HAMILTONIAN-GUIDED DIFFUSION FIELDS FOR VARIABLE-LENGTH RIGID-ARM TRAJECTORY GENERATION

Guorui Sang, Pedram Rooshenas
Department of Computer Science
University of Illinois Chicago
Chicago, IL 60607, USA
{gsang,pedram}@uic.edu

ABSTRACT

Generating rigid-arm trajectories is fundamental for planning, imitation learning, and proposal generation. In many deployment settings, a simulator may be unavailable, so trajectory generation must rely on offline data while remaining faithful to rigid-body dynamics. We use diffusion probabilistic fields (DPFs) to model trajectories as continuous-time fields, enabling variable-length generation by querying arbitrary time coordinates. Purely data-driven DPF samples can violate Hamiltonian structure, particularly when trajectories must satisfy Hamilton's equations under actuation and remain energy-consistent in unforced regimes. We therefore learn a system Hamiltonian with a Hamiltonian neural network (HNN) and use it to define a physics inconsistency residual over entire trajectories. We incorporate two guidance mechanisms that condition DPF sampling on this residual: a one-step, gradient-based guidance applied during denoising, and a multi-sample self-normalized importance sampling scheme that reweights candidate trajectories toward physics-consistent outcomes. On 2-DoF and 3-DoF MuJoCo arms across diverse torque policies, Hamiltonian-guided DPF sampling reduces error relative to simulator rollouts and improves generalization to unseen trajectory lengths and torque policies, with importance sampling providing stronger consistency at increased sampling cost.

1 INTRODUCTION

Generating rigid-arm trajectories is essential in robotics, with applications in simulation (Makoviychuk et al., 2021), motion planning (Janner et al., 2022), and imitation learning (Chi et al., 2023). For non-dissipative rigid-body systems, the dynamics are governed by Hamilton's equations, and physically consistent trajectories—those that satisfy these governing equations—are critical for downstream tasks. However, high-fidelity simulation can be computationally expensive, and in many settings only offline trajectory data are available. Deep generative models provide a data-driven alternative by learning trajectory distributions directly from such data. Given sufficient data, these models can produce trajectories at inference time without access to a physics simulator.

Diffusion models (Ho et al., 2020; Song et al., 2020) have become popular for generating robotic trajectories (Wolf et al., 2025; Urain et al., 2024; Janner et al., 2022; Chi et al., 2023; Serifi et al., 2024). Most of these works generate trajectories for planning or control (typically state–action sequences), without explicitly enforcing physical consistency of the underlying dynamics. Moreover, many diffusion-based trajectory models are trained and sampled with a fixed trajectory length (Chi et al., 2023; Carvalho et al., 2023), making it difficult to generalize to substantially different lengths without retraining. A natural alternative is autoregressive generation, but it suffers from error accumulation (Ross et al., 2011): small deviations compound over time, causing trajectories to drift and violate physical constraints. Diffusion models generate entire trajectories simultaneously (Janner et al., 2022), avoiding compounding errors. To enable variable-length generation while avoiding accumulated error, we adopt diffusion probabilistic fields (DPF; Zhuang et al., 2023). With a Perceiver IO (Jaegle et al., 2021) architecture, DPF models trajectories as continuous-time fields rather

than fixed-length sequences. By querying the field at different sets of time points, DPF can generate trajectories of variable lengths.

Although DPF can generate variable-length trajectories, it learns purely from data without encoding physical laws, and thus the generated trajectories may violate the governing dynamics. Several works (Ajay et al., 2023; Janner et al., 2022; Saha et al., 2024; Carvalho et al., 2023) incorporate task-specific objectives such as reward, cost, or collision avoidance into diffusion sampling via classifier guidance (CG; Dhariwal & Nichol, 2021), which uses gradients from a separately trained function to steer the sampling process, and classifier-free guidance (CFG; Ho & Salimans, 2021). While these approaches steer generation toward desired task outcomes, they do not enforce consistency with the underlying governing dynamics. A complementary line of work learns physical structure directly from data. Neural ODEs (Chen et al., 2018) model continuous-time dynamics, while Hamiltonian Neural Networks (HNNs; Greydanus et al., 2019) and Lagrangian Neural Networks (LNNs; Cranmer et al., 2020) explicitly encode conservation laws by parameterizing the Hamiltonian or Lagrangian. Although these models can roll out trajectories, they suffer from error accumulation and provide limited control over the resulting trajectory distribution.

In this work, we combine the strengths of both approaches. A DPF models the data distribution and generates variable-length trajectories, while an HNN-derived energy function measures deviations from Hamilton's equations and is minimized during diffusion sampling, steering trajectories toward physically consistent solutions. Our contributions are as follows:

1. We propose a framework that decouples variable-length rigid-arm trajectory generation into data-driven distribution learning via diffusion fields and physics enforcement via HNN-based energy guidance during sampling. By randomizing query and context pairs during training, the DPF generates trajectories at unseen lengths without retraining.

2. We introduce a Hamiltonian-based energy function derived from a trained HNN to measure deviations from Hamiltonian dynamics, and minimize it via guidance during diffusion sampling to improve the physical consistency of generated trajectories.

3. We demonstrate that Hamiltonian guidance consistently reduces trajectory error and improves physical consistency across varying lengths and diverse torque policies on 2-DoF and 3-DoF rigid-arm systems.

2 Related Work

Diffusion Models for Trajectory Generation. Diffusion models generate entire trajectories at once, avoiding error accumulation over time. Gradient-based guidance is often applied during sampling to make generated trajectories satisfy certain constraints. Janner et al. (2022) model the joint distribution of entire trajectories, converting planning into conditional generation. They use CG for reward maximization at sampling time. Carvalho et al. (2023) learn from expert demonstrations to generate feasible and smooth trajectories between start and goal configurations. They also use CG to steer sampling toward low-cost regions. EDMP (Saha et al., 2024) trains on kinematically valid trajectories and employs an ensemble of complementary cost functions to guide sampling. This approach robustly avoids collisions across diverse scenes without retraining. Decision Diffuser (Ajay et al., 2023) instead uses CFG for return-conditioning, eliminating the need for a separately trained reward function on noisy inputs.

Beyond gradient-based guidance, researchers have developed methods that do not require gradients of constraint functions. D-MPC (Zhou et al., 2024a) uses a factorized representation with two multi-step diffusion models: an action proposal model and a dynamics model. These are combined via sampling-based MPC planning for offline RL locomotion control, avoiding compounding errors through joint trajectory-level prediction. T-Diff (Yu et al., 2024) generates future waypoint sequences for Object Goal Navigation conditioned on semantic maps and target objects. Pasricha et al. (2025) propose a payload-conditioned diffusion model that generates dynamically feasible joint-space trajectories (position, velocity, and acceleration) by learning from dynamics-validated training data. Diffusion Policy (Chi et al., 2023) learns visuomotor policies from demonstrations, generating action sequences conditioned on visual observations without guidance. JM2D (Jung et al., 2025) jointly generates robot trajectories and safety correction parameters from shared noise. It uses importance-weighted Monte Carlo sampling to steer generation toward pairs where the tra-

jectory is correctable and the correction is minimal, without requiring gradients of the constraint function.

Guidance Methods for Diffusion Models. Various guidance methods have been developed to improve controllability in diffusion models. Classifier guidance (Dhariwal & Nichol, 2021) uses a classifier trained on noisy inputs to steer generation toward a target class. Classifier-free guidance (Ho & Salimans, 2021) achieves similar controllability by combining conditional and unconditional score estimates, eliminating the need for a separate classifier. There are also methods from the perspective of energy-based models. Steered Diffusion (Nair et al., 2023) applies energy-based guidance to the implicit clean image prediction instead of the noisy latent. This allows steering unconditional diffusion models without retraining on noisy images. Liu et al. (2022) interpret diffusion models as energy-based models and summing their score functions to compose multiple concepts at inference time without retraining, which enables compositional image generation. EDIS (Liu et al., 2024) uses three energy functions learned via contrastive prediction to guide a diffusion model to generate samples matching the online distribution, avoiding distribution shift.

Physics-Informed Deep Learning. To make neural networks capture underlying physics instead of specific data patterns, researchers have proposed various physics-informed learning methods. Hamiltonian Neural Networks (HNNs; Greydanus et al., 2019) model the dynamics of non-dissipative systems by learning the Hamiltonian function. Port-Hamiltonian Neural Networks (Desai et al., 2021) model non-autonomous dynamical systems with energy dissipation and external forcing. They decompose the dynamics into three learnable components: a stationary Hamiltonian, a time-dependent force, and a damping coefficient. Hamiltonian generative networks (Toth et al., 2020) learn a Hamiltonian from data and generate trajectories by rolling out states with a symplectic integrator applied to Hamilton's equations, enabling long-horizon forward/backward prediction and sample generation. Lagrangian Neural Networks (Cranmer et al., 2020) parameterize arbitrary Lagrangians with neural networks and derive dynamics via the differentiable Euler-Lagrange equation. This enables energy-conserving dynamics learning from non-canonical coordinates, a setting where HNNs struggle.

Physics-based Diffusion Models. PhysDiff (Yuan et al., 2023) uses a physics simulator to project denoised samples onto physically plausible states at each diffusion step, improving physical realism in human motion generation. Bastek et al. (2024) add a PDE-based loss term during training to enforce physical constraints on generated samples, reducing residual errors by up to two orders of magnitude. Zhou et al. (2024b) embed energy and momentum conservation laws as priors into the diffusion training objective.

3 METHOD

Given query times $\mathbf{T}_q = \{t_i\}_{i=1}^{N_q}$ and control torques $\boldsymbol{\tau}_q = \{\tau(t_i)\}_{i=1}^{N_q}$, we model rigid-arm trajectory states $\mathbf{S} = \big[(\mathbf{p}_{t_i}, \mathbf{q}_{t_i})\big]_{i=1}^{N_q}$ with momentum $\mathbf{p}_t$ and position $\mathbf{q}_t$. Our base generator defines $p_\theta(S \mid T_q, \tau_q, \mathcal{C})$, where $\mathcal{C}$ denotes optional observed context (e.g., an initial prefix). To enforce physical consistency at inference time without calling a simulator, we define a tilted (energy-guided) distribution

$$p_{\text{guided}}(\mathbf{S} \mid \mathbf{T}_q, \boldsymbol{\tau}_q, \mathcal{C}) \propto p_\theta(\mathbf{S} \mid \mathbf{T}_q, \boldsymbol{\tau}_q, \mathcal{C}) \exp\big(-\lambda\, E_\psi(\mathbf{S}; \boldsymbol{\tau}_q)\big), \tag{1}$$

where E_ψ is a differentiable physics-inconsistency energy computed from a learned Hamiltonian model and λ controls guidance strength.

p_θ is represented by a diffusion probabilistic field (DPF; Zhuang et al., 2023) that models trajectories as continuous-time fields and supports variable-length generation by querying arbitrary time coordinates. The energy E is derived from a Hamiltonian neural network (HNN;Greydanus et al., 2019) that learns a Hamiltonian $H_\psi(\mathbf{p}, \mathbf{q})$ and measures violations of Hamilton's equations under the applied torques. We approximately sample from p_{guided} using (i) gradient-based guidance during denoising (Nair et al., 2023) and (ii) self-normalized importance sampling over candidate trajectories (Grover et al., 2019).

3.1 DIFFUSION PROBABILISTIC FIELDS FOR POLICY-CONDITIONED TRAJECTORIES.

To support flexible-length trajectories, we adopt diffusion probabilistic fields (DPFs; Zhuang et al., 2023), which define a diffusion model over unordered sets of field evaluations and thereby represent distributions over continuous functions. We condition trajectories on a torque policy π_τ, which specifies the control input as a function of time, $\pi_\tau : \mathbb{R} \to \mathbb{R}^{d_\tau}$, $\tau(t) = \pi_\tau(t)$, where d_τ is the torque dimension, corresponding to the number of actuation channels, which is typically the number of actuated degrees of freedom of the rigid arm.

We view a rigid-arm trajectory as a policy-conditioned field $f_{\pi_\tau} : \mathbb{R} \to \mathbb{R}^{2d}$ that maps time to state,

$$f_{\pi_\tau}(t) = \big(\mathbf{p}(t), \mathbf{q}(t)\big), \tag{2}$$

where $\mathbf{p}(t)$ and $\mathbf{q}(t)$ denote momentum and position at time t, respectively. Given query times $\mathbf{T}_q = \{t_i\}_{i=1}^{N_q}$, we evaluate the policy to obtain torques $\boldsymbol{\tau}_q = \{\pi_\tau(t_i)\}_{i=1}^{N_q}$ and represent the field by the set of value and coordinate tuples

$$\big\{(t_i, \pi_\tau(t_i), \mathbf{s}_{t_i})\big\}_{i=1}^{N_q}, \qquad \mathbf{s}_{t_i} = f_{\pi_\tau}(t_i) = \big(\mathbf{p}_{t_i}, \mathbf{q}_{t_i}\big). \tag{3}$$

For convenience, we also write the corresponding ordered trajectory tensor as $\mathbf{S} = \big[(\mathbf{p}_{t_i}, \mathbf{q}_{t_i})\big]_{i=1}^{N_q}$. DPF learns a permutation-invariant conditional distribution over the queried states,

$$p_\theta(\mathbf{S} \mid \mathbf{T}_q, \boldsymbol{\tau}_q, \mathcal{C}), \tag{4}$$

where $\mathcal{C}$ denotes optional observed context (e.g., an initial prefix), and can be evaluated on arbitrary continuous query sets $\mathbf{T}_q$ with varying cardinality N_q, enabling generation at variable horizons and temporal resolutions.

Figure 1 illustrates the DPF architecture. At each denoising iteration, the model selects a subset of query pairs as context, in addition to any conditional pairs such as an observed prefix or initial states. The encoder maps the context pairs into a latent array through multiple attention layers. The decoder is conditioned on the torque sequence and uses the query pairs together with the latent array to predict the noise at each denoising step. To condition on the causal structure of dynamics, we modulate query features using AdaLN (Peebles & Xie, 2023) at t_{i+1} using an embedding derived from $(\mathbf{s}_{t_i}, \boldsymbol{\tau}_{t_i})$, where $\mathbf{s}_{t_i}$ is the noisy state at timestep t_i extracted from the query pairs at the current diffusion step. At high noise levels (early diffusion steps), the signal-to-noise ratio is low, rendering the causal conditioning less effective. As denoising progresses, the states become increasingly accurate and the conditioning provides meaningful transition structure. As illustrated in Figure 1, $\mathbf{s}_{t_{i-1}}$ and $\boldsymbol{\tau}_{t_{i-1}}$ $(i = 2, \ldots, N_q)$ are first embedded jointly into $\mathbf{h}_{t_i}$, so that the embedding at time t_{i-1} acts on features at time t_i. For the embedding at the first timestep, we set $\mathbf{h}_1$ to learned parameters. After this, we use the embedding to modulate the features via AdaLN:

$$(\boldsymbol{\gamma}_{t_i}, \boldsymbol{\beta}_{t_i}) = \mathrm{MLP}(\mathbf{h}_{t_i}), \quad \tilde{\mathbf{z}}_{t_i} = \mathbf{z}_{t_i} \odot (1 + \boldsymbol{\gamma}_{t_i}) + \boldsymbol{\beta}_{t_i}, \tag{5}$$

where $\mathbf{z}_{t_i}$ denotes the query features and $\odot$ is element-wise multiplication. **Training.** To train the model on variable-length trajectories, we sample N_q from a discrete uniform distribution over $\{100, 200, \ldots, 1000\}$, and sample N_c from another uniform distribution $\mathcal{U}\{1, 2, \ldots, N_q - 1\}$.

During the forward diffusion process, since time coordinates and control torques are fixed inputs, noise is added only to the state components:

$$\mathbf{C}_k = [\sqrt{\bar{\alpha}_k}\mathbf{S}_{(c,0)} + \sqrt{1 - \bar{\alpha}_k}\boldsymbol{\epsilon}_{(c,k)}, \mathbf{T}_c], \tag{6}$$

$$\mathbf{Q}_k = [\sqrt{\bar{\alpha}_k}\mathbf{S}_{(q,0)} + \sqrt{1 - \bar{\alpha}_k}\boldsymbol{\epsilon}_{(q,k)}, \mathbf{T}_q], \tag{7}$$

where $\mathbf{S}_{(q,0)}$, $\mathbf{S}_{(c,0)}$ are sets of query and context points from data manifold, respectively, $k = 1, \ldots, K$, $\boldsymbol{\epsilon}_{(c,k)}, \boldsymbol{\epsilon}_{(q,k)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\bar{\alpha}_k = \prod_{s=1}^{k}(1 - \beta_s)$ with noise schedule $\{\beta_s\}$.

The DPF is trained to predict $\boldsymbol{\epsilon}_{(q,k)}$ to satisfy the following objective:

$$\mathcal{L}_{\text{DPF}} = \mathbb{E}_{k,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon}_{(q,k)} - \boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q)\|_2^2\right], \tag{8}$$

where $\boldsymbol{\tau}_q = \boldsymbol{\tau}_{t_1:t_{N_q}}$. Note that the entire generative process is conditioned on the sequence of control torques.

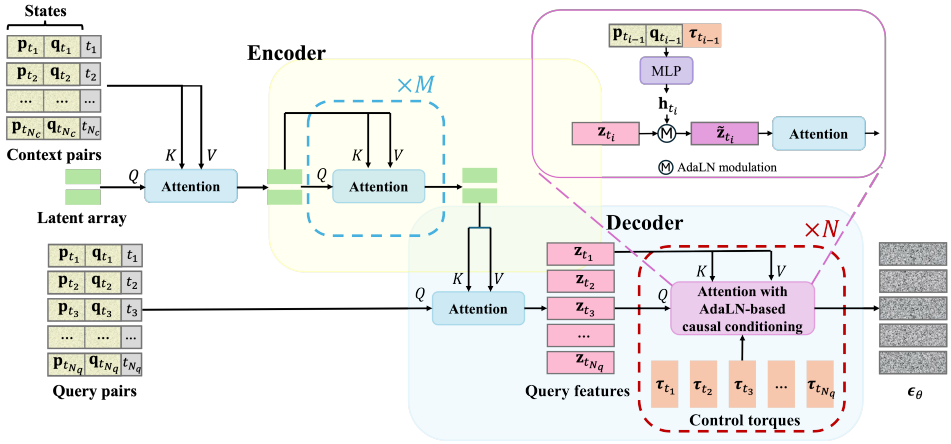


Figure 1: Architecture of DPF. The first N_c query pairs are used as context. The encoder maps context pairs to a latent array, and the decoder lets query tokens attend to this latent array to predict noise ϵ_θ for denoising the state components. Features are modulated by AdaLN-based causal conditioning.

3.2 Hamiltonian Neural Network for Physical Consistency

To enforce physical consistency of the generated trajectories, we guide the sampling using an energy function derived from an HNN. The HNN captures system dynamics by learning the Hamiltonian function. For a rigid arm without energy dissipation (no friction or damping), Hamilton's equations are:

$$\dot{\mathbf{q}}_t = \frac{\partial H}{\partial \mathbf{p}_t}, \qquad \dot{\mathbf{p}}_t = -\frac{\partial H}{\partial \mathbf{q}_t} + \boldsymbol{\tau}_t, \tag{9}$$

where $H(\mathbf{p}_t, \mathbf{q}_t)$ is the Hamiltonian. Note that with external torque $\boldsymbol{\tau}_t \neq 0$, the system is non-autonomous and the Hamiltonian H is not conserved, i.e., energy is injected by the applied torques. The HNN learns the functional form of H, not its conservation.

HNN Training. We train the HNN on trajectories generated by MuJoCo (Todorov et al., 2012). The training objective is:

$$\mathcal{L}_{\text{HNN}} = \sum_t \left(\left\| \frac{\partial H_\psi}{\partial \mathbf{p}_t} - \dot{\mathbf{q}}_t \right\|_2^2 + \left\| -\frac{\partial H_\psi}{\partial \mathbf{q}_t} + \boldsymbol{\tau}_t - \dot{\mathbf{p}}_t \right\|_2^2 \right), \tag{10}$$

where we abbreviate $H_\psi(\mathbf{p}_t, \mathbf{q}_t)$ as H_ψ for clarity. The training objective requires the time derivatives $\dot{\mathbf{q}}_t$ and $\dot{\mathbf{p}}_t$, which we approximate by central difference.

Energy function. During sampling, we apply gradient descent to steer trajectories toward physically consistent regions. We define a one-step Hamiltonian prediction energy:

$$E\big(\mathbf{S}_{(q,0)}\big) = \sum_{t_i} \left(\left\| \mathbf{q}_{t_{i+1}} - \widetilde{\mathbf{q}}_{t_{i+1}} \right\|_2^2 + \left\| \mathbf{p}_{t_{i+1}} - \widetilde{\mathbf{p}}_{t_{i+1}} \right\|_2^2 \right), \tag{11}$$

where

$$\widetilde{\mathbf{q}}_{t_{i+1}} = \mathbf{q}_{t_i} + \Delta t \, \frac{\partial H_\psi}{\partial \mathbf{p}_{t_i}}, \qquad \widetilde{\mathbf{p}}_{t_{i+1}} = \mathbf{p}_{t_i} + \Delta t \left(\boldsymbol{\tau}_{t_i} - \frac{\partial H_\psi}{\partial \mathbf{q}_{t_i}} \right). \tag{12}$$

Both terms enforce consistency with Hamiltonian dynamics by matching each generated next state to the HNN one-step prediction under the same torque input. This approach stands in contrast to HNN rollout (Toth et al., 2020), and it permits parallel computation. Training of DPF and HNN is outlined in Algorithm 1.

3.3 Sampling with Inference-Time Guidance.

We sample trajectories using DDIM (Song et al., 2020) while steering the reverse process toward low-energy solutions. Starting from $\mathbf{S}_{(q,K)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we iterate for $k = K, \ldots, 1$. At step k,

Algorithm 1 Training

Input: Trajectory $\mathbf{s}_{t_1:t_T}$, torques $\boldsymbol{\tau}_{t_1:t_T}$, learning rate λ_θ, λ_ψ
Output: DPF parameters θ, HNN parameters ψ
Stage I: Train DPF
repeat
  sample $N_q \sim \mathcal{U}\{100, 200, \ldots, 1000\}$, $N_c \sim \mathcal{U}\{1, \ldots, N_q - 1\}$
  Form $\mathbf{Q}_0 = [\mathbf{S}_{(q,0)}, \mathbf{T}_q]$ from first N_q timesteps; set control torques $\boldsymbol{\tau}_q = \boldsymbol{\tau}_{t_1:t_{N_q}}$
  Set $\mathbf{C}_0 = [\mathbf{S}_{(c,0)}, \mathbf{T}_c]$ as first N_c query pairs
  Sample $k \sim \text{Uniform}\{1, \ldots, K\}$, $\boldsymbol{\epsilon}_{(c,k)}, \boldsymbol{\epsilon}_{(q,k)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{C}_k \leftarrow [\sqrt{\bar{\alpha}_k}\mathbf{S}_{(c,0)} + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}_{(c,k)}, \mathbf{T}_c]$
  $\mathbf{Q}_k \leftarrow [\sqrt{\bar{\alpha}_k}\mathbf{S}_{(q,0)} + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}_{(q,k)}, \mathbf{T}_q]$
  $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \|\boldsymbol{\epsilon}_{(q,k)} - \boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q)\|_2^2$
until converged
Stage II: Train HNN
repeat
  Compute $\dot{\mathbf{q}}_{t_i}, \dot{\mathbf{p}}_{t_i}$ via central difference
  Compute $\mathcal{L}_{\text{HNN}}$ via equation 10
  $\psi \leftarrow \psi - \lambda_\psi \nabla_\psi \mathcal{L}_{\text{HNN}}$
until converged

we form the context set $\mathbf{C}_k$ from the current query set $\mathbf{Q}_k$ (e.g., the randomly sample N_c pairs, together with any observed conditional pairs such as an initial state or prefix), and predict noise $\boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q)$. We then compute the implied clean trajectory estimate

$$\hat{\mathbf{S}}_{(q,0)} = \frac{1}{\sqrt{\bar{\alpha}_k}} \left(\mathbf{S}_{(q,k)} - \sqrt{1-\bar{\alpha}_k} \cdot \boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q) \right), \tag{13}$$

and apply guidance on this clean estimate before updating the latent.

Gradient guidance. We take a single gradient step on the clean estimate,

$$\tilde{\mathbf{S}}_{(q,0)} = \hat{\mathbf{S}}_{(q,0)} - \eta^{(k)} \nabla_{\hat{\mathbf{S}}_{(q,0)}} E_\psi \Big(\hat{\mathbf{S}}_{(q,0)}; \boldsymbol{\tau}_q \Big), \qquad \eta^{(k)} = \eta \sqrt{1-\bar{\alpha}_k}, \tag{14}$$

where η is a step-size coefficient. This applies guidance to the predicted clean trajectory rather than the noisy latent, following Steered Diffusion (Nair et al., 2023).

Using $\tilde{\mathbf{S}}_{(q,0)}$ (with gradient guidance), we update the latent as

$$\mathbf{S}_{(q,k-1)} = \sqrt{\bar{\alpha}_{k-1}}\, \tilde{\mathbf{S}}_{(q,0)} + \sqrt{1-\bar{\alpha}_{k-1}}\, \boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q). \tag{15}$$

Self-normalized importance sampling guidance. Alternatively, instead of applying gradient guidance, we bias sampling toward low-energy trajectories via self-normalized importance sampling. At diffusion step k, we draw M candidate updates by sampling $\mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and forming

$$\mathbf{S}^{(m)}_{(q,k-1)} = \sqrt{\bar{\alpha}_{k-1}}\, \hat{\mathbf{S}}_{(q,0)} + \sqrt{1-\bar{\alpha}_{k-1}}\, \boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q) + \sigma_k\, \mathbf{z}_m, \qquad m = 1, \ldots, M, \tag{16}$$

and σ_k is the standard deviation of the Gaussian noise injected in the DDIM reverse step.

For each proposal, we compute an energy-based weight

$$w_m = \exp\Big(-\lambda\, E_\psi(\hat{\mathbf{S}}^{(m)}_{(q,0)}; \boldsymbol{\tau}_q) \Big), \qquad \bar{w}_m = \frac{w_m}{\sum_{j=1}^M w_j}, \tag{17}$$

and select one proposal $m^* \sim \text{Categorical}(\bar{w})$ to continue the reverse process. Here $\hat{\mathbf{S}}^{(m)}_{(q,0)}$ denotes the clean estimate associated with the mth proposal, and λ controls guidance strength (we set $\lambda = 1$ in this paper). The full procedure is summarized in Algorithm 2.

4 Experiments

Rigid arm configuration. We use MuJoCo to simulate two dissipation-free rigid arms (no damping and no friction). The first arm consists of 3 hinge joints rotating around the x, y, and z axes,

Algorithm 2 Hamiltonian-guided DDIM Sampling

Input: Query times $\mathbf{T}_q$, control torques $\boldsymbol{\tau}_q$, context ratio r_c, step size $\eta^{(k)} = \eta\sqrt{1-\bar{\alpha}_k}$, candidate number M
Output: Generated states $\mathbf{S}_{(q,0)}$
Initialize $\mathbf{S}_{(q,K)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; set $N_c = \max(1, \lfloor r_c \cdot N_q \rfloor)$
for $k = K$ **down to** 1 **do**
  $\mathbf{Q}_k \leftarrow [\mathbf{S}_{(q,k)}, \mathbf{T}_q]$
  Set $\mathbf{C}_k$ as a random subset of N_c pairs from $\mathbf{Q}_k$ (together with any observed conditional pairs)
  Compute $\hat{\mathbf{S}}_{(q,0)}$ using Eq. 13
  if resampling **then**
    Obtain M candidate updates $\mathbf{S}^{(m)}_{(q,k-1)}$ using Eq. 16
    Select $\mathbf{S}_{(q,k-1)}$ by weights calculated in Eq. 17
  else
    $\tilde{\mathbf{S}}_{(q,0)} \leftarrow \hat{\mathbf{S}}_{(q,0)} - \eta^{(k)} \nabla_{\hat{\mathbf{S}}_{(q,0)}} E(\hat{\mathbf{S}}_{(q,0)}; \boldsymbol{\tau}_q)$
    $\mathbf{S}_{(q,k-1)} \leftarrow \sqrt{\bar{\alpha}_{k-1}}\tilde{\mathbf{S}}_{(q,0)} + \sqrt{1-\bar{\alpha}_{k-1}} \cdot \boldsymbol{\epsilon}_\theta(k, \mathbf{C}_k, \mathbf{Q}_k, \boldsymbol{\tau}_q)$
  end if
end for
Return $\mathbf{S}_{(q,0)}$

respectively. The second arm is a two-link chain with 2 hinge joints, both rotating around the z axis, forming a planar double pendulum. Each hinge joint is actuated by a motor that applies torque. We refer to these as the 3-DoF and 2-DoF arms.

Trajectory generation. We simulate each trajectory for $T = 1000$ steps with timestep $\Delta t = 2 \times 10^{-4}$ s. We use a sum of 5 random sinusoids to create control torques and apply the torques to the arm, and record the angular positions $\mathbf{q}_{t_i}$, the momenta $\mathbf{p}_{t_i}$, and the applied torques $\boldsymbol{\tau}_{t_i}$ at each timestep t_i. Note that the momentum is calculated via MuJoCo's mass matrix. Using this method, we finally keep 40000 training trajectories. Additionally, to test the generalization performance, we generate diverse control torques with different torque policies, e.g., Gaussian process (GP), zero-value torque, and cubic splines. For each policy, we generate 100 torque sequences for test. Details are in Appendix B.

4.1 Evaluation Metrics

Let $\mathbf{q}_{t_i}, \mathbf{p}_{t_i}$ and $\hat{\mathbf{q}}_{t_i}, \hat{\mathbf{p}}_{t_i}$ denote the generated and MuJoCo reference states (position and momentum) at time t_i, respectively. Reference trajectories are obtained by rolling out the same initial state using the same control torques $\boldsymbol{\tau}_{t_1:t_T}$ in MuJoCo. We use $\mathbf{q}, \mathbf{p}$ to represent the full sequences.

Root Mean Square Error (RMSE). We report the RMSE between the generated trajectories and their MuJoCo references to quantify the accumulated deviation. Specifically, we compute the RMSE for position and momentum independently, averaged across their respective dimensions.

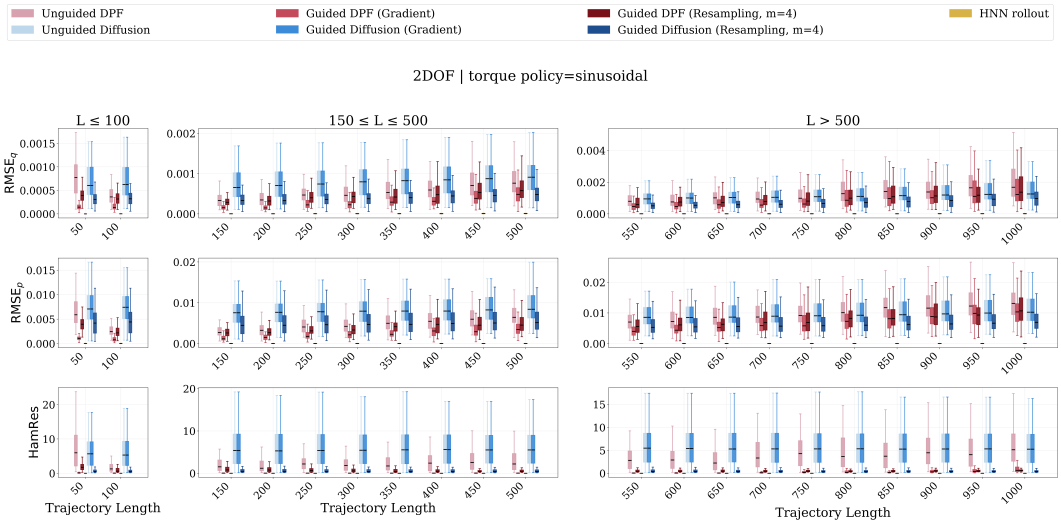
Hamiltonian Residual (HamRes). To assess how well the generated trajectories adhere to the underlying physical laws, we define the residuals of Hamilton's equations using the HNN Hamiltonian H_ψ: $\mathbf{r}^q_{t_i} = \frac{\partial H_\psi}{\partial \mathbf{p}}(\mathbf{q}_{t_i}, \mathbf{p}_{t_i}) - \dot{\mathbf{q}}_{t_i}, \quad \mathbf{r}^p_{t_i} = \frac{\partial H_\psi}{\partial \mathbf{q}}(\mathbf{q}_{t_i}, \mathbf{p}_{t_i}) - (\boldsymbol{\tau}_{t_i} - \dot{\mathbf{p}}_{t_i})$ and define Hamiltonian residual as

$$\mathrm{HamRes} = \frac{1}{T}\sum_{i=1}^{T} \left(\|\mathbf{r}^q_{t_i}\|_2^2 + \|\mathbf{r}^p_{t_i}\|_2^2 \right). \tag{18}$$

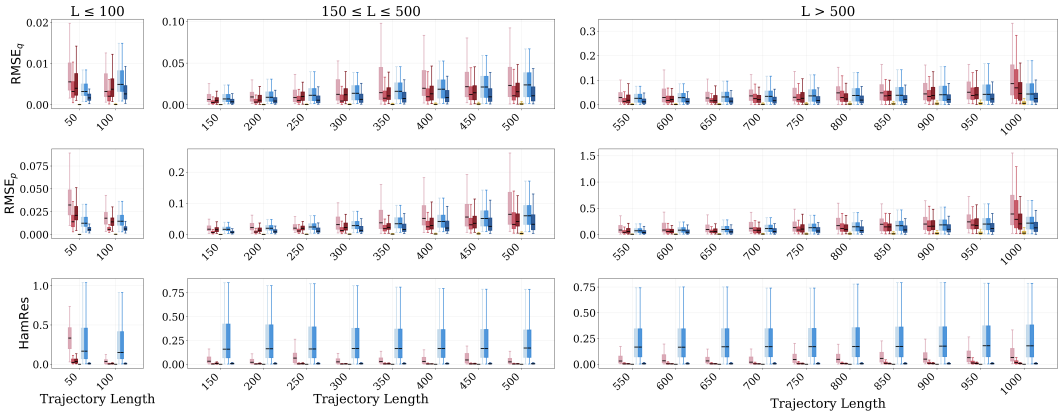
HamRes measures the local consistency between the generated trajectory and the HNN-implied dynamics. Since finite-difference estimates for $\dot{\mathbf{q}}_{t_i}$ and $\dot{\mathbf{p}}_{t_i}$ can be sensitive to local temporal outliers, we utilize a robust variant of HamRes provided in Appendix C.

4.2 Results

We demonstrate the effectiveness of our method through two experiments: (1) We generate variable-length trajectories with and without Hamiltonian guidance across different torque policies, then compare their RMSE and HamRes. (2) We compare the inference latency between HNN rollout and



(a) RMSE and HamRes on the 2-DoF setting



(b) RMSE and HamRes on the 3-DoF setting

Figure 2: Box plots of RMSE and HamRes across different lengths using sinusoidal torque policy (a) 2-DoF results (b) 3-DoF results. We compare HNN rollouts, unguided generation, and guided generation. Best zoomed in and viewed in color.

our method with fixed trajectory length. Both experiments are conducted on the 2-DoF and 3-DoF arms.

In the first experiment, we compare the results obtained from unguided DPF, guided DPF, unguided diffusion (Song et al., 2020), guided diffusion, and HNN rollout (Toth et al., 2020). Here, diffusion uses transformer blocks with comparable GPU compute. See appendix A for details about the architecture of the transformer-based diffusion. Note that the transformer-based diffusion model generates fixed-length trajectories ($T = 1000$), which we subsequently trim to the desired length. For DPF and diffusion, we use DDIM sampling with 20 steps. Context ratio r_c is fixed to 50% of the trajectory length. We apply two different guidance mechanisms described in Section 3 for comparison: (1) Self-normalized importance sampling with candidate number $M = 4$. (2) One-step guidance with step size coefficient $\eta = 0.01$.

In the second experiment, we compare the inference latency of HNN rollout and our method across different batch sizes. To ensure the fairness of the comparison, we only start to record the inference latency when data loading is finished and running is stable.

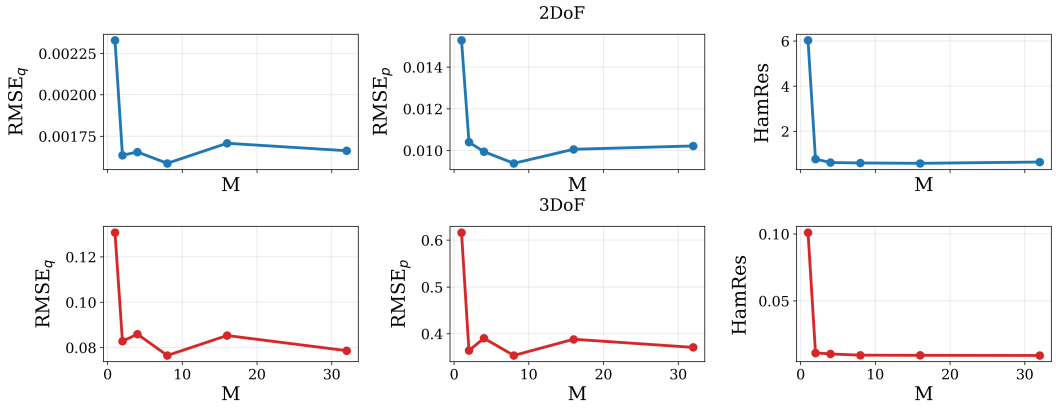


Figure 3: Performance sensitivity to the number of resampling candidates (M).

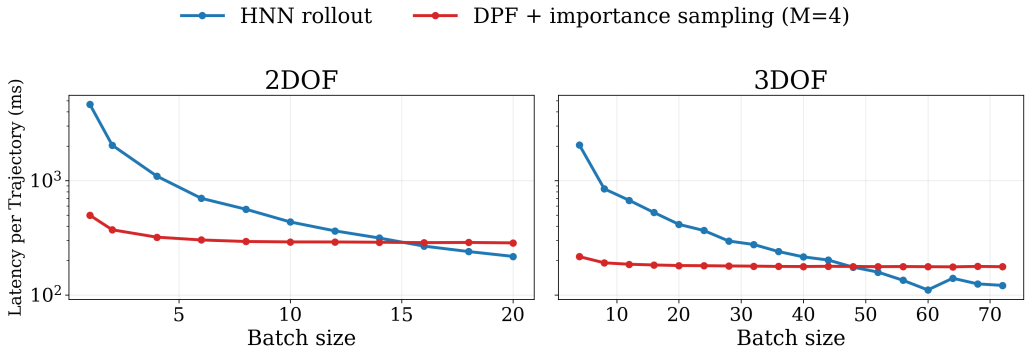


Figure 4: Latency per trajectory vs. batch size.

4.2.1 Variable-Length Trajectory Generation with Hamiltonian Guidance

We test trajectory generation for lengths in $\{50, 100, 150, \ldots, 1000\}$. Note that the model was not trained on trajectories with lengths $\{50, 150, \ldots, 950\}$.

As illustrated in Figure 2, our model achieves consistently low RMSE across both seen and unseen trajectory lengths. The introduction of guidance yields a systematic improvement in both RMSE and HamRes for both the DPF and the diffusion baseline. When guidance is active, our method achieves RMSE performance comparable to that of the transformer-based diffusion model. Notably, the DPF is more computationally efficient for shorter trajectories as it requires fewer query pairs. In contrast, transformer-based diffusion requires the generation of the entire trajectory, and its computational costs scales quadratically with sequence length. Furthermore, additional results in Appendix D demonstrate consistent performance across varying torque policies, highlighting the model's ability to generalize to unseen control torques. Figure 3 shows the effect of candidate number when using importance sampling. The guidance achieves optimal performance when M is 8, indicating a further improvement when using more candidates. Qualitative visualizations of the generated trajectories are provided in Appendix E.

4.2.2 Comparison of Inference Latency

Figure 4 reports per-trajectory inference latency versus batch size. In the small-batch regime, DPF with importance-sampling guidance attains lower latency than HNN rollout, indicating a small-batch latency advantage for DPF under this implementation setup.

5 CONCLUSION

We proposed a method to generate variable-length rigid arm trajectories that are physically plausible. Our key insight is that combining diffusion fields with physics-based energy guidance separates two concerns: the DPF learns to generate variable-length trajectories that capture the data distribution, while the HNN-derived energy function provides an optimization target for enforcing Hamiltonian consistency during sampling. The model also uses a AdaLN-based causal conditioning, modulating current-state features with a joint embedding of the previous state and torque. Experiments show our Hamiltonian guidance consistently reduces the misalignment between generated trajectories and their references. Our results suggest that separating distribution learning from physics enforcement is a promising direction for physically consistent trajectory generation, and we expect this approach to extend to higher-DoF systems and dissipative dynamics.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023.

Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-informed diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2024.

Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1916–1923. IEEE, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Shaan Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *CoRR*, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. *Advances in neural information processing systems*, 32, 2019.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2021.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Wonsuhk Jung, Utkarsh Aashu Mishra, Nadun Ranawaka Arachchige, Yongxin Chen, Danfei Xu, and Shreyas Kousik. Joint model-based model-free diffusion for planning with constraints. In *Conference on Robot Learning*, pp. 4328–4350. PMLR, 2025.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European conference on computer vision*, pp. 423–439. Springer, 2022.

Xu-Hui Liu, Tian-Shuo Liu, Shengyi Jiang, Ruifeng Chen, Zhilong Zhang, Xinwei Chen, and Yang Yu. Energy-guided diffusion sampling for offline-to-online reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 31541–31565, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu based physics simulation for robot learning. In *NeurIPS Datasets and Benchmarks*, 2021.

Nithin Gopalakrishnan Nair, Anoop Cherian, Suhas Lohit, Ye Wang, Toshiaki Koike-Akino, Vishal M Patel, and Tim K Marks. Steered diffusion: A generalized framework for plug-and-play conditional image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20850–20860, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Anuj Pasricha, Joewie J Koh, Jay Vakil, and Alessandro Roncone. Dynamics-compliant trajectory diffusion for super-nominal payload manipulation. In *Conference on Robot Learning*, pp. 4908–4925. PMLR, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Kallol Saha, Vishal Mandadi, Jayaram Reddy, Ajit Srikanth, Aditya Agarwal, Bipasha Sen, Arun Singh, and Madhava Krishna. Edmp: Ensemble-of-costs-guided diffusion for motion planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10351–10358, 2024. doi: 10.1109/ICRA57147.2024.10610519.

Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Robot motion diffusion model: Motion generation for robotic characters. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400711312. doi: 10.1145/3680528.3687626. URL https://doi.org/10.1145/3680528.3687626.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Peter Toth, Danilo J. Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJenn6VFvB.

Julen Urain, Ajay Mandlekar, Yilun Du, Mahi Shafiullah, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, and Jan Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *CoRR*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Rosa Wolf, Yitian Shi, Sheng Liu, and Rania Rayyes. Diffusion models for robotic manipulation: A survey. *Frontiers in Robotics and AI*, 12:1606247, 2025.

Xinyao Yu, Sixian Zhang, Xinhang Song, Xiaorong Qin, and Shuqiang Jiang. Trajectory diffusion for objectgoal navigation. *Advances in Neural Information Processing Systems*, 37:110388–110411, 2024.

Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16010–16021, 2023.

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Diffusion model predictive control. *Transactions on Machine Learning Research*, 2024a.

Zihan Zhou, Xiaoxue Wang, and Tianshu Yu. Generating physical dynamics under priors. In *The Thirteenth International Conference on Learning Representations*, 2024b.

Peiye Zhuang, Samira Abnar, Jiatao Gu, Alex Schwing, Joshua M Susskind, and Miguel Angel Bautista. Diffusion probabilistic fields. In *The Eleventh International Conference on Learning Representations*, 2023.

A Model Architectures and Training

Hamiltonian Neural Network (HNN). We use a separable Hamiltonian neural network that decomposes the Hamiltonian as $H(\mathbf{p}_t, \mathbf{q}_t) = K(\mathbf{p}_t, \mathbf{q}_t) + V(\mathbf{q}_t)$, where K is the kinetic energy and V is the potential energy. The kinetic energy network K takes the concatenated state $[\mathbf{p}_t, \mathbf{q}_t]$ as input and passes it through four fully connected layers of width 1024 with CELU activations, followed by a linear projection to a scalar. The potential energy network V has the same architecture but takes only $\mathbf{q}_t$.

We train with AdamW (learning rate 3×10^{-4}) and cosine annealing to 10^{-6}. We use a batch size of 8192, gradient clipping at norm 1.0, float32 precision, and train for 1000 epochs.

Diffusion Probabilistic Fields (DPF). The DPF generates state trajectories $\mathbf{s}_{t_1:t_T} = [\mathbf{s}_{t_1}, \ldots, \mathbf{s}_{t_T}]$ conditioned on control torques $\boldsymbol{\tau}_{t_1:t_T}$ via a denoising diffusion process with 1000 timesteps following a cosine noise schedual (Nichol & Dhariwal, 2021). The backbone is a Perceiver IO architecture detailed in Table 1.

Each input token is the concatenation of the noised state, a Fourier embedding of the diffusion step, and a sinusoidal temporal position encoding (Vaswani et al., 2017). Torque conditioning enters through the AdaLN-based causal conditioning described in Section 3.1: separate MLPs embed the torque $\boldsymbol{\tau}_{t_{i-1}}$ and the current noisy state $\mathbf{s}_{t_{i-1}}$, an interaction MLP fuses them into $\mathbf{h}_{t_i}$, and the result modulates scale and shift parameters via AdaLN at every self-attention block.

We train with AdamW (learning rate 10^{-4}) using 1000-step linear warmup followed by cosine decay. Batch size is 128, gradient clipping at norm 1.0. During training, the number of query pairs N_q is sampled from $\{100, 200, \ldots, 1000\}$ per batch, and the number of context pairs N_c is sampled from $\mathcal{U}\{1, \ldots, N_q - 1\}$. We train for 3000 epochs. Table 2 summarizes the training hyperparameters.

At inference we use DDIM with 20 steps, a context ratio $r_c = 0.5$.

Table 1: Perceiver IO architecture for DPF

Component	**Configuration**
Latent array size	256
Latent dimension	256
Encoder	
Cross-attention layers	1
Self-attention blocks	8
Decoder	
Cross-attention layers	1
Self-attention blocks	4
Attention settings	
Number of heads	8
Head dimension	32
Feedforward expansion	4
Input embeddings	
Diffusion step encoding	Fourier, 64 bands ($\mathbb{R}^{128}$)
Temporal position encoding	Sinusoidal, 32 bands ($\mathbb{R}^{64}$)
AdaLN embedding dimension	256

Transformer-based diffusion baseline. For comparison, we replace the Perceiver IO backbone with a standard Transformer encoder while keeping the same diffusion objective and training protocol. Given the state trajectory $\mathbf{s}_{t_1:t_T}$ and torques $\boldsymbol{\tau}_{t_1:t_T}$, each input token is formed by concatenating: (1) the noised normalized state at the current diffusion step, (2) torque, (3) Fourier encoding of the diffusion step, and (4) sinusoidal temporal position encoding.

Table 2: Hyperparameters for HNN and DPF training

Hyperparameter	**HNN**	**DPF**
Learning rate	3×10^{-4}	1×10^{-4}
Optimizer	AdamW	AdamW
Batch size	8192	128
Gradient clip norm	1.0	1.0
Training epochs	1000	3000
LR schedule	Cosine annealing	Cosine decay
Diffusion steps K	–	1000
Noise schedule	–	Cosine

All tokens are first mapped by a linear projection to a hidden dimension of 256, then processed by 4 Transformer encoder blocks, and finally projected back to the state dimension to predict noise. Architecture configuration is in Table 3.

Unlike DPF, this baseline has no latent bottleneck, no context-query split, and no AdaLN torque modulation; conditioning is injected directly through token concatenation.

Table 3: Transformer architecture for diffusion baseline

Component	**Configuration**
Backbone	Transformer encoder
Input projection	Linear($C_{\text{in}} \to 256$) + LayerNorm
Encoder layers	4
Attention heads	8
Head dimension	32
Feedforward expansion	4
Output projection	LayerNorm + Linear($256 \to d_{\text{state}}$)

B TORQUE POLICY DETAILS

Each torque policy generates a sequence $\boldsymbol{\tau}_{t_1:t_T}$ with $T = 1000$ timesteps. For actuator j, torque at timestep t_i is denoted by $\tau_{t_i}^{(j)}$.

Sinusoidal. Each actuator receives a sum of $M = 5$ sinusoids:

$$\tau_{t_i}^{(j)} = \sum_{m=1}^{M} A_{j,m} \, \sin(\omega_{j,m} \, t_i + \varphi_{j,m}), \tag{19}$$

with

$$A_{j,m} \sim \mathcal{U}(0,\, 0.5), \quad \omega_{j,m} \sim \mathcal{U}(0,\, 6\pi), \quad \varphi_{j,m} \sim \mathcal{U}(0,\, 2\pi). \tag{20}$$

The sampling rule is the same for both systems; only actuator dimension differs ($j = 1, 2$ for 2DoF and $j = 1, 2, 3$ for 3DoF).

Gaussian process. For each actuator, we sample i.i.d. standard Gaussian noise over time and smooth it along the time axis with a Gaussian kernel. For 3DoF, we use kernel width $\sigma = 100$ timesteps. For 2DoF, we set $\sigma = 115$.

Cubic spline. For each actuator, we sample random waypoints and fit a natural cubic spline through them. For 3DoF, we use $n \sim \mathcal{U}\{8, 9, \ldots, 15\}$ waypoints and waypoint values from $\mathcal{U}(-0.5, 0.5)$. For 2DoF, $n \sim \mathcal{U}\{10, 11, \ldots, 14\}$, waypoint values from $\mathcal{U}(-0.5, 0.5)$, and final torque clipping to $[-1.5, 1.5]$.

Zero value torques. For each actuator, we set the torque value as 0.

Amplitude matching. For the three non-sinusoidal policies, we rescale each actuator dimension so that its standard deviation matches a reference value. The reference is the per-dimension standard deviation computed over 100 sinusoidal trajectories generated with the parameters above.

Trajectory filtering. In a dissipation-free system with external forcing, velocities can grow unbounded. To ensure numerical stability during training, we focus on moderate-energy regimes typical of practical applications. We discard trajectories where joint velocities exceed 10 rad/s or accelerations exceed 100 rad/s^2. To further ensure numerical stability, we reject trajectories where the total energy exceeds 500 J or grows to more than 5 times the initial energy.

C METRICS DETAILS

HamRes used in evaluation. For each trajectory, we compute

$$\mathbf{r}^q_{t_i} = \frac{\partial H_\psi}{\partial \mathbf{p}}(\mathbf{p}_{t_i}, \mathbf{q}_{t_i}) - \dot{\mathbf{q}}_{t_i}, \qquad \mathbf{r}^p_{t_i} = \frac{\partial H_\psi}{\partial \mathbf{q}}(\mathbf{p}_{t_i}, \mathbf{q}_{t_i}) - \big(\boldsymbol{\tau}_{t_i} - \dot{\mathbf{p}}_{t_i}\big), \tag{21}$$

where $\dot{\mathbf{q}}_{t_i}, \dot{\mathbf{p}}_{t_i}$ are obtained by central difference.

Before differentiation, we apply light Gaussian smoothing along time to reduce finite-difference amplification. Then residuals are normalized dimension-wise:

$$\tilde{\mathbf{r}}^q_{t_i} = \frac{\mathbf{r}^q_{t_i}}{\boldsymbol{\sigma}_q}, \qquad \tilde{\mathbf{r}}^p_{t_i} = \frac{\mathbf{r}^p_{t_i}}{\boldsymbol{\sigma}_p}, \tag{22}$$

where σ_q, σ_p are per-dimension scales calculated from HNN training statistics with minimum floors.

Per timestep, we use pseudo-Huber penalty

$$\phi_\delta(x) = \delta^2 \left(\sqrt{1 + \left(\frac{x}{\delta}\right)^2} - 1 \right), \tag{23}$$

and define

$$e_{t_i} = \frac{1}{d_q} \sum_{j=1}^{d_q} \phi_\delta\left(\tilde{r}^{q,(j)}_{t_i}\right) + \frac{1}{d_p} \sum_{j=1}^{d_p} \phi_\delta\left(\tilde{r}^{p,(j)}_{t_i}\right). \tag{24}$$

Finally, the reported HamRes is aggregated over time by median:

$$\mathrm{HamRes} = \mathrm{median}_{i=1,\dots,T}\ e_{t_i}. \tag{25}$$

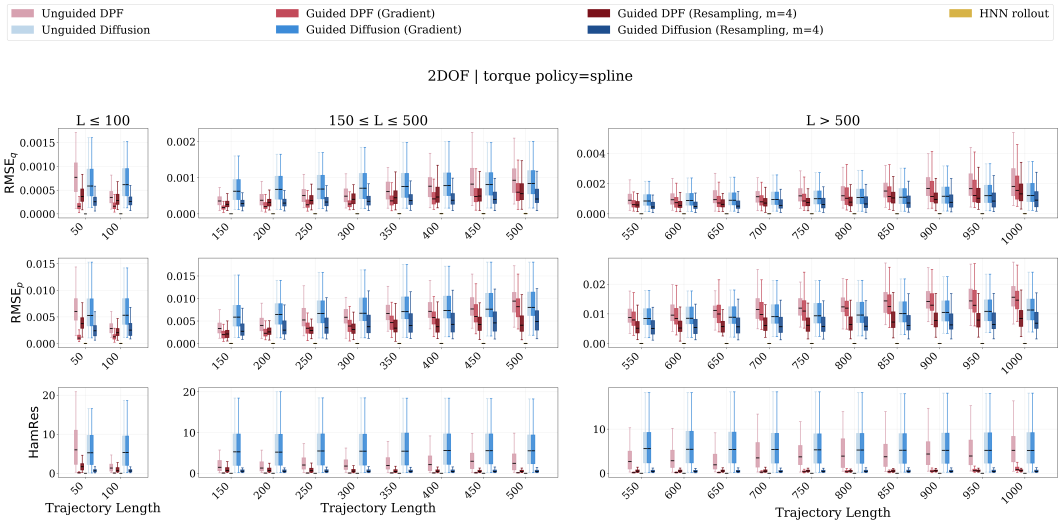
This form keeps sensitivity to consistent dynamics mismatch while avoiding domination by a few temporal outliers.

D ADDITIONAL RESULTS ON VARIABLE-LENGTH TRAJECTORY GENERATION

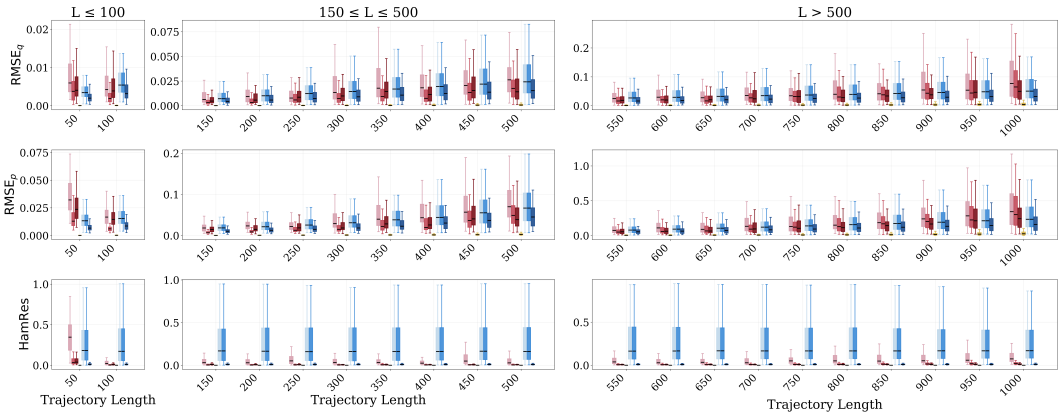
To evaluate the model's generalization capability, we employ unseen torque policies that are distinct from the sinusoidal combinations used during training. Figures 5 through 7 show the metric comparisons across these diverse control torques.

E QUALITATIVE VISUALIZATIONS

We provide qualitative visualizations of trajectories generated with and without guidance, alongside HNN rollouts for comparison. Figures 8 through 15 display representative samples produced under varying torque policies.

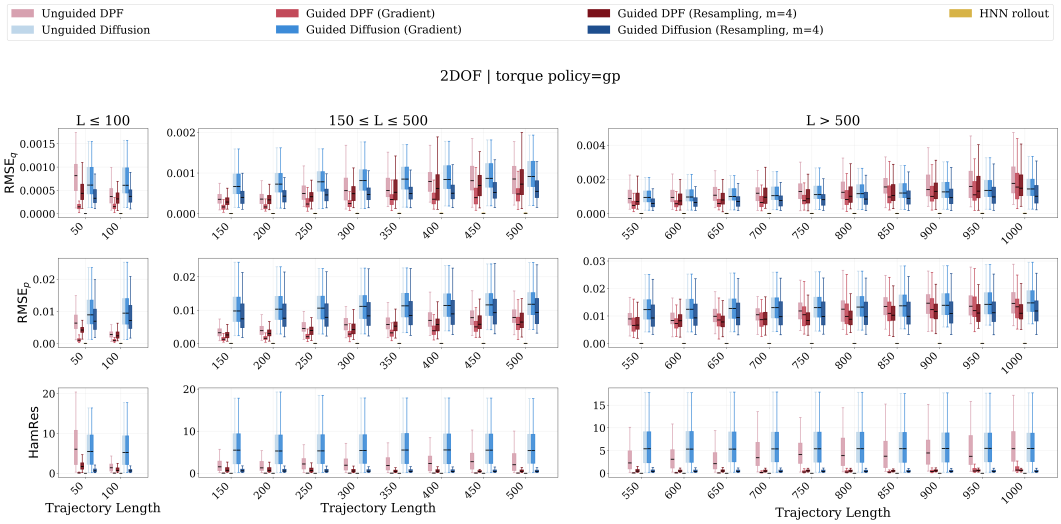


(a) RMSE and HamRes on the 2-DoF setting

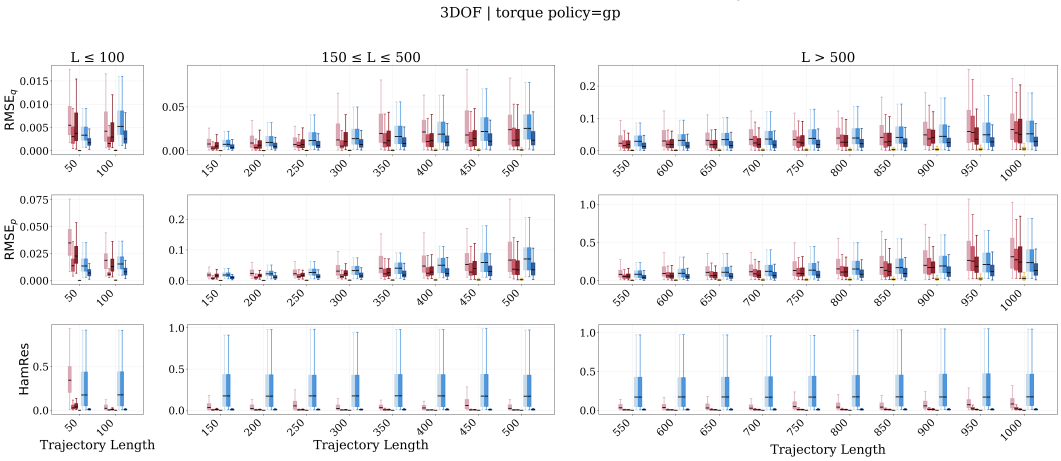


(b) RMSE and HamRes on the 3-DoF setting

Figure 5: Box plots of RMSE and HamRes across different lengths using torques created by cubic splines (a) 2-DoF results (b) 3-DoF results.

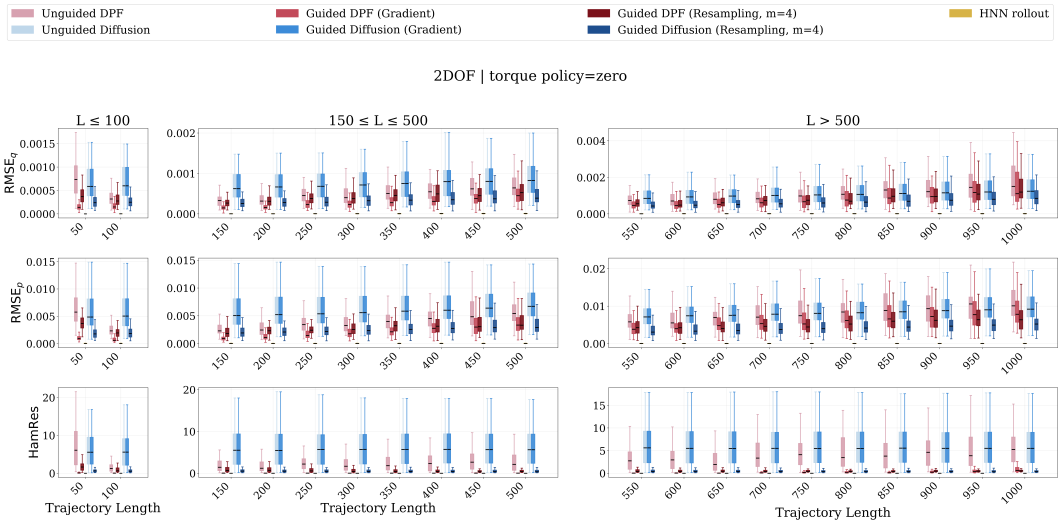


(a) RMSE and HamRes on the 2-DoF setting

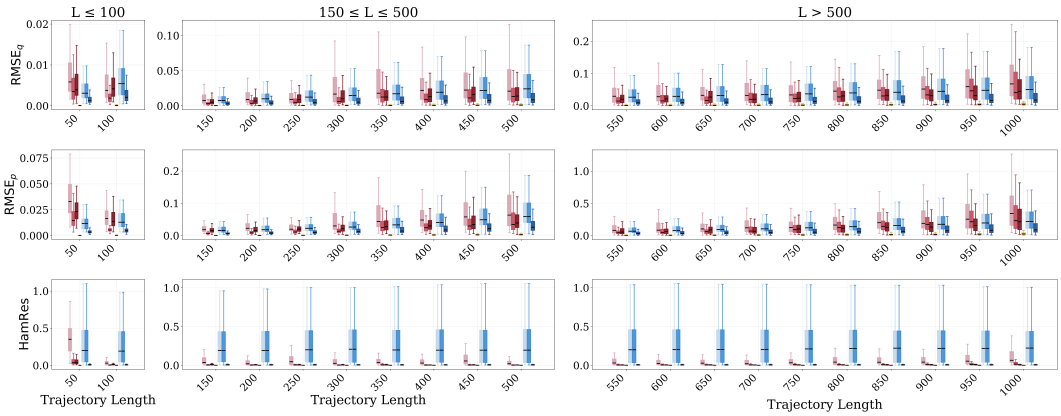


(b) RMSE and HamRes on the 3-DoF setting

Figure 6: Box plots of RMSE and HamRes across different lengths using torques created by Gaussian process (a) 2-DoF results (b) 3-DoF results.



(a) RMSE and HamRes on the 2-DoF setting



(b) RMSE and HamRes on the 3-DoF setting

Figure 7: Box plots of RMSE and HamRes across different lengths using zero value torques (a) 2-DoF results (b) 3-DoF results.

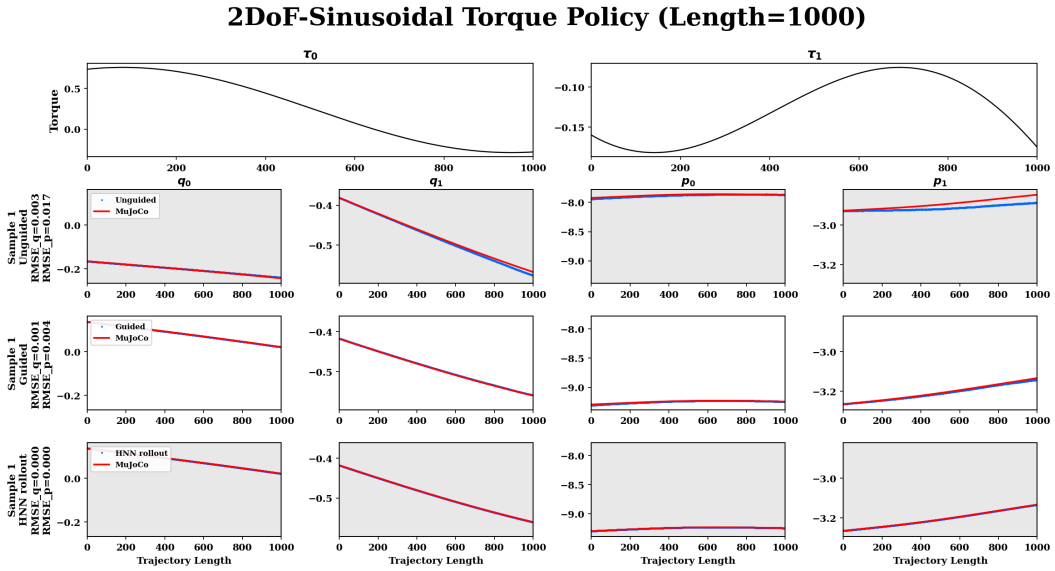


Figure 8: Qualitative comparison of 2-DoF trajectories under the sinusoidal torque policy.

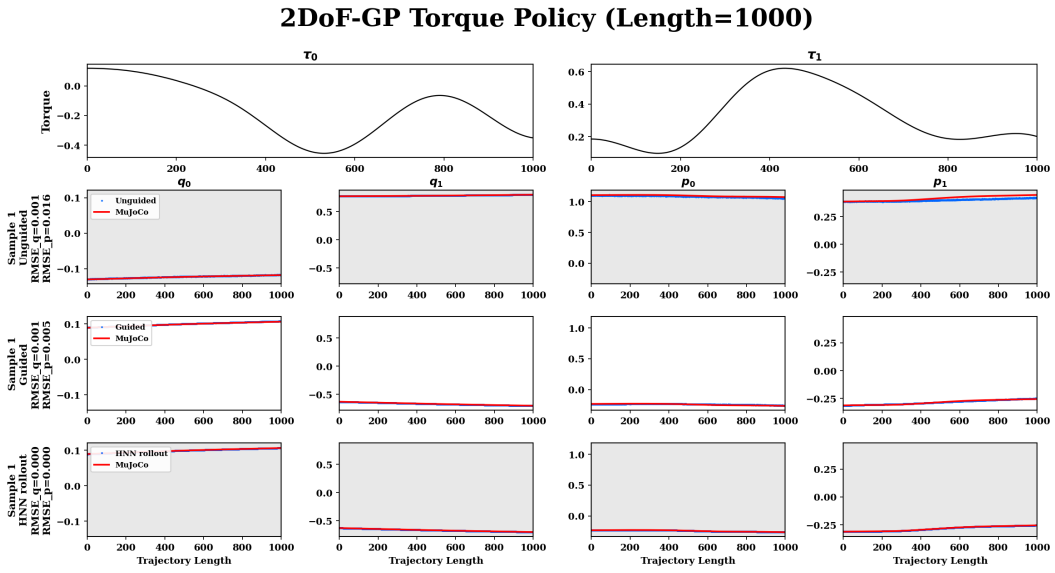


Figure 9: Qualitative comparison of 2-DoF trajectories under the Gaussian process torque policy.

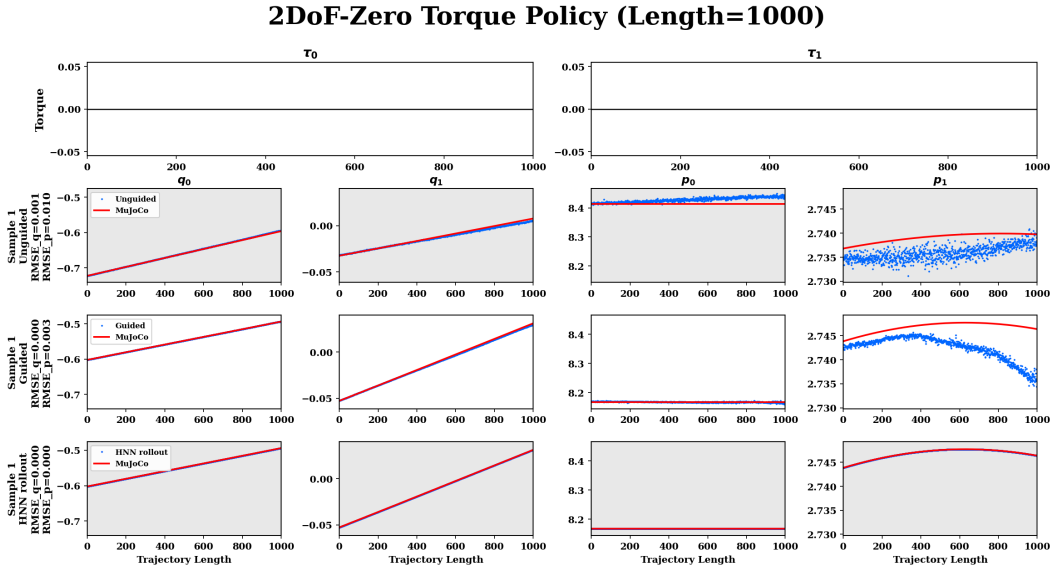


Figure 10: Qualitative comparison of 2-DoF trajectories under the zero value torques.

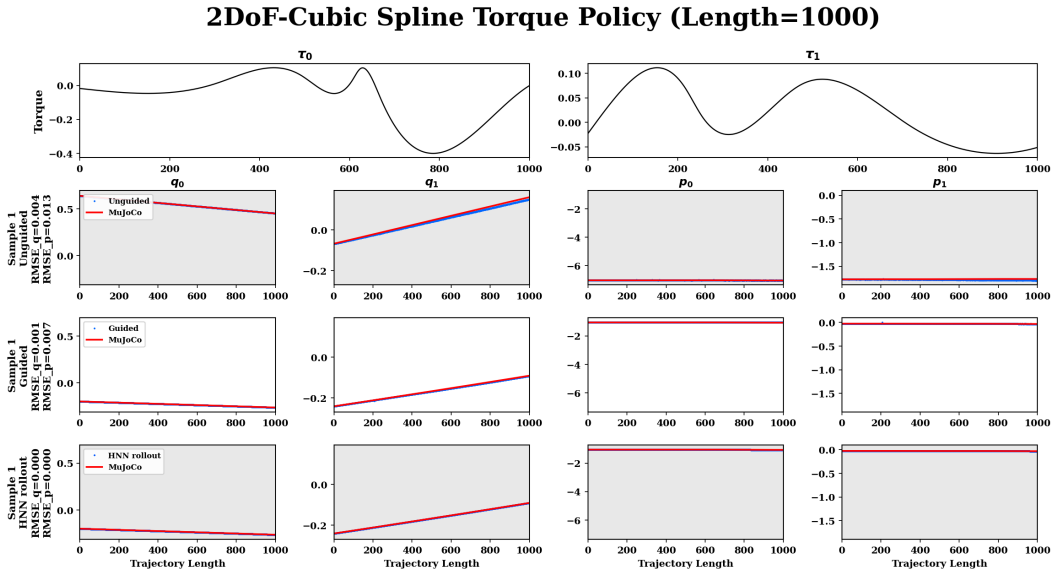


Figure 11: Qualitative comparison of 2-DoF trajectories under the cubic spline torque policy.

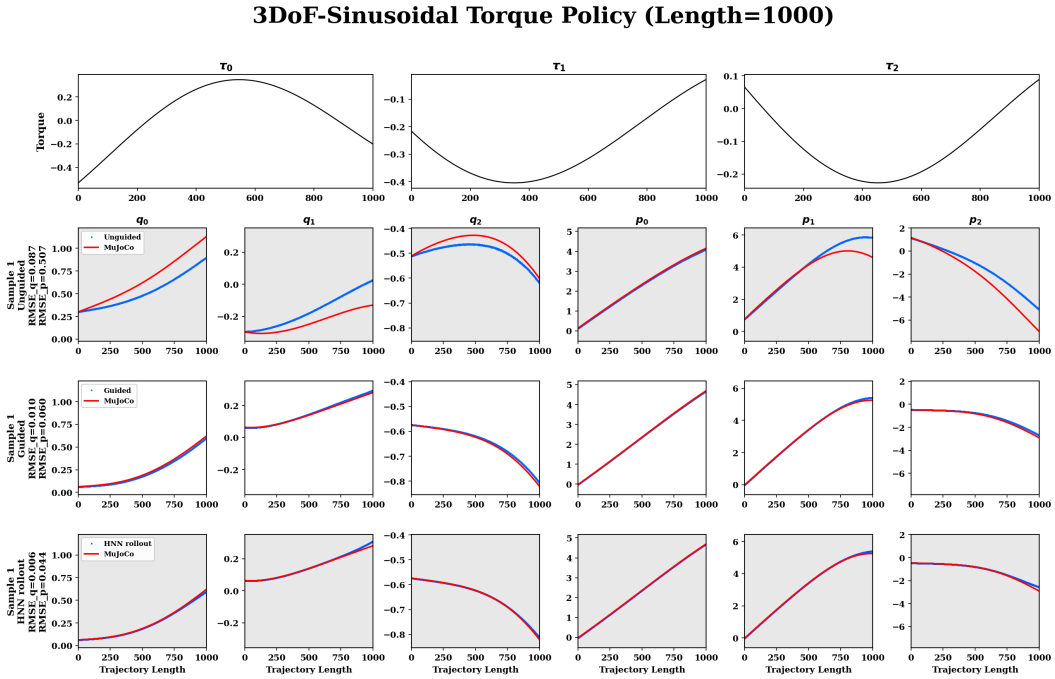


Figure 12: Qualitative comparison of 3-DoF trajectories under the sinusoidal torque policy.

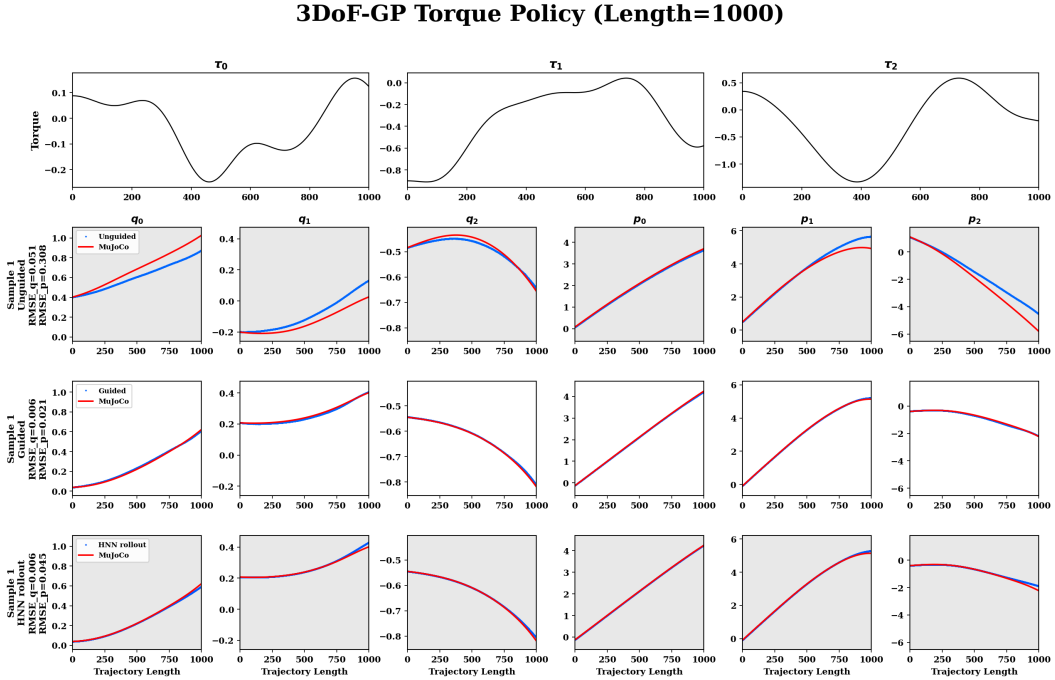


Figure 13: Qualitative comparison of 3-DoF trajectories under the Gaussian process torque policy.

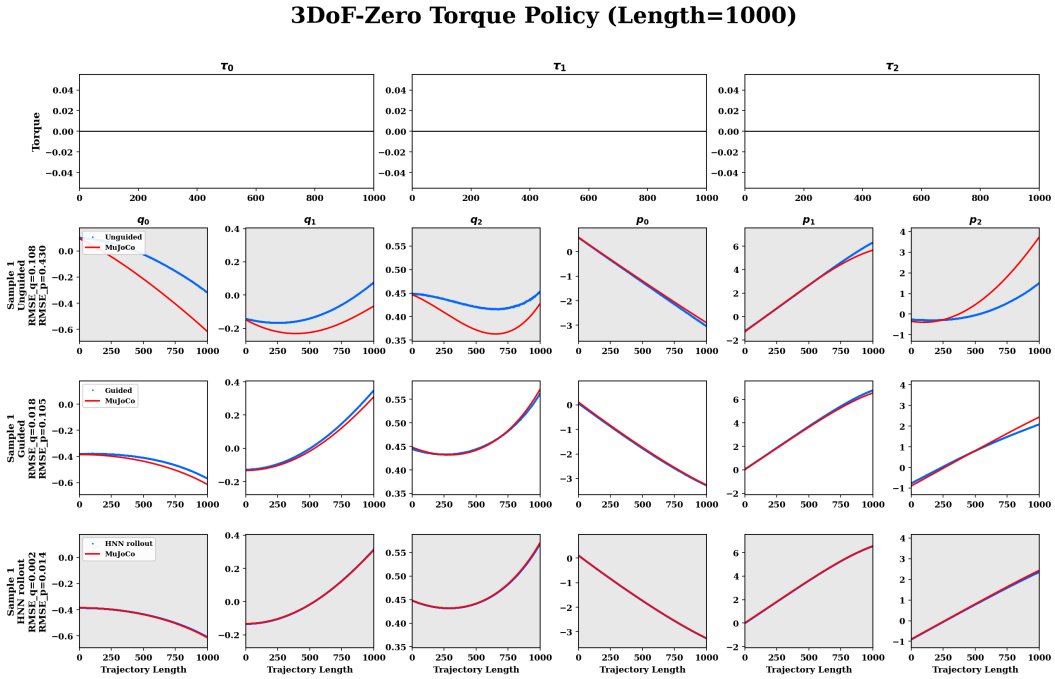


Figure 14: Qualitative comparison of 3-DoF trajectories under the zero value torques.

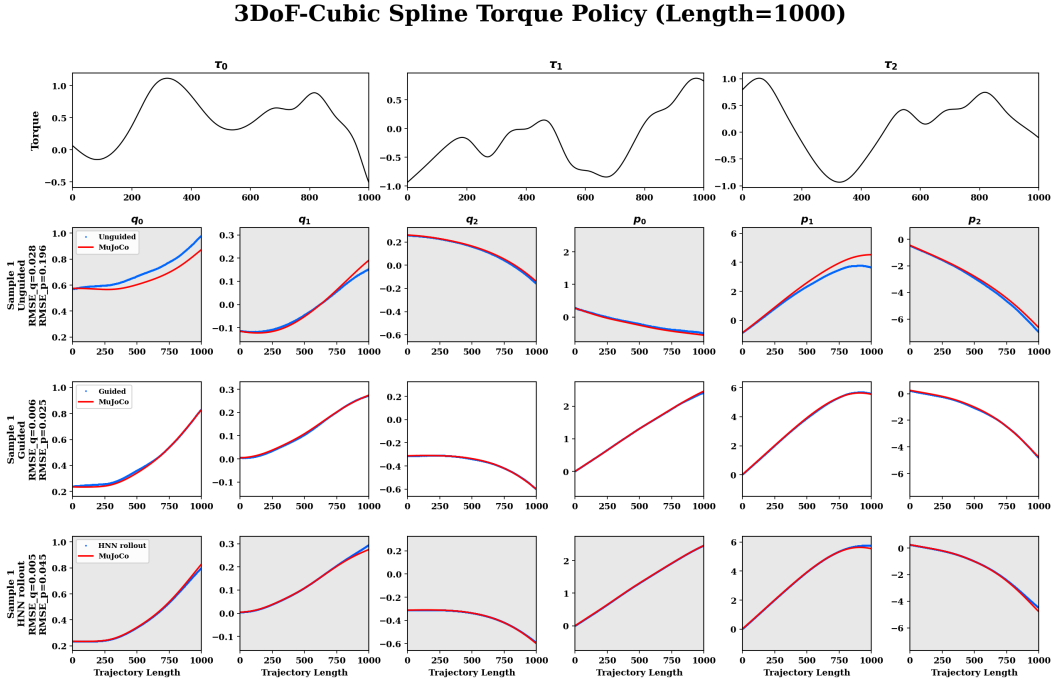


Figure 15: Qualitative comparison of 3-DoF trajectories under the cubic spline torque policy.