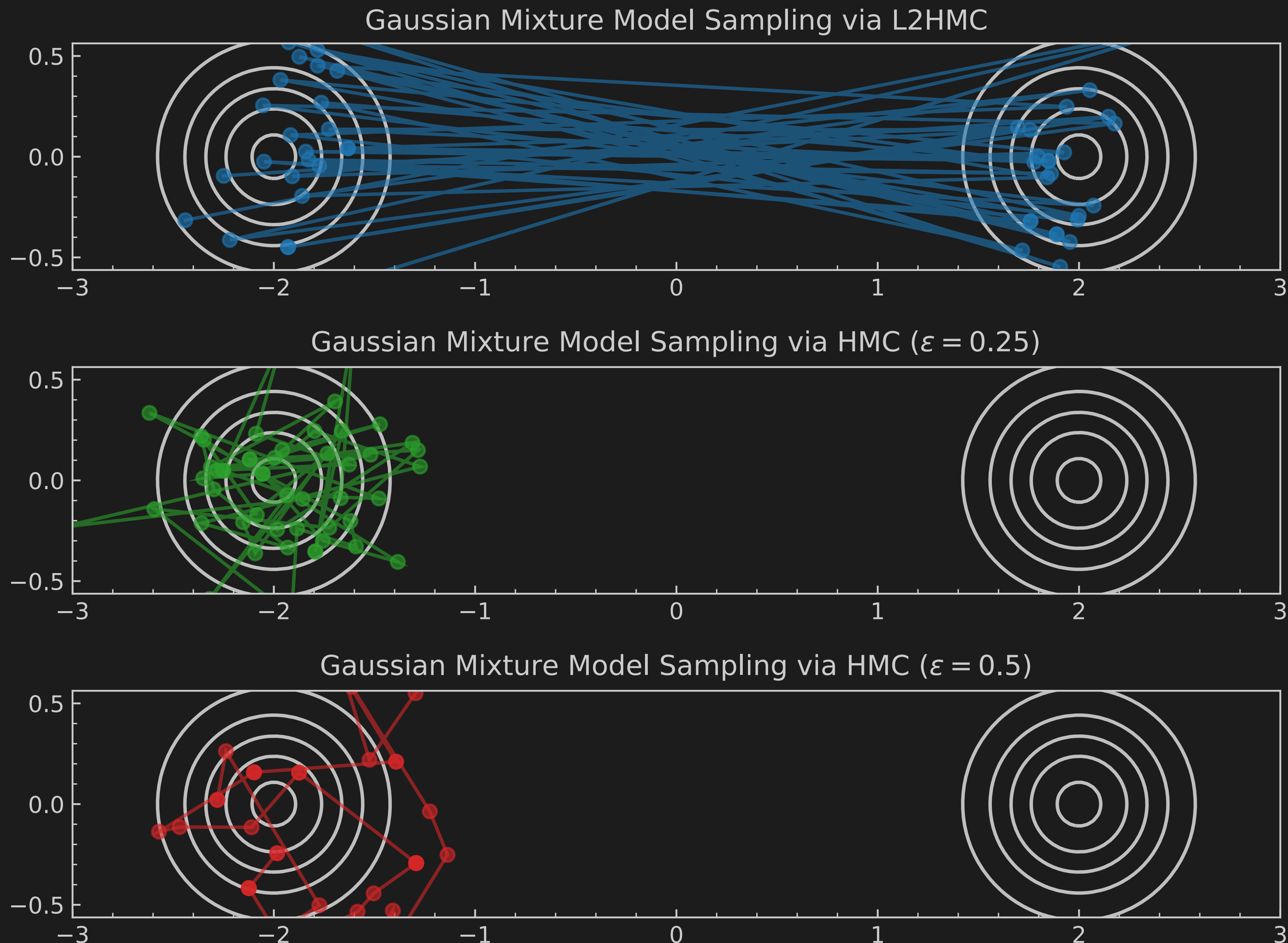
$$A(\xi^*|\xi) = \min\left\{1, \frac{p(\xi^*)}{p(\xi)}\left|\mathcal{J}\left(\xi^*, \xi\right)\right|\right\} \tag{1}$$

3. `backward` (if training):
   - Evaluate the **loss function**[1] $\mathcal{L} \leftarrow \mathcal{L}_\theta(\xi^*, \xi)$ and backprop
4. `return`: Evaluate MH criteria (1) and return accepted config, $\mathbf{x}_{i+1}$

$$\mathbf{x}_{i+1} \leftarrow \begin{cases} \mathbf{x}^* \text{ w/ prob } A(\xi^*|\xi) & ✅ \\ \mathbf{x} \quad \text{w/ prob } 1 - A(\xi^*|\xi) & 🚫 \end{cases}$$

1. For simple $\mathbf{x} \in \mathbb{R}^2$ example, $\mathcal{L}_\theta = A(\xi^*|\xi) \cdot (\mathbf{x}^* - \mathbf{x})^2$

Toy Example: GMM $\in \mathbb{R}^2$



Lattice Gauge Theory (2D $U(1)$)

Link Variables

$$U_\mu(n) = e^{i x_\mu(n)} \in \mathbb{C}, \quad \text{where}$$

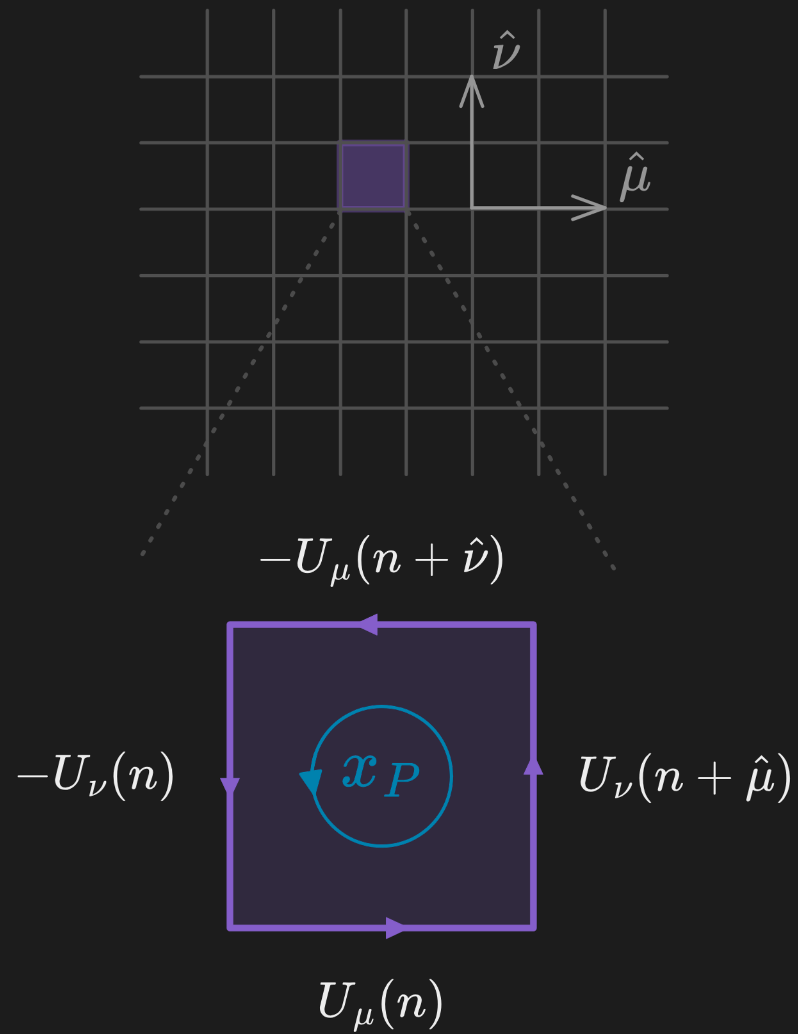
$$x_\mu(n) \in [-\pi, \pi)$$

Wilson Action

$$S_\beta(x) = \beta \sum_P \cos x_P,$$

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

Note: x_P is the product of links around 1×1 square, called a "plaquette"



2D Lattice

Physical Quantities

- To estimate physical quantities, we:
  - calculate physical observables at **increasing** spatial resolution
  - perform extrapolation to continuum limit

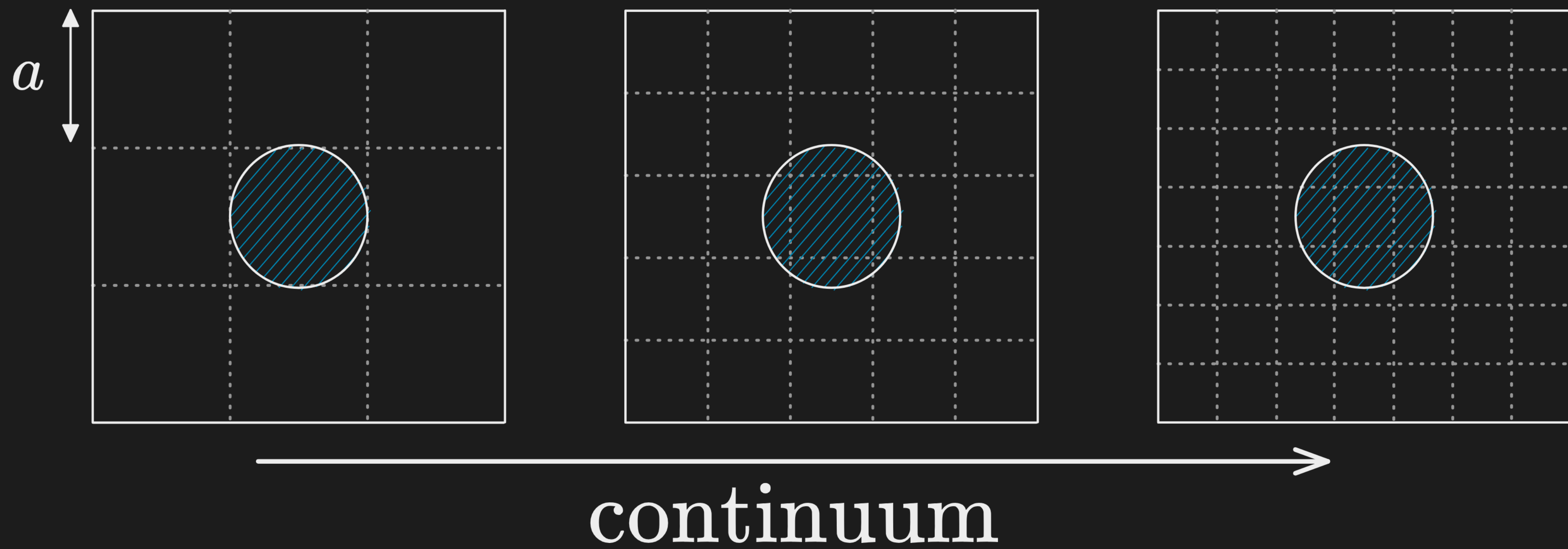


Figure 6: Increasing the physical resolution ($a \to 0$) allows us to make predictions about numerical values of physical quantities in the continuum limit.

Topological Freezing

Topological Charge:

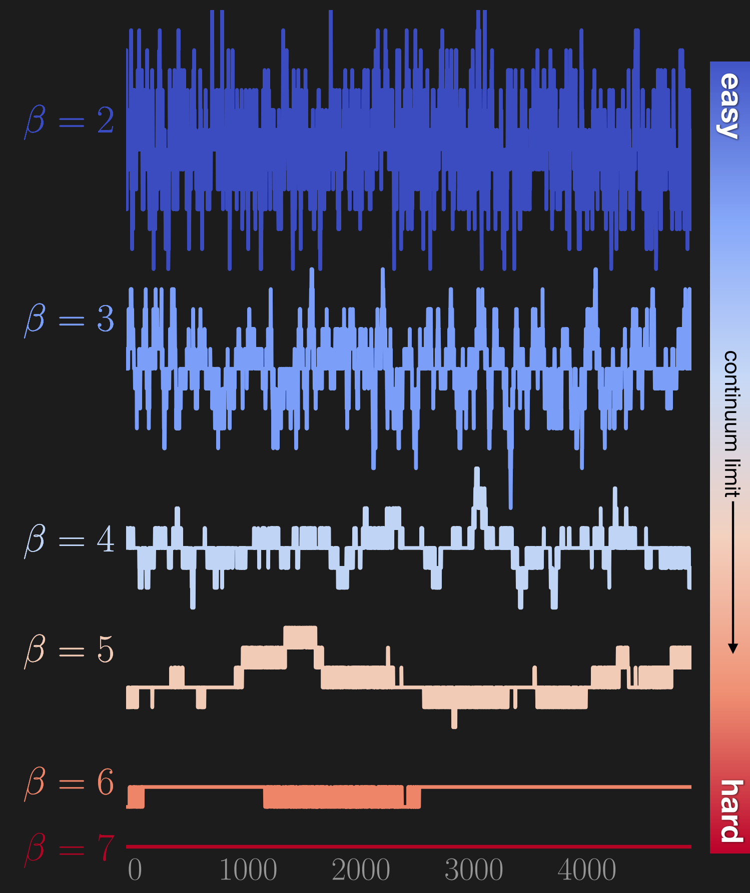
$$Q = \frac{1}{2\pi}\sum_{P}\lfloor x_P \rfloor \in \mathbb{Z}$$

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

Critical Slowing Down

- Q gets stuck!
  - as $\beta \longrightarrow \infty$:
    - $Q \longrightarrow \text{const.}$
    - $\delta Q = (Q^* - Q) \rightarrow 0 \Longrightarrow$
  - # configs required to estimate errors **grows exponentially**: $\tau_{\text{int}}^{Q} \longrightarrow \infty$

Q (HMC)



MD Time

Note $\delta Q \rightarrow 0$ at increasing β

Loss Function

- Want to maximize the *expected* squared charge difference[1]:

$$\mathcal{L}_{\theta}\left(\xi^{*}, \xi\right) = \mathbb{E}_{p(\xi)}\left[-\delta Q^{2}\left(\xi^{*}, \xi\right) \cdot A(\xi^{*}|\xi)\right]$$

- Where:
  - δQ is the *tunneling rate*:

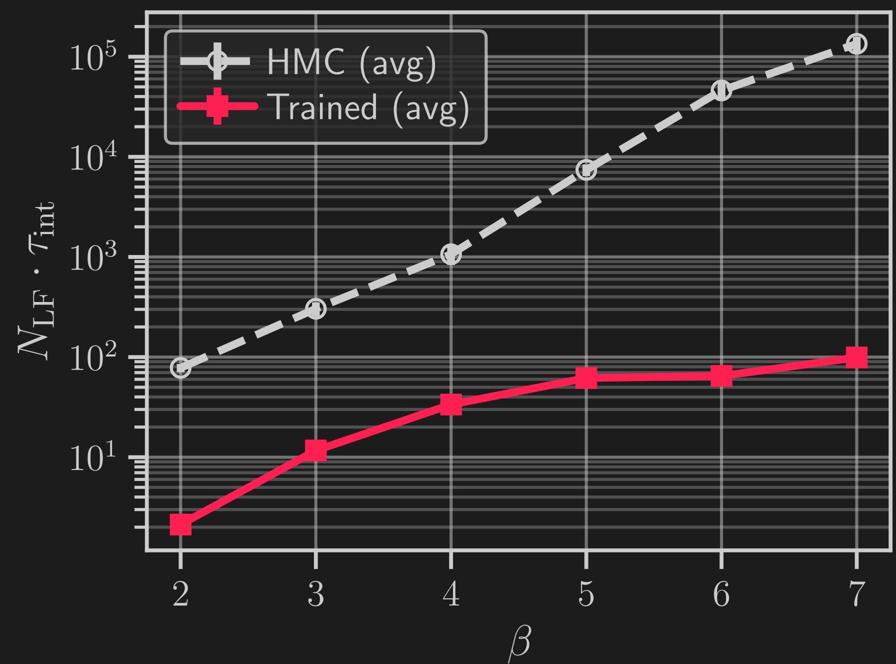
$$\delta Q(\xi^{*}, \xi) = |Q^{*} - Q|$$

  - $A(\xi^{*}|\xi)$ is the probability[2] of accepting the proposal ξ^{*}:

$$A(\xi^{*}|\xi) = \min\left(1, \frac{p(\xi^{*})}{p(\xi)}\left|\frac{\partial \xi^{*}}{\partial \xi^{T}}\right|\right)$$

1. Here, ξ^{*} is the *proposed* configuration (prior to Accept / Reject)
2. $\left|\frac{\partial \xi^{*}}{\partial \xi^{T}}\right|$ is the Jacobian of the transformation from $\xi \to \xi^{*}$

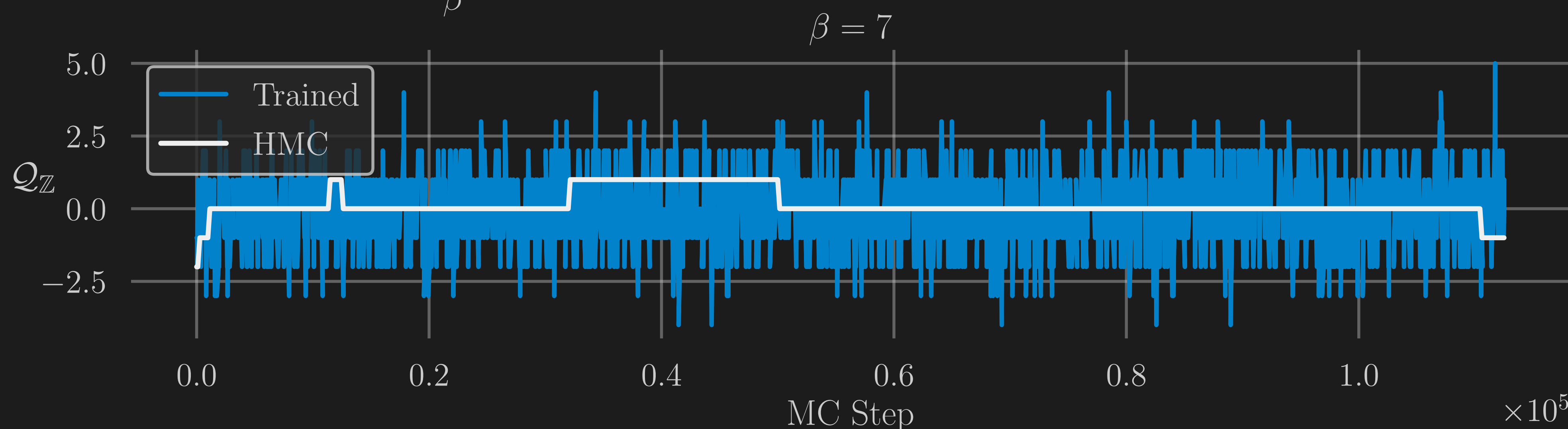
Integrated Autocorrelation time: τ_{int}



Improvement

We can measure the performance by comparing τ_{int} for the **trained model** vs. **HMC**.

Note: lower is better



Integrated Autocorrelation Time

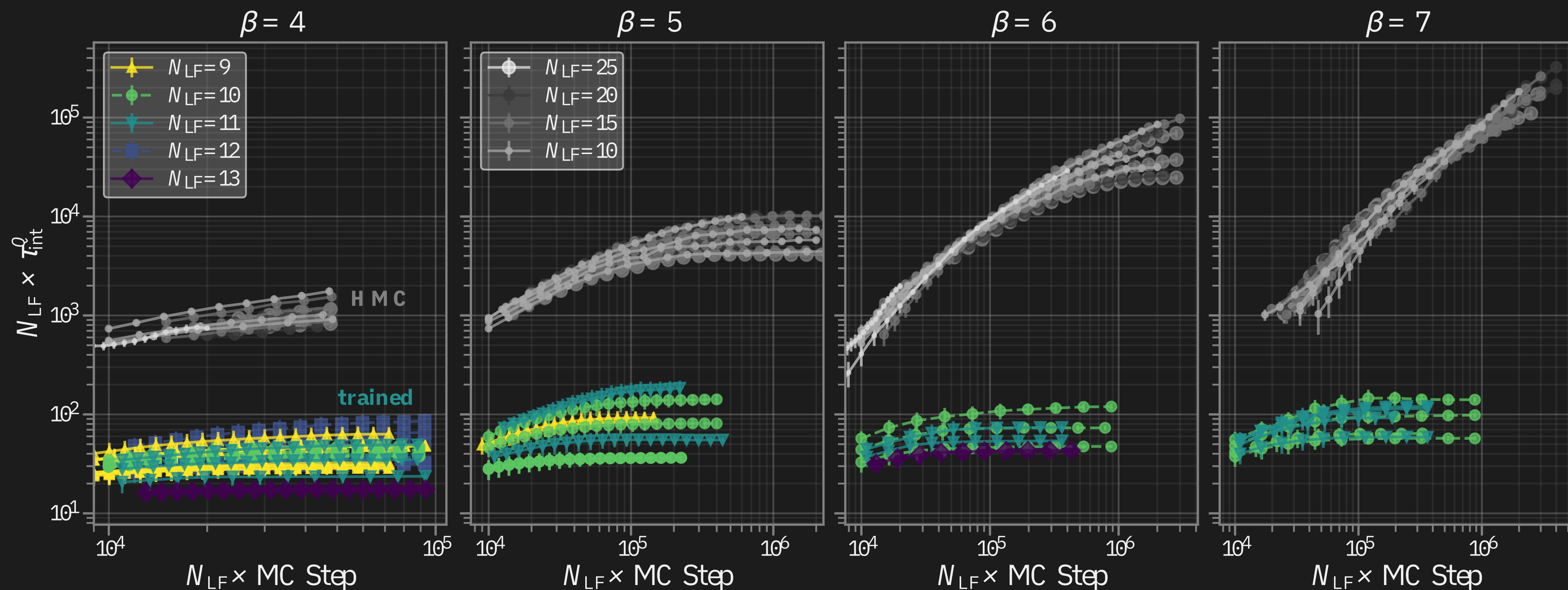


Figure 7: Plot of the integrated autocorrelation time for both the trained model (colored) and HMC (greyscale).

Interpretation

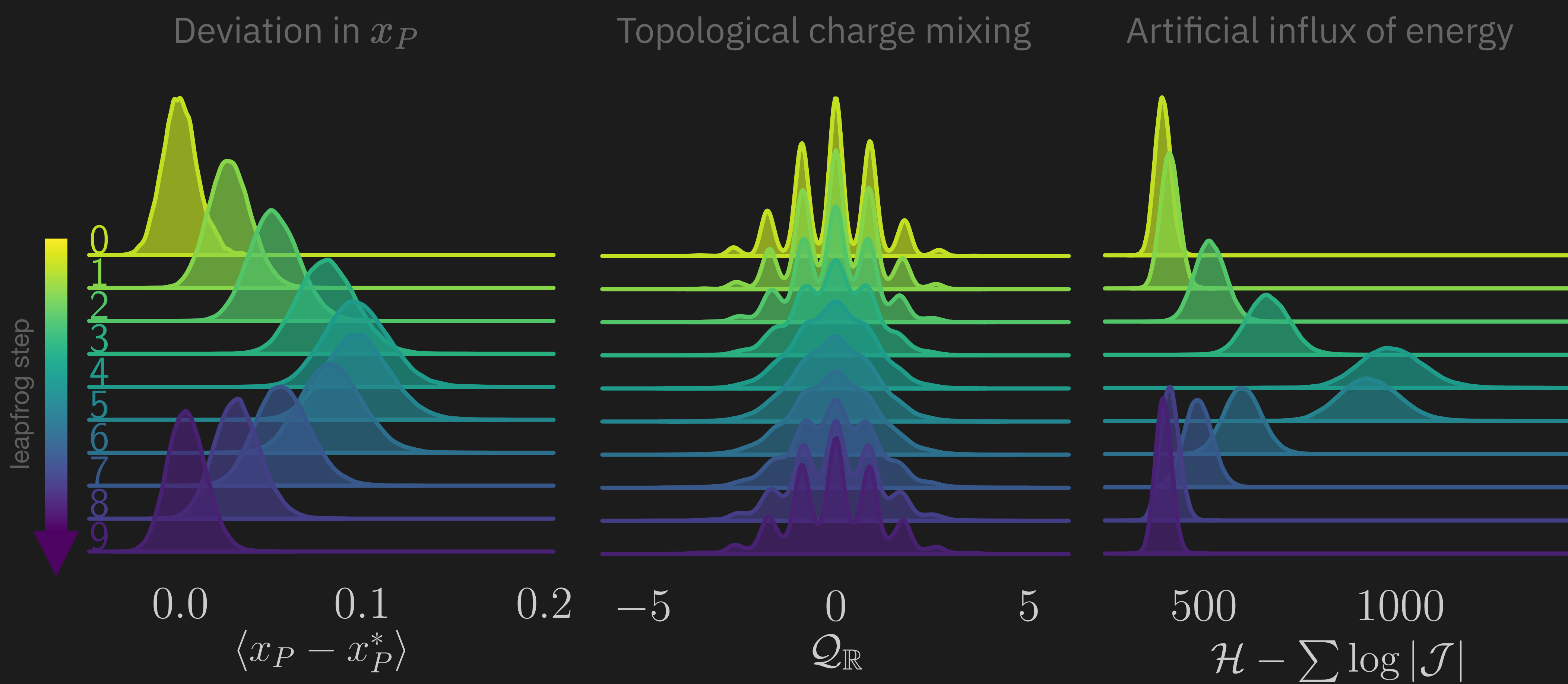
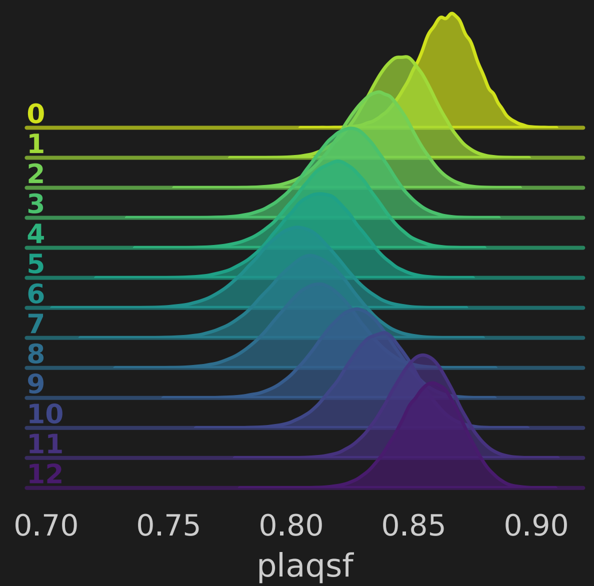
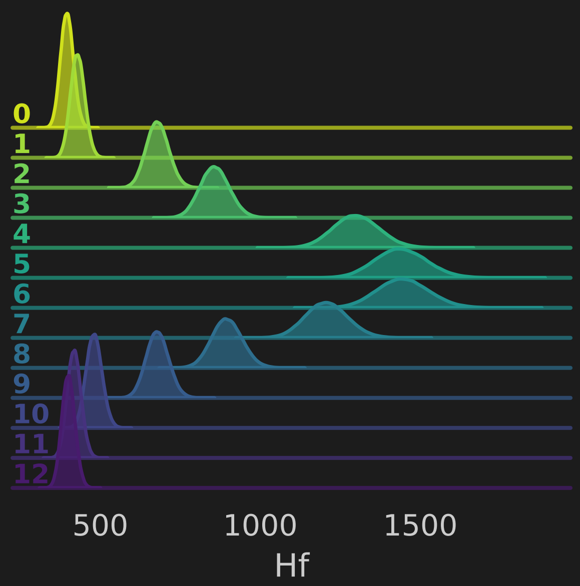


Figure 8: Illustration of how different observables evolve over a single L2HMC trajectory.

Interpretation



Average plaquette: $\langle x_P \rangle$ *vs LF step*



Average energy: $H - \sum \log |\mathcal{J}|$

Figure 9: The trained model artifically increases the energy towards the middle of the trajectory, allowing the sampler to tunnel between isolated sectors.

Plaquette analysis: x_P

Average $\langle x_P \rangle$, with x_P^* (dotted-lines)

Figure 11: Plot showing how **average plaquette**, $\langle x_P \rangle$ varies over a single trajectory for models trained at different β, with varying trajectory lengths N_{LF}

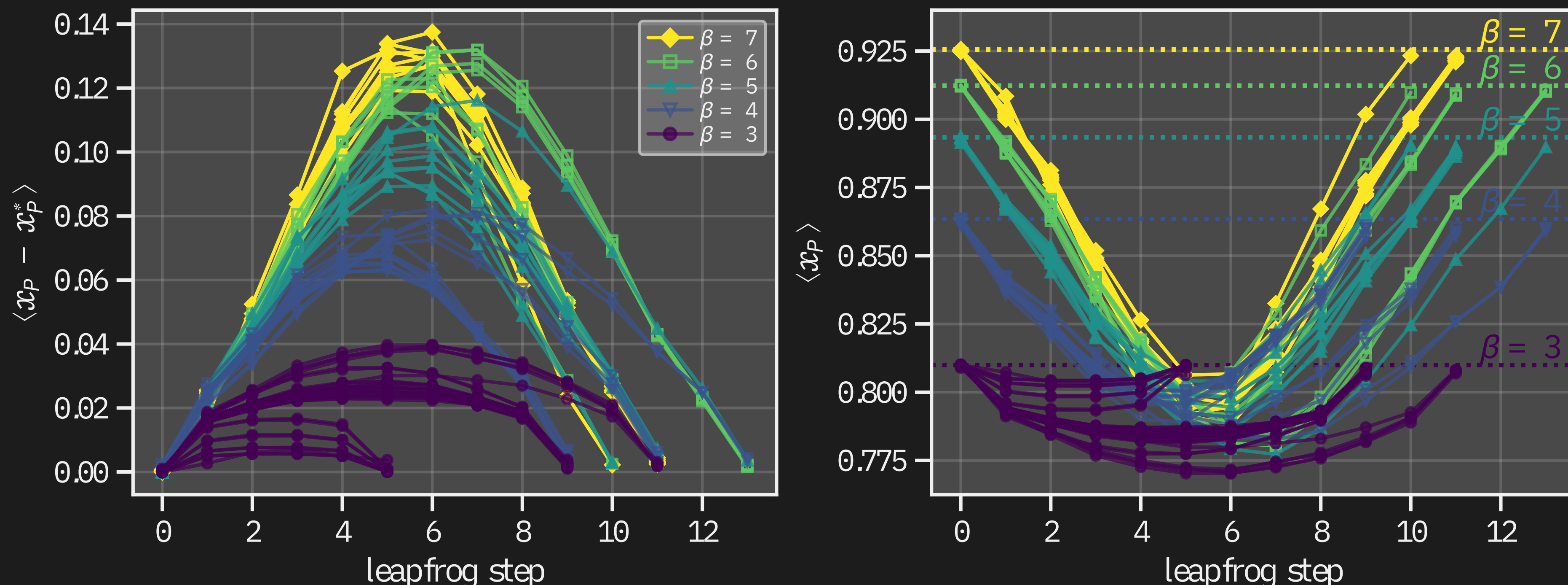
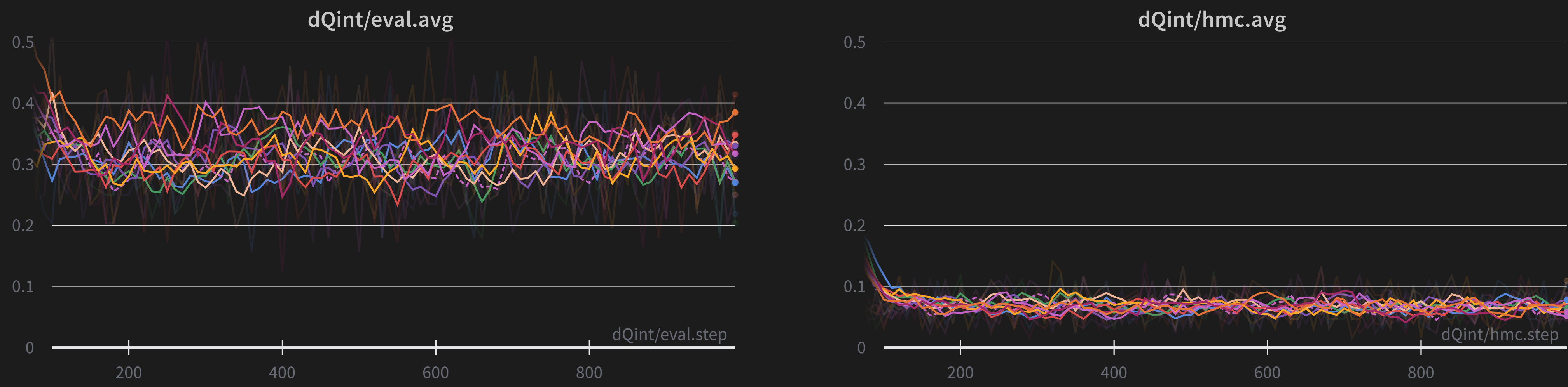


Figure 11: Plot showing how **average plaquette**, $\langle x_P \rangle$ varies over a single trajectory for models trained at different β, with varying trajectory lengths N_{LF}

Comparison

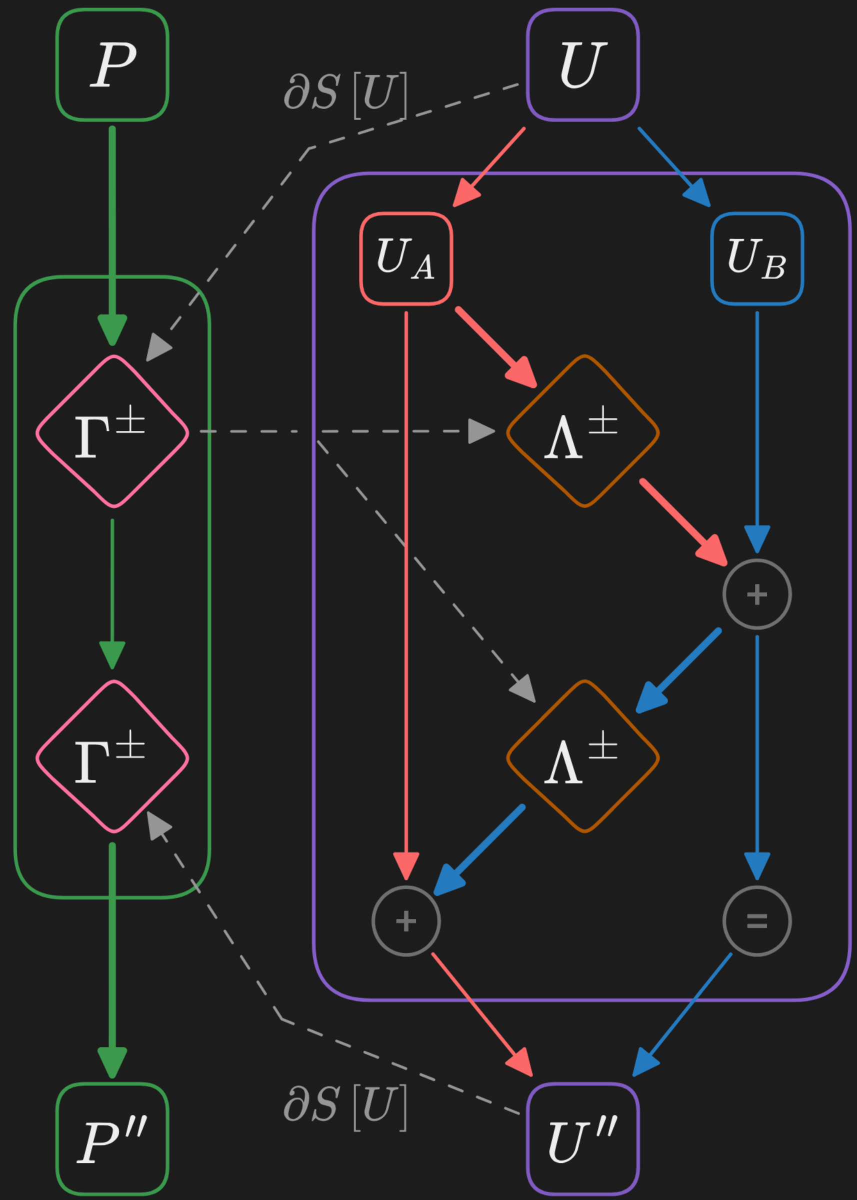


(a) Trained model *(b) Generic HMC*

Figure 12: Comparison of $\langle \delta Q \rangle = \frac{1}{N}\sum_{i=k}^{N} \delta Q_i$ for the trained model Figure 12 (a) vs. HMC Figure 12 (b)

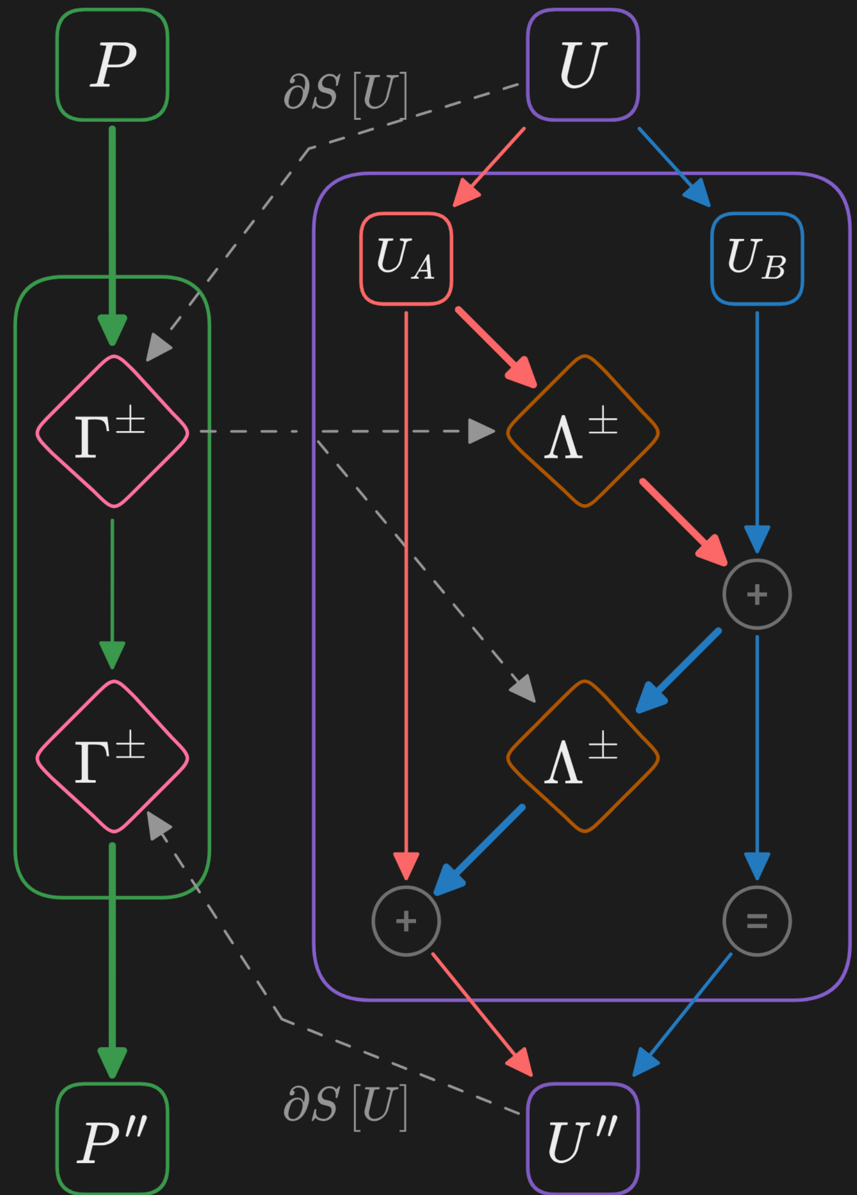
2D $U(1)$ Model

- `lattice.shape: [2, Nt, Nx]`
- maintain buffer of `Nb` chains, updated in parallel
  - `x.shape: [Nb, 2, Nt, Nx]`
- to deal with $x \in \mathbb{C}$, stack as: `[x.cos(), x.sin()]`
- final shapes:
  - `v.shape: [Nb, 2, Nt, Nx]`
  - `x.shape: [Nb, 2, 2, Nt, Nx]`



4D $SU(3)$ Model

- link variables:
  - $U_{\mu}(x) \in SU(3)$,
- + conjugate momenta:
  - $P_{\mu}(x) \in \mathfrak{su}(3)$
- We maintain a batch of `Nb` lattices, all updated in parallel
  - `lattice.shape`:
    - `[4, Nt, Nx, Ny, Nz, 3, 3]`
  - `batch.shape`:
    - `[Nb, *lattice.shape]`



4D $SU(3)$ Model

- link variables:
  - $U_{\mu}(x) \in SU(3)$,
- + conjugate momenta:
  - $P_{\mu}(x) \in \mathfrak{su}(3)$
- We maintain a batch of `Nb` lattices, all updated in parallel
  - `lattice.shape`:
    - `[4, Nt, Nx, Ny, Nz, 3, 3]`
  - `batch.shape`:
    - `[Nb, *lattice.shape]`

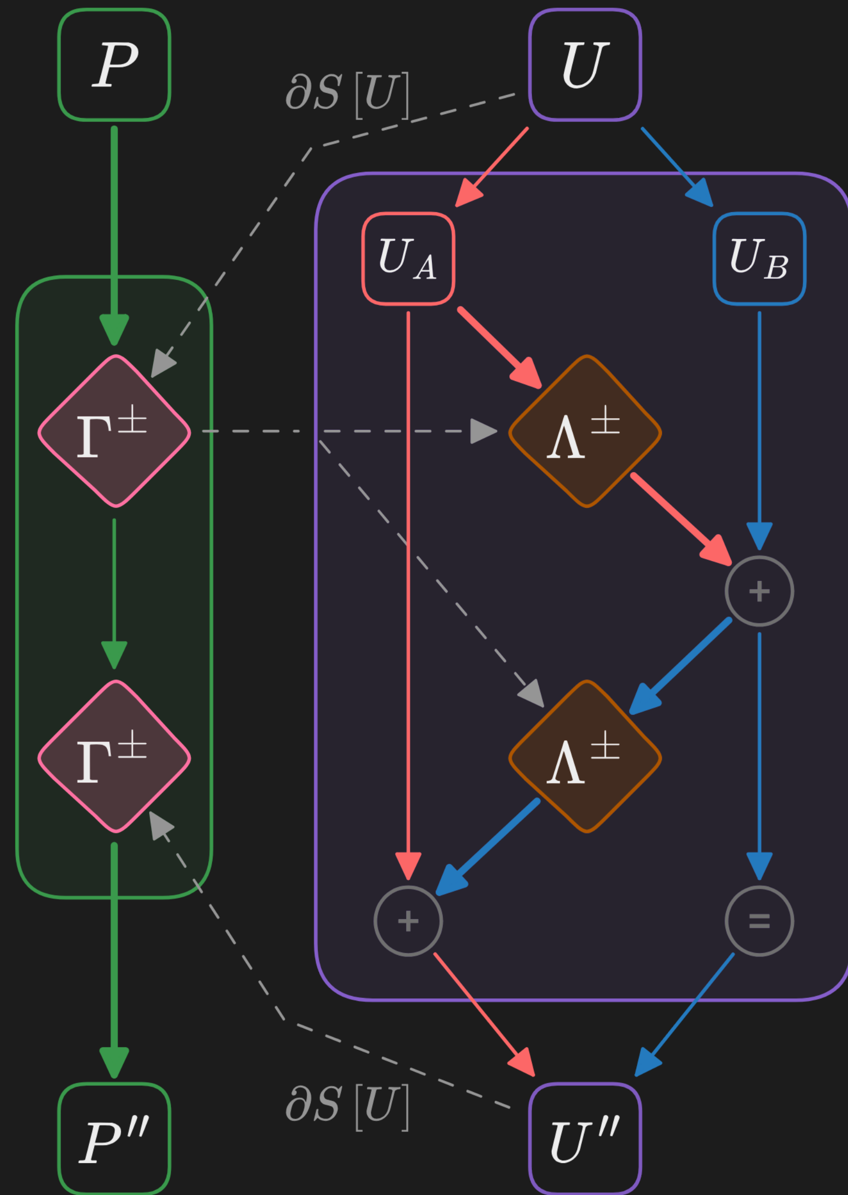


Figure 13: Jupyter Notebook

Links + References

- This talk: `saforem2/lqcd-pasc23`
- Code repo `saforem2/l2hmc-qcd`
- Slides saforem2.github.io/lqcd-pasc23
- Animated background
- Fermilab Muon g-2

Thank you!

samforeman.me @saforem2 foremans@anl.gov

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility,
which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

Acknowledgements

- Collaborators:
  - Xiao-Yong Jin
  - James C. Osborn
- References:
  - Link to HMC demo
  - Link to slides
    - link to github with slides
  - Link to github
  - reach out!
  - (Foreman et al. 2022)
  - (Foreman, Jin, and Osborn 2022)
  - (Boyda et al. 2022)
  - (Shanahan et al. 2022)
- Huge thank you to:
  - Yannick Meurice
  - Norman Christ
  - Akio Tomiya
  - Luchang Jin
  - Chulwoo Jung
  - Peter Boyle
  - Taku Izubuchi
  - ECP-CSD group
  - ALCF Staff + Datascience Group

References

Boyda, Denis et al. 2022. "Applications of Machine Learning to Lattice Quantum Field Theory." In *Snowmass 2021*. https://arxiv.org/abs/2202.05838.

Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth. 1987. "Hybrid Monte Carlo." *Phys. Lett. B* 195: 216–22. https://doi.org/10.1016/0370-2693(87)91197-X.

Foreman, Sam, Taku Izubuchi, Luchang Jin, Xiao-Yong Jin, James C. Osborn, and Akio Tomiya. 2022. "HMC with Normalizing Flows." *PoS* LATTICE2021: 073. https://doi.org/10.22323/1.396.0073.

Foreman, Sam, Xiao-Yong Jin, and James C. Osborn. 2022. "LeapfrogLayers: A Trainable Framework for Effective Topological Sampling." *PoS* LATTICE2021: 508. https://doi.org/10.22323/1.396.0508.

Shanahan, Phiala et al. 2022. "Snowmass 2021 Computational Frontier CompF03 Topical Group Report: Machine Learning," September. https://arxiv.org/abs/2209.07559.

Wittig, Hartmut. 2023. "Progress on $(g-2)_\mu$ from Lattice QCD." https://arxiv.org/abs/2306.04165.

Extras

Networks 2D $U(1)$

- Stack gauge links as `shape`(U_μ)`=[Nb, 2, Nt, Nx]` $\in \mathbb{C}$

$$x_\mu(n) \coloneqq [\cos(x), \sin(x)]$$

  with `shape`(x_μ)`= [Nb, 2, Nt, Nx, 2]` $\in \mathbb{R}$

- x-Network:
  - $\Lambda_k^{\pm}(x, v) \to \left[s_x^k, t_x^k, q_x^k\right]$
- v-Network:
  - $\Gamma_k^{\pm}(x, v) \to \left[s_v^k, t_v^k, q_v^k\right]$

v-update (pt. 1)

- network[1]: $(x, \partial_x S(x)) \coloneqq (x, F) \rightarrow (s_v, t_v, q_v)$, where

$$
\begin{aligned}
h_0 &= \sigma\left(w_x x + w_F F + b\right) \\
h_1 &= \sigma\left(w_1 h_0 + b_1\right) \\
&\ \ \vdots \\
h_n &= \sigma\left(w_n h_{n-1} + b_n\right) \longrightarrow
\end{aligned}
$$

$$
s_v = \lambda_s \tanh\left(w_s h_n + b_s\right), \quad t_v = w_t z + b_t, \quad q_v = \lambda_q w_q z + b_q
$$

1. $\sigma(\cdot)$ denotes an activation function

v-update (pt. 2)

- Network outputs[1]:

$$s_v = \lambda_s \tanh\left(w_s\, h_n + b_s\right), \quad t_v = w_t h_n + b_t, \quad q_v = \lambda_q \tanh\left(w_q h_n + b_q\right)$$

- Use these to update $\Gamma^{\pm} : (x, v) \to (x, v_{\pm})$[2]:
  - forward $(d = +)$:

$$\Gamma^{+}(x, v) := v_{+} = v \cdot e^{\frac{\varepsilon}{2} s_v} - \frac{\varepsilon}{2}\left[F \cdot e^{\varepsilon q_v} + t_v\right]$$

  - backward $(d = -)$:

$$\Gamma^{-}(x, v) := v_{-} = e^{-\frac{\varepsilon}{2} s_v}\left\{v + \frac{\varepsilon}{2}\left[F \cdot e^{\varepsilon q_v} + t_v\right]\right\}$$

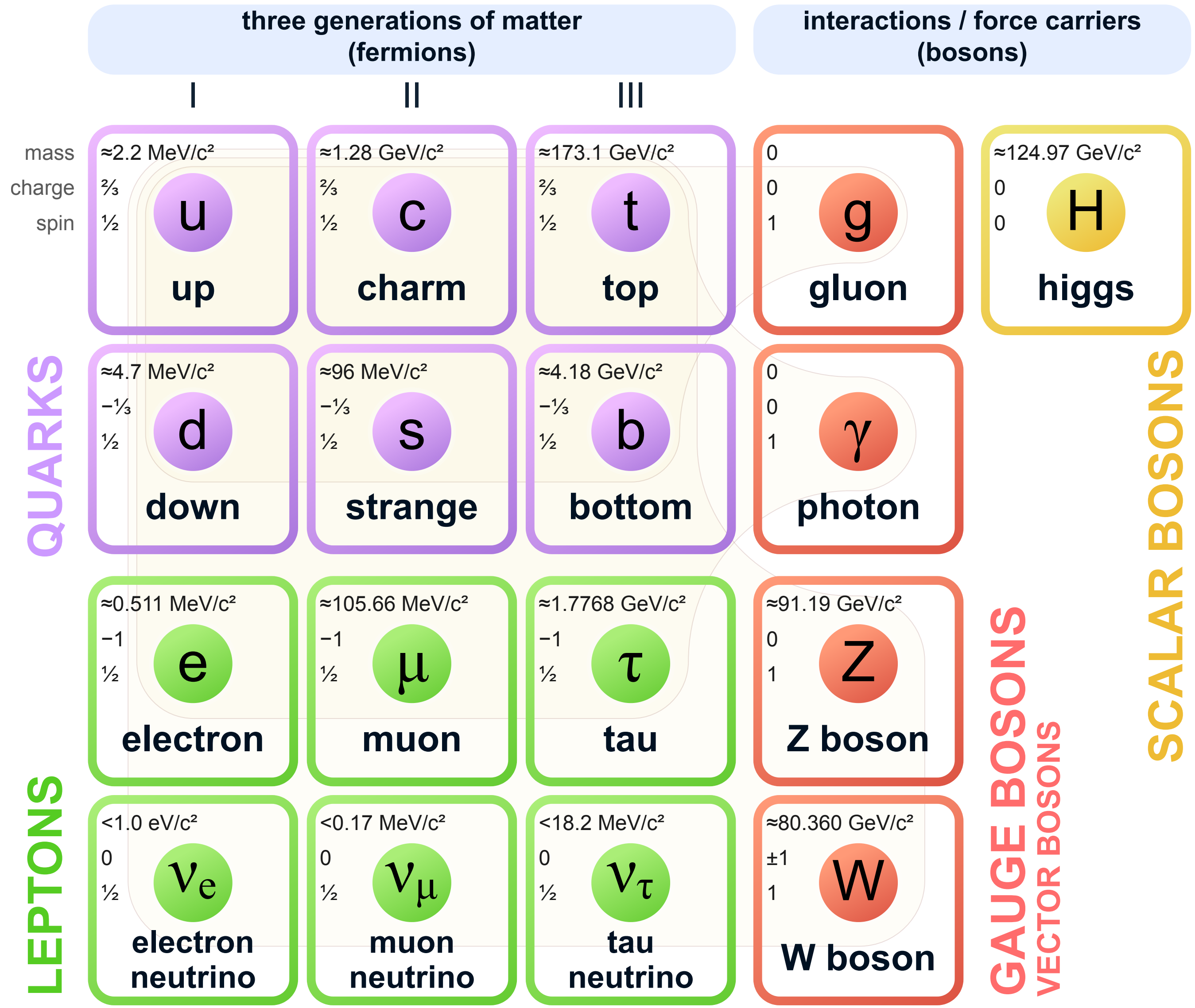
1. $\lambda_s,\ \lambda_q \in \mathbb{R}$ are trainable parameters
2. Note that $(\Gamma^{+})^{-1} = \Gamma^{-}$, i.e. $\Gamma^{+}\left[\Gamma^{-}(x, v)\right] = \Gamma^{-}\left[\Gamma^{+}(x, v)\right] = (x, v)$

LQCD @ ALCF (2008)

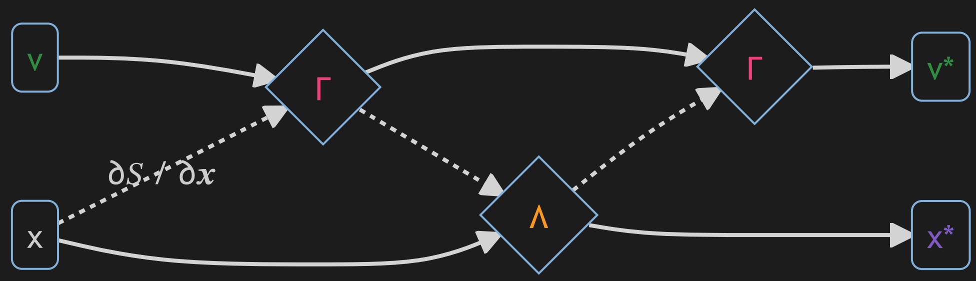
> *The Blue Gene/P at the ALCF has tremendously accelerated the generation of the gauge configurations—in many cases, by a factor of 5 to 10 over what has been possible with other machines. Significant progress has been made in simulations with two different implementations of the quarks—domain wall and staggered*[1]
>
> *— Mike Papka, 2008*

1. Argonne Leadership Computing Facility • 2008 Annual Report

Standard Model of Elementary Particles



HMC



Python

▶ Code

```
0.5188124755961172
```

Python

For a demonstration of a line plot on a polar axis, see Figure 14

```python
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({
    'axes.facecolor': 'none',
    'figure.facecolor': 'none',
    'savefig.facecolor': 'none',
    'savefig.format': 'svg',
    'axes.edgecolor': 'none',
    'axes.grid': True,
    'axes.labelcolor': '#666',
    'axes.titlecolor': '#666',
    'grid.color': '#666',
    'text.color': '#666',
    'grid.linestyle': '--',
    'grid.linewidth': 0.5,
    'grid.alpha': 0.4,
    'xtick.color': 'none',
```

Python

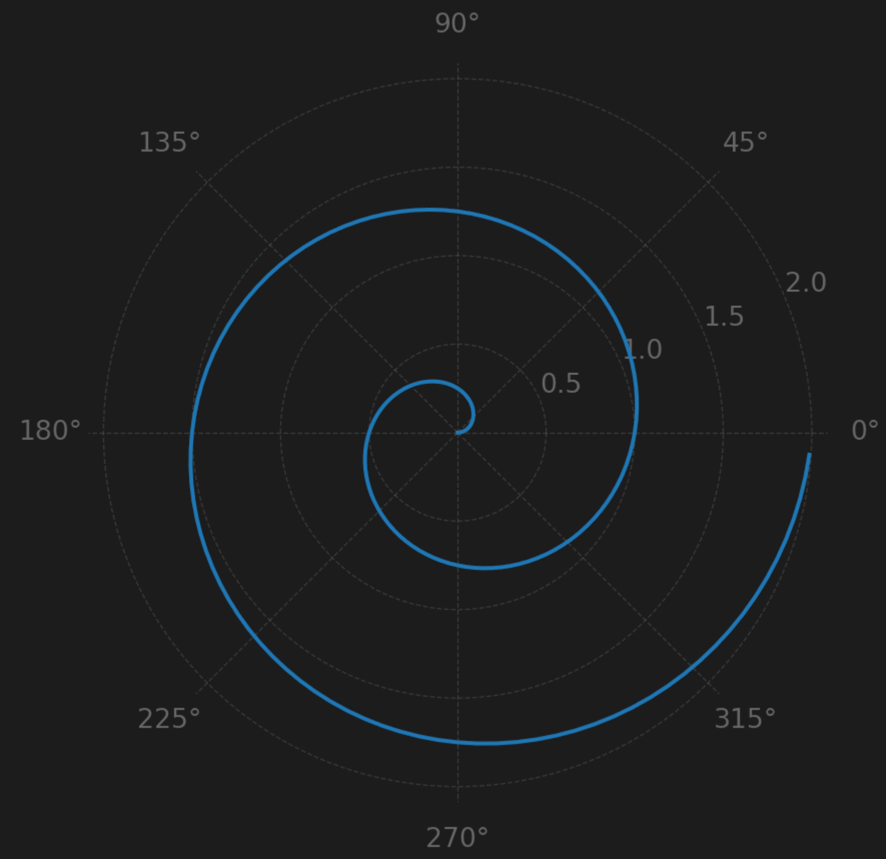
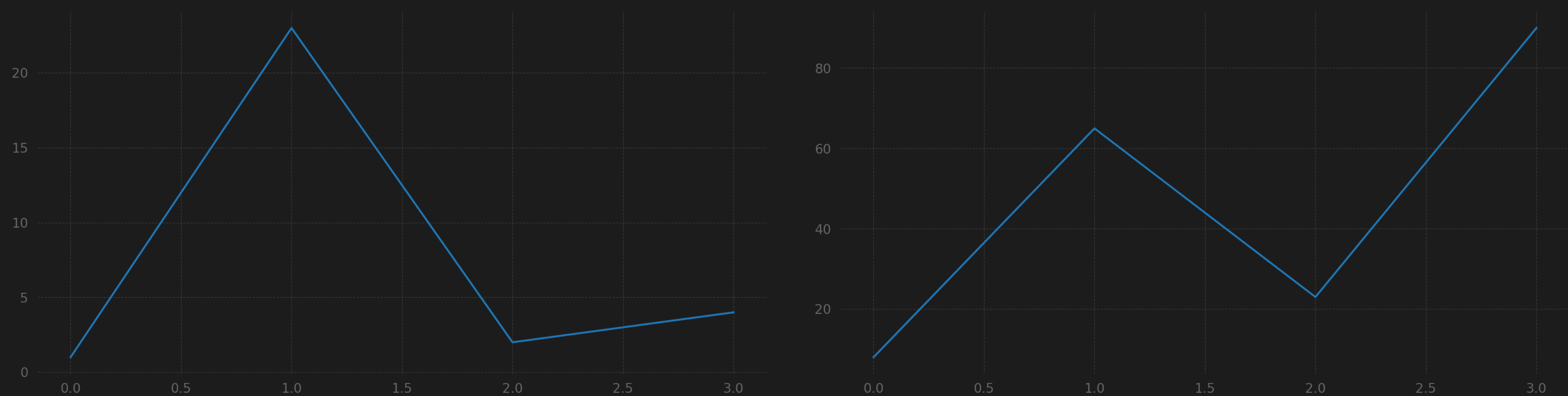


Figure 14: A line plot on a polar axis

Figures with Subcaptions

```python
import matplotlib.pyplot as plt
plt.plot([1,23,2,4])
plt.show()

plt.plot([8,65,23,90])
plt.show()
```



(a) First *(b) Second*

Figure 15: Charts

Testing Callouts

Note

Testing note callouts with `default` appearance

Tip

Testing tip callout with `default` appearance

Caution

Testing tip callout with `default` appearance

Warning

Testing warning callout with `default` appearance

Important

Testing important callout with `default` appearance