PATH INTEGRAL VALUE MATCHING: REGULARIZED CONTROL OF ARBITRARY MARKOV PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

KL-regularized control of continuous-time Markov processes seeks an optimal path distribution that balances expected path cost against deviation from a reference Markov process. Many existing solvers directly optimize control policies, but often require process-specific formulations and costly full-horizon rollouts, with off-policy reuse limited by accumulated likelihood-ratio variability. We propose Path Integral Value Matching (PI-VM), a process-agnostic value-based control method. Our method builds on a common path-integral representation of the value, which determines the optimal controlled dynamics through a Doob's h-transform once the reference process is fixed. However, direct path-integral estimation still requires full-horizon trajectory simulation and can suffer from high variance. By truncating and marginalizing the path integral, we obtain a temporal recursion for the value function. PI-VM learns this recursion through temporal-difference learning on short trajectories, while experience replay and likelihood correction support off-policy training. The same learning objective applies to Euclidean and manifold diffusions, continuous-time Markov chains, and finite-activity jump processes. Experiments show that PI-VM achieves the best reported performance on multiple tasks across diverse process families and performs well in image fine-tuning.

1 INTRODUCTION

KL-regularized control of continuous-time Markov processes connects probabilistic inference and stochastic control (Kappen et al., 2012). Given a reference Markov path law $\mathbb{P}^0$ and path cost $\mathcal{C}(X)$, it seeks a path distribution minimizing

$$\mathbb{E}_{\mathbb{P}}[\mathcal{C}(X)] + \lambda \mathrm{KL}(\mathbb{P} \| \mathbb{P}^0)$$

over candidate path laws $\mathbb{P} \ll \mathbb{P}^0$. It is closely connected to diffusion reward fine-tuning (Domingo-Enrich et al., 2025; Liu et al., 2025), Schrödinger bridges (Chen et al., 2021), and energy-based sampling (Liu et al., 2025; Havens et al., 2025; Zhang & Chen, 2021).

Many modern solvers follow a policy-based paradigm, including variance-based methods (Nüsken & Richter, 2021), Stochastic Optimal Control Matching (Domingo i Enrich et al., 2024), and Adjoint Matching (Domingo-Enrich et al., 2025). They optimize parameterized controls using simulated trajectories. Their formulations are often dynamics-specific, so extensions to discrete chains (So et al., 2026) or manifold-valued dynamics (Mosso et al., 2026) can require new derivations. Their training often relies on costly on-policy, full-horizon rollouts, while off-policy likelihood ratios can degenerate with horizon or dimension (Domingo-Enrich, 2024), limiting experience replay.

We observe that value representation provides a process-agnostic alternative. The Gibbs–Donsker–Varadhan principle (Deuschel & Stroock, 2001; Hartmann et al., 2017) provides a common path-integral representation of the exact value without assuming a particular form of dynamics. Once the reference process is fixed, the value function itself can determine the optimal Markov dynamics through a Doob's h-transform (Oksendal, 2013). However, direct path integral value learning still requires full-horizon trajectory simulation to the terminal time, with potentially high-variance exponential weights (Thijssen & Kappen, 2015).

To address this bottleneck, we truncate and marginalize the path integral to derive a temporal recursion for the value function. Building on this recursion, we propose **Path Integral Value Matching (PI-VM)**. PI-VM learns the value from short trajectory segments, replacing full-horizon continuations in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

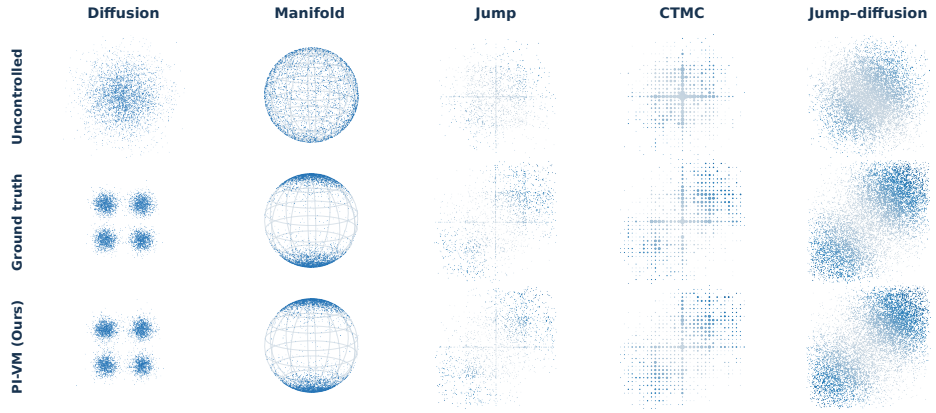

Figure 1: Terminal distributions under PI-VM and ground-truth optimal controls. PI-VM learns controls that steer the process to terminal distributions closely matching those induced by the ground-truth optimal controls across five stochastic process settings: Euclidean diffusion, manifold diffusion, continuous-state jump processes, finite-state continuous-time Markov chains (CTMCs), and jump-diffusions.

its regression targets with value bootstrapping. We further combine experience replay with likelihood correction to support off-policy training.

This learning principle applies to Euclidean and manifold diffusions, continuous-time Markov chains (CTMCs), and finite-activity jump processes. Although simulation, likelihood evaluation, and control recovery are process-specific, the value-learning objective and training procedure remain unchanged.

Our contributions are:

- Building on classical path-integral and Doob's h-transform, we formulate a common value-learning framework for KL-regularized control of continuous-time Markov processes.
- We develop PI-VM, a temporal-difference solver that learns from short trajectories and supports off-policy training through experience replay and likelihood correction.
- We instantiate the framework on Euclidean diffusions, manifold diffusions, CTMCs, and finite-activity jump processes, obtaining low control or induced-drift errors across these families and lower reported OU update costs.

2 Related Work

Policy-based stochastic control. Policy-based solvers parameterize controlled dynamics and optimize relative entropy, cross-entropy, variance, or log-variance objectives (Holdijk et al., 2023; Nüsken & Richter, 2021). Stochastic Optimal Control Matching and Adjoint Matching instead regress controls against stochastic or adjoint-based targets (Domingo i Enrich et al., 2024; Domingo-Enrich et al., 2025; Domingo-Enrich, 2024). Trust-region methods stabilize path-space optimization (Blessing et al., 2025; von Klitzing et al., 2025); discrete and manifold extensions adapt adjoint targets to their state spaces (So et al., 2026; Mosso et al., 2026). PI-VM takes a different route by separating value learning from the particular form of the control: it learns a common scalar value representation and recovers the controlled dynamics through the Doob h-transform associated with the reference process, rather than designing a separate control-regression target for each process family.

Path-integral and value-based methods. Grid-based HJB solvers become costly in high dimensions (Frédéric Bonnans, J. et al., 2004; Jensen & Smears, 2013). For suitably structured KL control, exponential transformations yield linear value equations (Kappen, 2005; Todorov, 2006), supporting methods such as MPPI and adaptive importance sampling (Williams et al., 2017; Kappen & Ruiz, 2016). Value Matching (VM) learns a value from sampled current-policy costs-to-go to guide a frozen generator (Jensen et al., 2026). Value-gradient guidance uses reward derivatives to construct guidance targets (Liu et al., 2026). Closely related off-policy VM uses trajectory-residual objectives, capped

Table 1: One value-to-process principle, different control forms. Substituting each reference generator into Eq. 7 determines its optimal drift or jump-rate modification.

Reference process	Generator $\mathcal{L}^0\varphi$	Optimal modification
Euclidean diffusion	$b^0 \cdot \nabla\varphi + \frac{1}{2}\operatorname{tr}(a\nabla^2\varphi)$	$b^\star = b^0 - \lambda^{-1} a\nabla V$
Manifold diffusion	$b^0\varphi + \frac{1}{2}\sum_i H_i(H_i\varphi)$	$b^\star = b^0 - \lambda^{-1}\sum_i (H_i V)H_i$
CTMCs	$\sum_{j\neq i} q^0_{ij}(\varphi_j - \varphi_i)$	$q^\star_{ij} = q^0_{ij} e^{(V_i - V_j)/\lambda}$
Finite-activity jumps	$\int [\varphi(y) - \varphi(x)]\, \nu^0(x, dy)$	$\nu^\star(x, dy) = e^{(V_x - V_y)/\lambda}\nu^0(x, dy)$
Jump–diffusion	$b^0 \cdot \nabla\varphi + \frac{1}{2}\operatorname{tr}(a\nabla^2\varphi)$ $+ \int [\varphi(y) - \varphi(x)]\, \nu^0(x, dy)$	$b^\star = b^0 - \lambda^{-1} a\nabla V$ $\nu^\star(x, dy) = e^{(V_x - V_y)/\lambda}\nu^0(x, dy)$

horizons, and continuation estimates for rare-event control, with finite-horizon and discrete-chain extensions (Du et al., 2026). PI-VM instead builds its regression target directly from the temporal recursion of the optimal path-integral value. It averages likelihood-corrected exponential costs over short trajectory branches, bootstraps the remaining horizon with a detached continuation value, and reuses the resulting branches through replay. This yields a common value-learning objective for Euclidean and manifold diffusions, continuous-time Markov chains, and finite-activity jump processes, without requiring full-horizon simulation for every regression update.

3 Path-Integral Control of Markov Processes

We first identify the common value function and then show how it determines controlled dynamics.

3.1 A Common Path-Integral Value

Let $\mathbb{P}^0$ be the law of a continuous-time Markov process $(X_t)_{0\le t\le T}$ on a state space $\mathcal{X}$, and write $\mathbb{P}^0_{t,x}$ for its future law conditional on $X_t = x$. The state space may be Euclidean, discrete, or a manifold. For a running cost c and terminal cost g, define

$$\mathcal{A}_{t,s} := \int_t^s c(r, X_r)\, dr, \qquad C_{t:T} := \mathcal{A}_{t,T} + g(X_T). \tag{1}$$

The associated value is the minimum future cost regularized by path-space relative entropy:

$$V(t,x) := \inf_{\mathbb{P}\ll\mathbb{P}^0_{t,x}} \left\{\mathbb{E}_{\mathbb{P}}[C_{t:T}] + \lambda\,\mathrm{KL}(\mathbb{P}\|\mathbb{P}^0_{t,x})\right\}, \qquad \lambda > 0. \tag{2}$$

To express this optimum as an expectation, define the conditional partition function $Z(t,x) := \mathbb{E}^0_{t,x}[e^{-C_{t:T}/\lambda}]$. Assume finite path costs, $0 < Z(t,x) < \infty$, and $\mathbb{E}^0_{t,x}[|C_{t:T}|e^{-C_{t:T}/\lambda}] < \infty$. The Gibbs–Donsker–Varadhan principle (Deuschel & Stroock, 2001; Hartmann et al., 2017) then gives the value and unique optimal path law:

$$V(t,x) = -\lambda \log \mathbb{E}^0_{t,x}[e^{-C_{t:T}/\lambda}], \quad \mathbb{P}^\star_{t,x}(d\tau) = \frac{e^{-C_{t:T}(\tau)/\lambda}}{Z(t,x)}\,\mathbb{P}^0_{t,x}(d\tau), \tag{3}$$

The optimizer exponentially favors low-cost paths relative to the reference. This path-law characterization applies across Markov process classes; its simulation and local control representation depend on the reference dynamics. Appendix A.1 gives the variational identity and explains why the stated assumptions make the optimizer admissible.

3.2 From Values to Controlled Dynamics

Eq. 3 characterizes the optimal path law by a global reweighting. To simulate that law, we need a local description of how the process evolves. The Feynman–Kac formula and the Doob h-transform provide this connection, with the same function $Z = e^{-V/\lambda}$ linking the value and the dynamics.

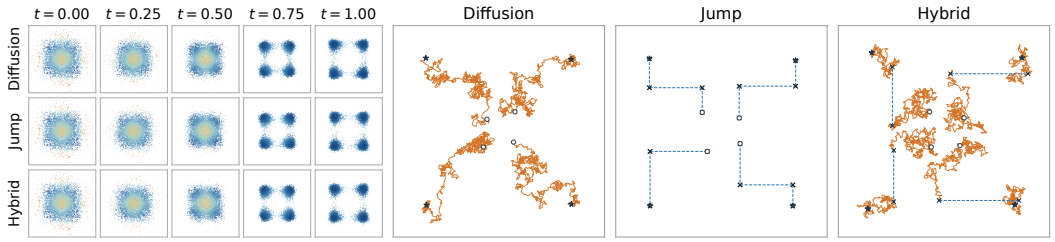

Figure 2: PI-VM-controlled dynamics under diffusion, jump, and hybrid (jump-diffusion) settings. Left: marginal distributions at selected time points. Right: trajectories learned by PI-VM.

We describe the reference through its infinitesimal generator. For a suitable test function φ,

$$(\mathcal{L}_t^0\varphi)(x) = \lim_{\varepsilon\downarrow 0} \frac{\mathbb{E}_{t,x}^0[\varphi(X_{t+\varepsilon})] - \varphi(x)}{\varepsilon}. \tag{4}$$

The generator describes the instantaneous change of an observable's conditional expectation. The calculations below assume the required generator-domain and time-regularity conditions.

The partition function in Eq. 3 is a Feynman–Kac expectation with terminal function $e^{-g/\lambda}$. Consequently, it satisfies the linear backward equation (Oksendal, 2013)

$$\partial_t Z(t,x) + (\mathcal{L}_t^0 Z(t,\cdot))(x) - \frac{c(t,x)}{\lambda} Z(t,x) = 0, \quad Z(T,x) = e^{-g(x)/\lambda}. \tag{5}$$

For $t < s \le T$, conditioning the Gibbs density on the segment through s gives

$$\frac{d\mathbb{P}^\star_{[t,s]}(\cdot \mid X_t = x)}{d\mathbb{P}^0_{[t,s]}(\cdot \mid X_t = x)} = e^{-\mathcal{A}_{t,s}/\lambda} \frac{Z(s,X_s)}{Z(t,x)}. \tag{6}$$

These densities form a mean-one martingale under the reference law and consistently define the generalized Doob h-transform with $h = Z$ (Chetrite & Touchette, 2015). At $s = T$, the density is exactly the Gibbs weight in Eq. 3. The transformed process therefore realizes the optimizer, including its intermediate-time transitions. Its generator is

$$\boxed{(\mathcal{L}_t^\star\varphi)(x) = \frac{\mathcal{L}_t^0(Z_t\varphi)(x) - \varphi(x)\mathcal{L}_t^0 Z_t(x)}{Z_t(x)}, \qquad Z_t = e^{-V(t,\cdot)/\lambda}.} \tag{7}$$

This is the standard Doob generator transform applied to the Gibbs optimizer (Léonard, 2011; Chetrite & Touchette, 2015). Appendix A.2 derives the transition law and generator of the optimal process. Thus the exact value determines the optimal generator for the fixed reference. Absolute continuity preserves the diffusion covariance and forbids jumps outside the reference support. The strictly positive Gibbs tilt also preserves the reference jump support. A learned value supplies a candidate control, subject to well-posedness, rather than an exact Gibbs law.

Table 1 gives the corresponding controls for the process families considered here. Here $a = \sigma\sigma^\top$ is the diffusion covariance. Time arguments are suppressed, $V_x = V(t,x)$, and H_i are tangent noise fields acting as directional derivatives. For manifold diffusions, the drift is in the Stratonovich representation. CTMCs rates are off-diagonal, with diagonal entries chosen so each row sums to zero. In each row, V is the value associated with that reference. We provide the complete derivation in Appendix A.3. Figure 2 illustrates the resulting controlled dynamics across diffusion, jump, and hybrid processes.

4 Path Integral Value Matching

We first derive a temporal recursion by truncating and marginalizing the path integral (Sec. 4.1). We then extend this recursion to off-policy sampling (Sec. 4.2) and characterize its variance, before presenting the PI-VM training algorithm (Sec. 4.3).

4.1 Value Function Temporal Recursion

Directly estimating Eq. 3 requires full-horizon simulation for each target and can produce high-variance weights. Instead, we truncate each trajectory at $s \in (t, T]$ and marginalize the remaining path. By the Markov property and cost additivity,

$$Z(t,x) = \mathbb{E}^0_{t,x}\left[e^{-\mathcal{A}_{t,s}/\lambda} Z(s, X_s)\right]. \tag{8}$$

Taking $-\lambda \log$ gives the temporal recursion

$$\boxed{V(t,x) = -\lambda \log \mathbb{E}^0_{t,x}\left[e^{-[\mathcal{A}_{t,s}+V(s,X_s)]/\lambda}\right], \qquad V(T,x) = g(x).} \tag{9}$$

The future after s is represented by $V(s, X_s)$, so only the short segment $[t, s]$ needs to be simulated. This recursion holds across the reference processes in Sec. 3 and provides the basis for TD learning. Its derivation is given in Appendix B.

Under the reference law, let $Y = e^{-C_{t:T}/\lambda}$ and $Y_s = e^{-[\mathcal{A}_{t,s}+V(s,X_s)]/\lambda}$. With the exact continuation, $Y_s = \mathbb{E}^0_{t,x}[Y \mid \mathcal{F}_s]$. If $Y \in L^2$, the law of total variance gives

$$\mathrm{Var}^0_{t,x}(Y) = \mathrm{Var}^0_{t,x}(Y_s) + \mathbb{E}^0_{t,x}[\mathrm{Var}^0_{t,x}(Y \mid \mathcal{F}_s)]. \tag{10}$$

Marginalization therefore removes future-path randomness from the exponential weight. This is an exact-continuation result, rather than a variance guarantee for a learned logarithmic target (Appendix D). The population recursion is also stable to continuation errors: a uniform error of at most δ changes the recursive target by at most δ (Appendix C).

4.2 Off-Policy Correction

The recursion need not be evaluated using reference trajectories. We may instead simulate short segments from a behavior process $\mathbb{P}^\mu$ and correct their likelihoods. For a fixed starting pair (t, x) and cutoff time s, assume $\mathbb{P}^0_{[t,s]}(\cdot \mid X_t = x) \ll \mathbb{P}^\mu_{[t,s]}(\cdot \mid X_t = x)$ and define

$$\ell_{t,s} := \log \frac{d\mathbb{P}^0_{[t,s]}(\cdot \mid X_t = x)}{d\mathbb{P}^\mu_{[t,s]}(\cdot \mid X_t = x)}. \tag{11}$$

Changing measure in Eq. 9 yields

$$\boxed{V(t,x) = -\lambda \log \mathbb{E}^\mu_{t,x}\left[e^{\ell_{t,s}-[\mathcal{A}_{t,s}+V(s,X_s)]/\lambda}\right].} \tag{12}$$

The factor $e^{\ell_{t,s}}$ corrects only the simulated segment and is one under reference sampling. For numerical simulation, the likelihood must match the actual transition law; discretization adds a separate approximation error (Appendix B.2).

Likelihood correction preserves the expectation, while the behavior process determines its simulation variance. To characterize this dependence, define the corrected weight with the exact continuation value:

$$M_{t,s} := \exp\left(\ell_{t,s} - \frac{\mathcal{A}_{t,s} + V(s, X_s)}{\lambda}\right). \tag{13}$$

Proposition 4.1 (Optimal simulation and variance)**.** *Fix (t, x) and $t < s \le T$, and assume the above change of measure is valid and $M_{t,s} \in L^2(\mathbb{P}^\mu_{t,x})$. Then*

$$\frac{\mathrm{Var}^\mu_{t,x}(M_{t,s})}{Z(t,x)^2} = \chi^2\left(\mathbb{P}^\star_{[t,s]} \| \mathbb{P}^\mu_{[t,s]}\right), \tag{14}$$

where both segment laws are conditional on $X_t = x$ and $\chi^2(P\|Q) = \mathbb{E}_Q[(dP/dQ - 1)^2]$. Under the finite-cost assumptions above, the optimal law is equivalent to the reference, so it is an admissible proposal and gives

$$M_{t,s} = Z(t,x) \quad \mathbb{P}^\star_{t,x}\text{-a.s.}, \qquad \mathrm{Var}^\star_{t,x}(M_{t,s}) = 0. \tag{15}$$

Proof. Combining the likelihood ratio with the Doob density in Eq. 6 gives

$$\frac{M_{t,s}}{Z(t,x)} = \frac{d\mathbb{P}^{\star}_{[t,s]}}{d\mathbb{P}^{\mu}_{[t,s]}}. \tag{16}$$

This ratio has unit mean under $\mathbb{P}^{\mu}$; its centered second moment gives Eq. 14. Under optimal simulation, the ratio is identically one. □

Thus, the optimal policy is a zero-variance proposal for the complete path-integral weight with the exact continuation, not for the likelihood ratio alone. A learned value-guided process need not reduce the variance; practical targets also depend on continuation accuracy and finite simulation. Further variance analysis is given in Appendix D.

4.3 Algorithm Implementation

PI-VM approximates V with a scalar value network V_θ and learns the off-policy recursion from replayed short path branches. The replay buffer stores the branch data used to construct the PI-TD target; branches are generated when the buffer is refreshed, rather than resimulated at every gradient update.

Branch collection and replay. We first roll out anchor trajectories under the reference process or a valid value-guided behavior process. For a selected state–time pair (t, x) on a rollout, called an anchor, choose a cutoff $s \in (t, T]$ and simulate N conditionally independent branches under the behavior law $\mathbb{P}^{\mu}$ from the same anchor to the same cutoff. A replay entry stores

$$\xi = \left(t, x, s, \left\{\tau^{(j)}_{[t,s]}, X^{(j)}_s, \mathcal{A}^{(j)}_{t,s}, \ell^{(j)}_{t,s}\right\}_{j=1}^{N}\right), \qquad \xi \in \mathcal{D}, \tag{17}$$

where $\tau^{(j)}_{[t,s]}$ denotes the sampled branch trajectory, $\mathcal{A}^{(j)}_{t,s}$ its accumulated running cost, and $\ell^{(j)}_{t,s}$ the log likelihood correction associated with the behavior law that actually generated that branch. In implementation, $\mathcal{A}^{(j)}_{t,s}$ and $\ell^{(j)}_{t,s}$ may be cached when the branch is generated, or reconstructed from the stored trajectory and its behavior-process metadata. Thus replay reuses the same stochastic branch realization across multiple optimization steps.

On a time grid, each branch spans at most M simulator steps, with $s = \min\{t + M\Delta t, T\}$. Event-driven simulators instead terminate at a chosen time cutoff. Since historical branches can have been generated by earlier behavior processes, the likelihood term must always correspond to the process that generated the stored branch, not the current guided process.

Replay target and value update. At each optimization step, sample a mini-batch $\{\xi_b\}_{b=1}^{B}$ from $\mathcal{D}$. No new branch simulation is required for these updates. Let $\widetilde{V}$ be the detached online network or its EMA copy, as specified for each experiment. The continuation uses the known terminal cost at T and $\widetilde{V}$ otherwise:

$$G^{(b,j)} = \begin{cases} g(X^{(b,j)}_T), & s_b = T, \\ \widetilde{V}(s_b, X^{(b,j)}_{s_b}), & s_b < T. \end{cases} \tag{18}$$

The replayed path-integral temporal-difference target is

$$\boxed{\widehat{V}(t_b, x_b) = -\lambda \log\left[\frac{1}{N}\sum_{j=1}^{N} \exp\left(\ell^{(b,j)}_{t_b,s_b} - \frac{\mathcal{A}^{(b,j)}_{t_b,s_b} + G^{(b,j)}}{\lambda}\right)\right].} \tag{19}$$

The target averages exponential costs separately for each anchor, using log-sum-exp for numerical stability. Branch data stay fixed while continuation values change. With a fixed continuation and fresh independent branches, the unlogged average is unbiased, but its negative logarithm is generally upward biased. Learned continuations and repeated fitting to replay introduce additional errors (Appendix D.4).

PI-VM then minimizes

$$\boxed{\mathcal{L}_{\text{PI-VM}}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\left[V_\theta(t_b, x_b) - \mathrm{sg}[\widehat{V}(t_b, x_b)]\right]^2.} \tag{20}$$

Table 2: Control error $\mathcal{L}_2$ and per-update time on diffusion benchmarks.

	OU Linear		OU Quadratic (easy)		OU Quadratic (hard)	
Method	Control $\mathcal{L}_2$ (↓)	Time (↓)	Control $\mathcal{L}_2$ (↓)	Time (↓)	Control $\mathcal{L}_2$ (↓)	Time (↓)
RE	0.00003 ± 0.00001	87 ms	0.00019 ± 0.00002	54 ms	0.00453 ± 0.00039	50 ms
CE	0.00056 ± 0.00040	41 ms	0.00215 ± 0.00022	22 ms	1.46971 ± 0.01363	21 ms
VAR	0.00032 ± 0.00018	38 ms	0.09234 ± 0.00065	22 ms	1.46971 ± 0.01363	21 ms
LVAR	0.00017 ± 0.00006	41 ms	0.00061 ± 0.00004	22 ms	0.01521 ± 0.00103	21 ms
REINFORCE	0.00097 ± 0.00039	50 ms	0.01615 ± 0.00041	28 ms	0.20843 ± 0.02053	27 ms
SOCM	$\mathbf{0.00000 \pm 0.00000}$	128 ms	0.00044 ± 0.00001	183 ms	1.46971 ± 0.01363	194 ms
SOCM-A	$\mathbf{0.00000 \pm 0.00000}$	41 ms	0.00073 ± 0.00006	21 ms	1.46971 ± 0.01363	21 ms
AM	$\mathbf{0.00000 \pm 0.00000}$	41 ms	0.00028 ± 0.00002	21 ms	0.00674 ± 0.00042	21 ms
VM	0.00035 ± 0.00004	90 ms	0.00099 ± 0.00012	45 ms	0.01180 ± 0.00264	46 ms
PI-VM (Ours)	$\mathbf{0.00000 \pm 0.00000}$	**8** ms	$\mathbf{0.00010 \pm 0.00001}$	**7** ms	$\mathbf{0.00078 \pm 0.00006}$	**7** ms

Table 3: Scalability analysis on the Quadratic OU Easy task with increasing dimensions.

	SOCM		AM		VM		PI-VM (Ours)	
Dim	Control $\mathcal{L}_2$ (↓)	Time (↓)	Control $\mathcal{L}_2$ (↓)	Time (↓)	Control $\mathcal{L}_2$ (↓)	Time (↓)	Control $\mathcal{L}_2$ (↓)	Time (↓)
10	0.00027 ± 0.00001	130 ms	0.00026 ± 0.00001	29 ms	0.00058 ± 0.00004	63 ms	$\mathbf{0.00006 \pm 0.00000}$	**7** ms
20	0.00044 ± 0.00001	182 ms	0.00027 ± 0.00003	29 ms	0.00099 ± 0.00012	45 ms	$\mathbf{0.00011 \pm 0.00001}$	**8** ms
80	Out of Memory		0.02960 ± 0.00028	28 ms	0.00354 ± 0.00039	48 ms	$\mathbf{0.00044 \pm 0.00004}$	**12** ms
150	Out of Memory		0.07620 ± 0.00054	32 ms	0.00899 ± 0.00037	50 ms	$\mathbf{0.00080 \pm 0.00008}$	**18** ms
200	Out of Memory		0.08983 ± 0.00023	38 ms	0.02290 ± 0.00052	52 ms	$\mathbf{0.00256 \pm 0.00062}$	**21** ms

Here sg denotes stop-gradient: only the anchor prediction is differentiated. Refreshing $\mathcal{D}$ collects branches under updated behavior processes. Stored likelihoods correct the conditional branch law, not the anchor distribution, so adequate state–time coverage remains necessary. The OU and image diffusion runs use EMA continuations; the Jump, CTMCs, and manifold benchmarks use the online network. These variants and the image experiments' time weights are specified in the appendix. Algorithm 1 summarizes the basic replay procedure.

5 Experiments

We evaluate control accuracy, reward–diversity trade-offs, and training cost across Markov processes and image generators, with short-trajectory ablations. Settings and budgets are in Appendices F–K.

5.1 Benchmarks across Markov Processes

We consider linear and quadratic Ornstein–Uhlenbeck (OU) diffusions (Domingo i Enrich et al., 2024), continuous-state jumps, finite-state CTMCs, and manifold diffusions. Quadratic OU tests state-dependent feedback up to $d = 200$. We measure dimension-normalized squared control error $\mathcal{L}_2$ against optimal dynamics; for jumps and CTMCs, this measures induced-drift error.

Baselines. For OU, we include four trajectory-objective baselines: relative entropy (RE), cross entropy (CE) (Holdijk et al., 2023), variance (VAR), and log-variance (LVAR) (Nüsken & Richter, 2021). REINFORCE (Williams, 1992) uses score-function policy gradients. Stochastic Optimal Control Matching (SOCM) and its adjoint variant (SOCM-A) (Domingo i Enrich et al., 2024) learn controls through stochastic regression, while Adjoint Matching (AM) (Domingo-Enrich et al., 2025) uses adjoint-based control targets. Value Matching (VM) (Jensen et al., 2026) instead learns a scalar value function from sampled costs-to-go. The dimensional-scaling comparison includes SOCM, AM, and VM. For jumps and CTMCs, we use Discrete Adjoint Matching (DAM) (So et al., 2026); for manifolds, we use Hard-SOC (Mosso et al., 2026).

Euclidean diffusions. PI-VM matches the best Linear OU error at the displayed precision and has the lowest error on both Quadratic OU tasks (Table 2). On the hard task, it attains 0.00078, versus 0.00453 for RE, 0.00674 for AM, and 0.01180 for VM. At $d = 200$, it attains 0.00256, versus 0.08983 for AM and 0.02290 for VM (Table 3). SOCM runs out of memory from $d = 80$ in the tested configuration. PI-VM has the lowest reported update time across dimensions, reaching 21 ms at $d = 200$. Appendix F details method-specific budgets, auxiliary training, and hardware.

Table 4: Performance analysis on jump process and CMTC benchmarks with control $\mathcal{L}_2(\downarrow)$.

METHOD	PARTIAL (Jump)	REFRESH (Jump)	PARTIAL (CTMC)	REFRESH (CTMC)
DAM	0.000005 ± 0.000000	0.000070 ± 0.000025	0.000006 ± 0.000001	0.000138 ± 0.000116
PI-VM (OURS)	$\mathbf{0.000000 \pm 0.000000}$	$\mathbf{0.000014 \pm 0.000004}$	$\mathbf{0.000005 \pm 0.000000}$	$\mathbf{0.000090 \pm 0.000043}$

Table 5: Performance analysis on manifold diffusion benchmarks with control $\mathcal{L}_2(\downarrow)$.

METHOD	SPHERE $\mathbb{S}^2$	STIEFEL $\mathrm{St}(4,2)$, $\beta = 0.01$	STIEFEL $\mathrm{St}(4,2)$, $\beta = 0.5$
HARD-SOC	0.549290 ± 0.036150	0.000060 ± 0.000005	0.072907 ± 0.010727
PI-VM (OURS)	$\mathbf{0.016224 \pm 0.003147}$	$\mathbf{0.000001 \pm 0.000000}$	$\mathbf{0.003151 \pm 0.000167}$

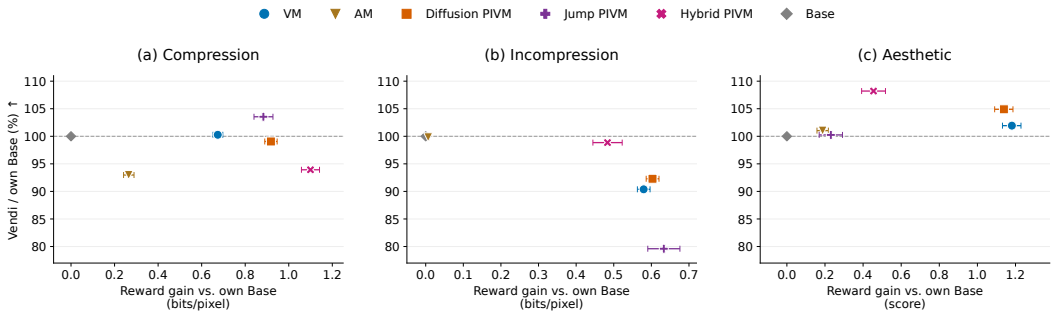

Figure 3: CIFAR-10 reward gain and Vendi diversity relative to each process's base generator.

Jumps and CTMCs. PI-VM has lower mean induced-drift error than DAM on all four tasks (Table 4). Refresh-jump error falls from 0.000070 to 0.000014, while PI-VM also achieves lower error on Partial CTMC (0.000005 versus 0.000006).

Manifolds diffusions. PI-VM outperforms Hard-SOC on all three manifold benchmarks (Table 5). On the sphere, the control $\mathcal{L}_2$ error decreases from 0.549290 to 0.016224. On Stiefel, it decreases from 0.000060 to 0.000001 for $\beta = 0.01$, and from 0.072907 to 0.003151 in the more challenging $\beta = 0.5$ setting.

5.2 REWARD-GUIDED IMAGE GENERATION

We adapt frozen CIFAR-10 (Krizhevsky et al., 2009) and DiT (Peebles & Xie, 2023) generators for compression (Black et al., 2024), incompression (Black et al., 2024), and aesthetics (Radford et al., 2021), favoring smaller JPEG files, larger files, and higher learned aesthetic scores, respectively. We report reward and Vendi diversity (Friedman & Dieng, 2022) against VM (Jensen et al., 2026). CIFAR also includes AM (Domingo-Enrich et al., 2025) and jump/hybrid PI-VM.

CIFAR-10. As shown in Figure 3, diffusion PI-VM achieves higher reward gains than VM on both JPEG tasks, while attaining comparable aesthetic reward with higher Vendi diversity. The jump and hybrid variants further offer distinct reward–diversity trade-offs. Relative to their respective base generators, hybrid PI-VM achieves the largest compression gain and the highest relative Vendi score on aesthetics, whereas jump PI-VM achieves the largest incompression gain with a greater reduction in diversity. These complementary outcomes highlight the flexibility of PI-VM: the same value-learning objective supports different reference processes with distinct generation profiles beyond diffusion.

DiT. PI-VM achieves higher rewards than VM on all three tasks (Table 6). On both JPEG tasks, it simultaneously lowers the control cost and the regularized objective J, improving the overall control objective rather than reward alone. On aesthetics, PI-VM obtains a higher reward with a higher control cost and J. Figure 4 shows selected paired samples. Overall, these experiments demonstrate

Table 6: Image reward fine-tuning on DiT-XL/2 (CFG 2) for Compression, Incompression, and Aesthetic objectives. Metrics are Reward (↑), Control cost (↓), and J (↓), from left to right.

METHOD	COMPRESSION ($\eta = 25$)			INCOMPRESSION ($\eta = 25$)			AESTHETIC ($\eta = 10$)		
BASE	-2.019	0.000	50.482	2.019	0.000	-50.482	5.159	0.000	-51.588
VM	-0.660	32.603	49.112	4.510	38.509	-74.234	5.291	**4.088**	**-48.820**
PI-VM (OURS)	**-0.438**	**23.611**	**34.564**	**4.642**	**37.740**	**-78.300**	**5.428**	13.053	-41.227



Figure 4: DiT samples for compression, incompression, and aesthetics (left to right).

stronger reward-guided adaptation with PI-VM than with VM, while the jump and hybrid variants provide complementary choices for balancing reward and diversity.

5.3 ABLATION STUDIES

We study branch count N, maximum branch length M, network depth, and anchor/branch allocation in Appendix J. The default $(N, M) = (8, 8)$ attains an error of 0.00010 at 7 ms per update; increasing both to 16 yields 0.00009 at 18 ms (Table 11). Deeper networks and more branches help early training, with little advantage at later checkpoints (Table 12 and 14). At fixed BN, more distinct anchors lower error (Table 13).

6 CONCLUSION

PI-VM provides a value-based approach to KL-regularized stochastic control across a broad class of Markov processes. By combining short-segment path-integral targets with continuation estimates, it reduces control learning to a common scalar regression objective, while leaving only simulation and the conversion from values to controlled dynamics process specific. Across Euclidean and manifold diffusions, continuous-state jump processes, CTMCs, and jump-diffusions, PI-VM learns controls or induced dynamics that closely match ground-truth optima. The same framework also extends to reward-guided image generation with pretrained generative models. Together, these results show that local path-integral value matching can serve as a common learning principle across substantially different stochastic control settings.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design, assist with method implementation, assist with translation, and support the interpretation of experimental results.

We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, propose or refine hypotheses, clean or reformat datasets, or conduct qualitative or thematic data analysis.

Additionally, we used generative AI tools to assist with code editing, manuscript drafting and revision, literature search and summarization, and improving readability and presentation.

All AI-assisted outputs were reviewed, verified, and revised by the authors. Code and experimental suggestions were validated through implementation and experiments, and suggested references were checked against the original sources. We take responsibility for the final content of this work.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024.

Denis Blessing, Julius Berner, Lorenz Richter, Carles Domingo-Enrich, Yuanqi Du, Arash Vahdat, and Gerhard Neumann. Trust region constrained measure transport in path space for stochastic optimal control and inference. *arXiv preprint arXiv:2508.12511*, 2025.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. Stochastic control liaisons: Richard sinkhorn meets gaspard monge on a schrodinger bridge. *Siam Review*, 63(2):249–313, 2021.

Raphaël Chetrite and Hugo Touchette. Nonequilibrium markov processes conditioned on large deviations. In *Annales Henri Poincaré*, volume 16, pp. 2005–2057. Springer, 2015.

Jean-Dominique Deuschel and Daniel W Stroock. *Large deviations*, volume 342. American Mathematical Soc., 2001.

Carles Domingo-Enrich. A taxonomy of loss functions for stochastic optimal control. *arXiv preprint arXiv:2410.00345*, 2024.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xQBRrtQM8u.

Carles Domingo i Enrich, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky TQ Chen. Stochastic optimal control matching. *Advances in Neural Information Processing Systems*, 37:112459–112504, 2024.

Yuanqi Du, Jiajun He, Dinghuai Zhang, Eric Vanden-Eijnden, and Carles Domingo-Enrich. Rare event analysis via stochastic optimal control. *arXiv preprint arXiv:2604.13213*, 2026.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *arXiv preprint arXiv:2210.02410*, 2022.

Frédéric Bonnans, J., Ottenwaelter, Élisabeth, and Zidani, Housnaa. A fast algorithm for the two dimensional hjb equation of stochastic control. *ESAIM: M2AN*, 38(4):723–735, 2004. doi: 10.1051/m2an:2004034. URL https://doi.org/10.1051/m2an:2004034.

Carsten Hartmann, Lorenz Richter, Christof Schütte, and Wei Zhang. Variational characterization of free energy: Theory and algorithms. *Entropy*, 19(11):626, 2017.

Aaron J Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Daniel S. Levine, Brandon M Wood, Bin Hu, Brandon Amos, Brian Karrer, Xiang Fu, Guan-Horng Liu, and Ricky T. Q. Chen. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6Eg1OrHmg2.

Lars Holdijk, Yuanqi Du, Ferry Hooft, Priyank Jaini, Berend Ensing, and Max Welling. Stochastic optimal control for collective variable free sampling of molecular transition paths. *Advances in Neural Information Processing Systems*, 36:79540–79556, 2023.

Cristian Jensen, Luca Schaufelberger, Riccardo De Santi, Kjell Jorner, and Andreas Krause. Value matching: Scalable and gradient-free reward-guided flow adaptation. In *International Conference on Learning Representations*, volume 2026, pp. 144926–144958, 2026.

Max Jensen and Iain Smears. On the convergence of finite element methods for hamilton–jacobi–bellman equations. *SIAM Journal on Numerical Analysis*, 51(1):137–162, 2013. doi: 10.1137/110856198. URL https://doi.org/10.1137/110856198.

Hilbert J Kappen. Linear theory for control of nonlinear stochastic systems. *Physical review letters*, 95(20):200201, 2005.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Hilbert Johan Kappen and Hans Christian Ruiz. Adaptive importance sampling for control and inference. *Journal of Statistical Physics*, 162(5):1244–1266, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Christian Léonard. Stochastic derivatives and generalized h-transforms of markov processes. *arXiv preprint arXiv:1102.3172*, 2011.

Guan-Horng Liu, Jaemoo Choi, Yongxin Chen, Benjamin Kurt Miller, and Ricky TQ Chen. Adjoint schr\" odinger bridge sampler. *arXiv preprint arXiv:2506.22565*, 2025.

Zhen Liu, Tim Xiao, Carles Domingo i Enrich, Weiyang Liu, and Dinghuai Zhang. Value gradient guidance for flow matching alignment. *Advances in Neural Information Processing Systems*, 38: 11372–11404, 2026.

Mattia Mosso, Jaemoo Choi, and Heng Yang. Hard-constrained sampling on embedded riemannian manifolds via adjoint schr\"{o} dinger bridges. *arXiv preprint arXiv:2608.25838*, 2026.

Nikolas Nüsken and Lorenz Richter. Solving high-dimensional hamilton–jacobi–bellman pdes using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2(4):48, 2021.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Oswin So, Brian Karrer, Chuchu Fan, Ricky TQ Chen, and Guan-Horng Liu. Discrete adjoint matching. *arXiv preprint arXiv:2602.07132*, 2026.

Sep Thijssen and HJ Kappen. Path integral control and state-dependent feedback. *Physical Review E*, 91(3):032104, 2015.

Emanuel Todorov. Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19, 2006.

Christopher von Klitzing, Denis Blessing, Henrik Schopmans, Pascal Friederich, and Gerhard Neumann. Learning boltzmann generators via constrained mass transport. *arXiv preprint arXiv:2510.18460*, 2025.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

A Path-Space Optimum and Value-Induced Dynamics

We use the notation of the main text, with an arbitrary horizon T and KL coefficient $\lambda > 0$. All path laws below are conditional on a fixed starting pair (t, x) unless otherwise stated. We assume a Markov reference with regular conditional laws and measurable additive costs that are finite along reference paths. At every starting pair considered, assume $0 < Z(t, x) < \infty$ and $\mathbb{E}^0_{t,x}[|C_{t:T}|e^{-C_{t:T}/\lambda}] < \infty$. The variational problem is over laws with finite cost expectation and relative entropy. Generator calculations additionally require Z and the test functions to lie in the relevant domains and have the stated time regularity.

A.1 Gibbs Variational Identity

Let $C = C_{t:T}$ and define $d\mathbb{P}^\star = Z(t, x)^{-1}e^{-C/\lambda}d\mathbb{P}^0$. The density is strictly positive, so the two laws are equivalent. Moreover, $\mathbb{E}_{\mathbb{P}^\star}|C| = Z(t, x)^{-1}\mathbb{E}^0[|C|e^{-C/\lambda}] < \infty$; the log density $-C/\lambda - \log Z(t, x)$ is therefore integrable under $\mathbb{P}^\star$, giving finite relative entropy. Thus the proposed optimizer belongs to the admissible class. For an admissible $\mathbb{P} \ll \mathbb{P}^0$ for which the terms are finite,

$$\lambda \,\mathrm{KL}(\mathbb{P}\|\mathbb{P}^\star) = \lambda\,\mathbb{E}_{\mathbb{P}}\left[\log \frac{d\mathbb{P}}{d\mathbb{P}^0} + \frac{C}{\lambda} + \log Z(t, x)\right] \tag{21}$$

$$= \mathbb{E}_{\mathbb{P}}[C] + \lambda\,\mathrm{KL}(\mathbb{P}\|\mathbb{P}^0) + \lambda \log Z(t, x). \tag{22}$$

Nonnegativity of relative entropy proves equation 3; equality holds only at $\mathbb{P}^\star$. This is a variational statement about path measures and uses no process-specific control parameterization.

If the initial distribution ρ_0 is prescribed, the optimization is performed conditionally at each initial state and then averaged over ρ_0. Its optimal conditional likelihood is $e^{-C_{0:T}/\lambda}/Z(0, X_0)$. A single global normalization of $e^{-C_{0:T}/\lambda}$ generally changes the initial marginal and therefore solves a different problem. The conditional formulation in the main text keeps the starting state fixed.

A.2 The Doob Density and Its Generator

Let $\mathcal{F}_s$ be the path history from t through s. The Markov property and cost additivity give

$$D_{t,s} := \mathbb{E}^0_{t,x}\left[\frac{e^{-C_{t:T}/\lambda}}{Z(t, x)}\,\middle|\,\mathcal{F}_s\right] = e^{-\mathcal{A}_{t,s}/\lambda}\frac{Z(s, X_s)}{Z(t, x)}. \tag{23}$$

Thus $D_{t,s}$ is a mean-one martingale and defines consistent segment laws. For a test function φ, the optimal transition satisfies

$$\mathbb{E}^\star_{t,x}[\varphi(X_s)] = \frac{1}{Z(t, x)}\mathbb{E}^0_{t,x}\left[e^{-\mathcal{A}_{t,s}/\lambda}Z(s, X_s)\varphi(X_s)\right]. \tag{24}$$

The right-hand side depends on the past only through (t, x), so the transformed law is Markov. Taking $s = t + h$, its short-time expansion is

$$\mathbb{E}^0_{t,x}[e^{-\mathcal{A}_{t,t+h}/\lambda}Z(t + h, X_{t+h})\varphi(X_{t+h})] = Z_t(x)\varphi(x) \tag{25}$$

$$+ h\left[\mathcal{L}^0_t(Z_t\varphi) + \varphi\,\partial_t Z_t - \frac{c}{\lambda}Z_t\varphi\right](x) + o(h). \tag{26}$$

Substituting the Feynman–Kac equation $\partial_t Z_t - cZ_t/\lambda = -\mathcal{L}^0_t Z_t$ gives

$$\mathcal{L}^\star_t\varphi = Z_t^{-1}\left[\mathcal{L}^0_t(Z_t\varphi) - \varphi\mathcal{L}^0_t Z_t\right]. \tag{27}$$

The subtraction ensures $\mathcal{L}^\star_t\mathbf{1} = 0$. The running cost affects the transition law through Z; it does not remain as a killing term in the controlled generator.

A.3 Specialization to the Process Families

Euclidean diffusions. For $\mathcal{L}^0\varphi = b^0 \cdot \nabla\varphi + \frac{1}{2}\,\mathrm{tr}(a\nabla^2\varphi)$,

$$\mathcal{L}^0(Z\varphi) = Z\mathcal{L}^0\varphi + \varphi\mathcal{L}^0 Z + (\nabla Z)^\top a\nabla\varphi.$$

Hence $b^\star = b^0 + a\nabla \log Z = b^0 - a\nabla V/\lambda$. For the matched representation $dX = (b^0 + \sigma u)dt + \sigma dW$ with $a = \sigma\sigma^\top$, the minimum-norm noise-space control is $u^\star = -\sigma^\top \nabla V/\lambda$. When Girsanov's conditions hold, its path-KL penalty is $\frac{\lambda}{2}\mathbb{E}^u \int \|u\|^2 dt$ for an identifiable noise-space control. If σ has a null space, adding components in that null space leaves the state dynamics unchanged and the minimum-norm representation must be used for this identification.

Manifold diffusions. Write the Stratonovich generator as $\mathcal{L}^0 = b^0 + \frac{1}{2}\sum_i H_i^2$, with tangent vector fields b^0, H_i acting on scalar functions. The product rule gives

$$\mathcal{L}^0(Z\varphi) - \varphi\mathcal{L}^0 Z = Z\mathcal{L}^0\varphi + \sum_i (H_i Z)(H_i\varphi).$$

Therefore the added tangent drift is $-\lambda^{-1}\sum_i (H_i V)H_i$. For isotropic noise of scale σ this becomes $-\sigma^2 \operatorname{grad}_{\mathcal{M}} V/\lambda$; if the controlled drift is written as $b^0 + \sigma u$, then $u^\star = -\sigma \operatorname{grad}_{\mathcal{M}} V/\lambda$. No extrinsic normal component is introduced.

CTMCs and finite-activity jumps. For $\mathcal{L}^0\varphi(x) = \int [\varphi(y) - \varphi(x)]\nu^0(x, dy)$,

$$\mathcal{L}^0(Z\varphi)(x) - \varphi(x)\mathcal{L}^0 Z(x) = \int Z(y)[\varphi(y) - \varphi(x)]\nu^0(x, dy), \tag{28}$$

$$\nu^\star(x, dy) = \frac{Z(y)}{Z(x)}\nu^0(x, dy) = e^{[V(x)-V(y)]/\lambda}\nu^0(x, dy). \tag{29}$$

A finite-state CTMC is obtained by replacing the integral by a sum: $q^\star_{ij} = q^0_{ij} e^{(V_i - V_j)/\lambda}$ for $i \neq j$, with $q^\star_{ii} = -\sum_{j\neq i} q^\star_{ij}$. These formulas preserve the reference jump support. Finite reference activity alone does not guarantee that an arbitrary learned exponential tilting is nonexplosive; the constructed rates must remain admissible. Likewise, substituting a learned $\widehat{Z} = e^{-\widehat{V}/\lambda}$ into the segment expression does not generally give a normalized density: $\widehat{Z}$ need not satisfy the Feynman–Kac recursion. Behavior likelihoods must be computed from the actual simulated law.

Jump–diffusions. For a finite-activity generator $\mathcal{L}^0 = \mathcal{L}^0_{\text{diff}} + \mathcal{L}^0_{\text{jump}}$, the transform is linear in the generator. The diffusion drift and jump kernel therefore change by the formulas above, using the *same* value of the composite reference problem. Values learned for separate reference processes cannot in general be combined to obtain this value.

B Recursive Value Evaluation and Likelihood Correction

B.1 Exact Conditional Recursion

For $t < s \leq T$, split $C_{t:T} = \mathcal{A}_{t,s} + C_{s:T}$. Since $\mathcal{A}_{t,s}$ is $\mathcal{F}_s$-measurable,

$$Z(t, x) = \mathbb{E}^0_{t,x}\left[e^{-\mathcal{A}_{t,s}/\lambda}\mathbb{E}^0_{t,x}[e^{-C_{s:T}/\lambda} \mid \mathcal{F}_s]\right] \tag{30}$$

$$= \mathbb{E}^0_{t,x}\left[e^{-\mathcal{A}_{t,s}/\lambda} Z(s, X_s)\right]. \tag{31}$$

The first step is the tower property and the second is the Markov property. Taking $-\lambda \log$ proves equation 9. Changing measure on the segment proves equation 12. The expectation identity is exact. A Monte Carlo average is unbiased for its positive *unlogged* expectation when the continuation is exact; the logarithm generally destroys unbiasedness.

B.2 Likelihoods for Simulated Segments

Matched diffusions. Under the behavior law, let $dX_r = (b^0_r + \sigma_r u_r)dr + \sigma_r dW^u_r$. Assume Girsanov's absolute-continuity and martingale conditions, for example an applicable Novikov condition. For a nondegenerate diffusion, the state-path reference-over-behavior log likelihood is

$$\ell_{t,s} = -\int_t^s u_r^\top dW^u_r - \frac{1}{2}\int_t^s \|u_r\|^2 dr. \tag{32}$$

Substitution into equation 12 yields

$$V(t,x) = -\lambda \log \mathbb{E}^u_{t,x}\left[\exp\left(-\frac{\mathcal{A}_{t,s} + V(s,X_s)}{\lambda} - \int_t^s u_r^\top dW_r^u - \frac{1}{2}\int_t^s \|u_r\|^2 dr\right)\right]. \tag{33}$$

With redundant noise coordinates, the same Brownian likelihood can be used on a common augmented noise trace under the corresponding change-of-measure assumptions; it should not automatically be identified with the likelihood of the state path. Tangent-noise shifts on manifolds admit the same construction when the intrinsic SDE and its change of measure are well defined.

Finite-activity jump processes. Suppose the reference and behavior jump kernels have densities $k_r^0(x,y)$ and $k_r^\mu(x,y)$ with respect to a common measure. Let $a_r^v(x) = \int k_r^v(x,y)dy$ for $v \in \{0,\mu\}$. For a segment with jump times τ_k, the ratio of the jump-event factors and survival factors gives

$$\ell_{t,s} = \sum_{\tau_k \in (t,s]} \log \frac{k^0_{\tau_k}(X_{\tau_k-}, X_{\tau_k})}{k^\mu_{\tau_k}(X_{\tau_k-}, X_{\tau_k})} + \int_t^s [a_r^\mu(X_{r-}) - a_r^0(X_{r-})]dr. \tag{34}$$

For CTMCs, use the corresponding off-diagonal rates and their sums. The behavior must cover every reference transition that contributes to the expectation, and the resulting processes and likelihoods must be well defined. The holding-time integral is essential when total rates change; a product of event-rate ratios alone is insufficient.

Numerical simulators and replay. For a grid simulator with transition densities p_k^0, p_k^μ, the exact likelihood of the simulated chain is

$$\ell = \sum_k \log \frac{p_k^0(X_{k+1} \mid X_k)}{p_k^\mu(X_{k+1} \mid X_k)}. \tag{35}$$

Alternatively, likelihoods may be defined on a shared simulator-trace space if costs and endpoints are trace-measurable. The recursion then evaluates the numerical reference and cost approximation; agreement with the intended continuous-time problem also requires discretization control. For a value-guided proposal, the likelihood must correspond to the actual transitions after any rate clipping or numerical modification.

The main algorithm retains groups of branches from a common anchor and cutoff, together with their original behavior likelihoods. Branches are generated during buffer collection or refresh and reused between refreshes. Changing the value network does not change which behavior generated a stored segment. Conditional on a fixed continuation, the stored weights retain the same change-of-measure interpretation; repeatedly fitting to those data creates statistical dependence and is not equivalent to fresh independent Monte Carlo estimates at every update.

C Bootstrap Stability and Ideal Backward Propagation

Define the population operator on a continuation function W by

$$(\mathcal{T}_{t,s}W)(x) := -\lambda \log \mathbb{E}^0_{t,x}\left[e^{-[\mathcal{A}_{t,s} + W(X_s)]/\lambda}\right], \tag{36}$$

whenever the expectation is finite and positive. The same operator is obtained from a support-compatible behavior law with exact likelihoods.

Proposition C.1 (Stability to continuation error)**.** *If $|W(y) - U(y)| \le \delta$ on the relevant endpoint support, then $|\mathcal{T}_{t,s}W(x) - \mathcal{T}_{t,s}U(x)| \le \delta$.*

Proof. The bound on $W - U$ implies

$$e^{-\delta/\lambda}e^{-U/\lambda} \le e^{-W/\lambda} \le e^{\delta/\lambda}e^{-U/\lambda}.$$

Multiply by $e^{-\mathcal{A}_{t,s}/\lambda}$, take expectations, and apply $-\lambda \log$. The two target values differ by at most δ. □

The operator is nonexpansive, not generally a strict contraction: $\mathcal{T}_{t,s}(U + a) = \mathcal{T}_{t,s}U + a$ for a constant a. This distinction prevents a contraction argument from being transferred without qualification to neural value regression. It also does not control spatial derivatives: in one dimension, $\widehat{V}(x) = V(x) + \varepsilon \sin(x/\varepsilon^2)$ has value error at most ε but derivative error as large as $1/\varepsilon$. Diffusion-control error therefore requires additional derivative control.

Proposition C.2 (Exact propagation on a finite grid)**.** *Let $0 = t_0 < \cdots < t_K = T$. Set $V_m(t_K, \cdot) = g$ for all m and initialize arbitrary interior functions whose ensuing expectations are finite and positive. Suppose every update is evaluated exactly:*

$$V_{m+1}(t_k, \cdot) = \mathcal{T}_{t_k, t_{k+1}} V_m(t_{k+1}, \cdot), \qquad k < K.$$

Then $V_m(t_k, \cdot) = V(t_k, \cdot)$ whenever $m \geq K - k$.

Proof. The terminal value is exact at initialization. After one update the value at t_{K-1} is exact by the recursion. If the values at the last m grid intervals are exact, the next update makes the preceding interval exact as well. Induction proves the claim. □

This finite-horizon statement needs no strictly positive lower bound on the running cost. It applies to exact expectations and unrestricted function updates. Finite branching, restricted network classes, incomplete state coverage, and gradient-based optimization introduce additional errors and are not covered by the proposition.

For approximate backward evaluation, let $\widehat{V}_k$ denote the approximation at t_k, with $\widehat{V}_K = g$, and assume the operators below are well defined. Set $e_k := \|\widehat{V}_k - V(t_k, \cdot)\|_\infty$, where the supremum norm is over the state space. If the one-step residual satisfies

$$\|\widehat{V}_k - \mathcal{T}_{t_k, t_{k+1}} \widehat{V}_{k+1}\|_\infty \leq \varepsilon_k < \infty, \qquad k < K,$$

then the triangle inequality and Proposition C.1 give

$$e_k \leq \varepsilon_k + e_{k+1}, \qquad e_k \leq \sum_{j=k}^{K-1} \varepsilon_j,$$

since $e_K = 0$. This is an error-propagation bound under a stipulated uniform local error, not a claim that neural training attains such a bound.

D Variance and Finite-Branch Estimation

D.1 Marginalizing Future-Path Randomness

Under $\mathbb{P}^0_{t,x}$, let $Y = e^{-C_{t:T}/\lambda} \in L^2$ and $Y_s = e^{-\mathcal{A}_{t,s}/\lambda} Z(s, X_s)$. The Markov property gives $Y_s = \mathbb{E}^0_{t,x}[Y \mid \mathcal{F}_s]$. The law of total variance proves equation 10. Moreover, for $t < s_1 < s_2 \leq T$, the tower property yields $Y_{s_1} = \mathbb{E}^0_{t,x}[Y_{s_2} \mid \mathcal{F}_{s_1}]$, and therefore

$$\mathrm{Var}^0_{t,x}(Y_{s_1}) \leq \mathrm{Var}^0_{t,x}(Y_{s_2}) \leq \mathrm{Var}^0_{t,x}(Y).$$

The result holds for exact continuation values and unlogged reference-law weights. It does not compare errors of learned targets with different cutoffs.

D.2 Off-Policy Weights and the Optimal Proposal

With the exact continuation, define the behavior-sampled weight

$$M_{t,s} = e^{\ell_{t,s}} e^{-[\mathcal{A}_{t,s} + V(s, X_s)]/\lambda}. \tag{37}$$

Combining the likelihood ratio with the Doob density gives

$$\frac{M_{t,s}}{Z(t,x)} = \frac{d\mathbb{P}^\star_{[t,s]}}{d\mathbb{P}^\mu_{[t,s]}}. \tag{38}$$

Consequently, when the second moment is finite,

$$\frac{\mathrm{Var}^{\mu}_{t,x}(M_{t,s})}{Z(t,x)^2} = \chi^2(\mathbb{P}^{\star}_{[t,s]} \| \mathbb{P}^{\mu}_{[t,s]}). \tag{39}$$

Under the finite-cost assumptions in Appendix A, the optimal and reference laws are equivalent, so the optimal conditional process is an admissible zero-variance proposal for this exact weight. If infinite costs are allowed and the optimizer loses reference support, this specialization requires a separate support argument. A learned guided process need not lower the divergence, so off-policy correction establishes validity rather than an automatic variance improvement. For restrictions of fixed full-path laws, shortening the segment cannot increase this divergence, by conditional Jensen's inequality for their likelihood-ratio martingale.

D.3 A Diffusion Bound in Terms of Control Error

Consider a matched diffusion with behavior control u and optimal control $u^* = -\sigma^\top \nabla V/\lambda$. Assume the regularity, change-of-measure conditions, and square integrability needed below, and $\|u_r - u^*_r\|^2 \le \kappa$ uniformly. The HJB equation obtained from the log transform is

$$\partial_t V + b^0 \cdot \nabla V + \tfrac{1}{2}\,\mathrm{tr}(a\nabla^2 V) + c - \frac{1}{2\lambda}\|\sigma^\top \nabla V\|^2 = 0. \tag{40}$$

Itô's formula for equation 37, combined with equation 32, then gives

$$d\log M_{t,r} = -\tfrac{1}{2}\|u_r - u^*_r\|^2 dr + (u^*_r - u_r)^\top dW^u_r, \tag{41}$$

$$dM_{t,r} = M_{t,r}(u^*_r - u_r)^\top dW^u_r, \qquad M_{t,t} = Z(t,x). \tag{42}$$

Applying Itô's formula once more,

$$\frac{d}{dr}\mathbb{E}^u_{t,x}[M^2_{t,r}] = \mathbb{E}^u_{t,x}[M^2_{t,r}\|u_r - u^*_r\|^2] \le \kappa \mathbb{E}^u_{t,x}[M^2_{t,r}].$$

Gronwall's inequality and $\mathbb{E}^u[M_{t,s}] = Z(t,x)$ imply

$$\mathrm{Var}^u_{t,x}(M_{t,s}) \le Z(t,x)^2\left[e^{\kappa(s-t)} - 1\right]. \tag{43}$$

The uniform control bound ensures the relevant stochastic exponentials are true martingales on a finite interval. The calculation illustrates how segment length and proposal mismatch enter an exact-continuation weight bound; it is not a bound for arbitrary approximate values.

D.4 The Logarithm and Finite Branching

Fix (t,x), s, the behavior law, and the continuation W before drawing the branches. Let $w_j = e^{\ell^{(j)} - [\mathcal{A}^{(j)} + W(X^{(j)}_s)]/\lambda}$ be conditionally independent and identically distributed branch weights and $m = \mathbb{E}^\mu[w_j] \in (0,\infty)$. Then $\overline{w}_N = N^{-1}\sum_j w_j$ is unbiased for m, whereas convexity of $-\log$ gives

$$\mathbb{E}[-\lambda \log \overline{w}_N] \ge -\lambda \log m \tag{44}$$

whenever the left side is defined; it may be infinite if all sampled weights can vanish. With a positive sample mean and an integrable weight, the strong law gives almost-sure consistency as $N \to \infty$. If, for example, $0 < a \le w_j \le b < \infty$, Taylor expansion gives

$$\mathbb{E}[-\lambda \log \overline{w}_N] + \lambda \log m = \frac{\lambda}{2N}\frac{\mathrm{Var}(w_j)}{m^2} + o(N^{-1}), \tag{45}$$

$$\mathrm{Var}(-\lambda \log \overline{w}_N) = \frac{\lambda^2}{N}\frac{\mathrm{Var}(w_j)}{m^2} + o(N^{-1}). \tag{46}$$

These expansions require more than existence of the mean; bounded positive weights are one sufficient condition. Learned continuations add a separate approximation error, so the overall bias relative to V need not have the same sign as the finite-sample log bias.

E Full Training Procedure

Algorithm 1 summarizes the branch-replay procedure of Section 4.3. Write $\widetilde{V}$ for the detached online value or its EMA target copy, as specified for each experiment. Simulation and likelihood evaluation use the selected process adapter. An anchor-only variant draws fresh branches at each update, as in the CIFAR jump/hybrid runs; it uses the same target and regression loss.

Algorithm 1 Path Integral Value Matching (PI-VM)

Require: Segment simulator and likelihood evaluator, costs c, g, KL strength λ, training batch size B, branch count N, maximum branch length M, replay buffer $\mathcal{D}$

Initialize V_θ, set $\widetilde{V} \leftarrow V_\theta$, and initialize replay buffer $\mathcal{D}$
while training **do**
  if $\mathcal{D}$ is empty or a refresh is due **then**
    Roll out anchor trajectories using the reference or a chosen value-guided behavior process
    for each selected anchor (t, x) on the rollouts **do**
      Set a common cutoff $s \in (t, T]$
      Simulate N conditionally independent branches $\{\tau_{[t,s]}^{(j)}\}_{j=1}^{N}$ under the behavior law
      For each branch, compute/cache endpoint $X_s^{(j)}$, running cost $\mathcal{A}_{t,s}^{(j)}$, and log likelihood correction $\ell_{t,s}^{(j)}$
      Store $\big(t, x, s, \{\tau_{[t,s]}^{(j)}, X_s^{(j)}, \mathcal{A}_{t,s}^{(j)}, \ell_{t,s}^{(j)}\}_{j=1}^{N}\big)$ in $\mathcal{D}$
    end for
  end if
  Sample B replay entries $\{\xi_b\}_{b=1}^{B} \sim \mathcal{D}$
  for each replay entry ξ_b **do**
    Evaluate $g(X_T^{(b,j)})$ if $s_b = T$, otherwise $\widetilde{V}(s_b, X_{s_b}^{(b,j)})$, for all stored branch endpoints
    Form the PI-TD target $\widehat{V}(t_b, x_b)$ using the stored $\mathcal{A}^{(b,j)}$ and $\ell^{(b,j)}$ via equation 19
  end for
  Detach all targets and update θ using equation 20
  Update $\widetilde{V}$ from V_θ directly or by the specified EMA rule
end while
return V_θ and its well-posed value-guided dynamics

F Euclidean Benchmark Details

F.1 Problems and reference solutions

All OU tasks use $T = \lambda = 1$, $X_0 \sim \mathcal{N}(0, 0.25I)$, and

$$\mathrm{d}X_t = (AX_t + \Sigma u_t)\,\mathrm{d}t + \Sigma\,\mathrm{d}W_t, \qquad J(u) = \mathbb{E}\left[\int_0^1 \left(f(X_t) + \frac{1}{2}\|u_t\|^2\right)\mathrm{d}t + g(X_1)\right].$$

Table 7 gives the problem parameters. Linear OU uses the same $\Xi_{ij} \sim \mathcal{N}(0, 0.1^2)$ in A and Σ. For $\psi_t = e^{A^\top(1-t)}\mathbf{1}$,

$$u_t^* = -\Sigma^\top \psi_t, \qquad V^*(t, x) = \psi_t^\top x - \frac{1}{2}\int_t^1 \psi_s^\top \Sigma\Sigma^\top \psi_s\,\mathrm{d}s.$$

For Quadratic OU, $f(x) = x^\top Px$ and $g(x) = x^\top Qx$. The easy-task sweep uses $d \in \{10, 20, 40, 80, 100, 150, 200\}$ without dimension-normalizing the costs. The reference solution is

$$\dot{F}_t + A^\top F_t + F_t A - 2F_t\Sigma\Sigma^\top F_t + P = 0, \qquad F_1 = Q,$$
$$u_t^*(x) = -2\Sigma^\top F_t x,$$
$$V^*(t, x) = x^\top F_t x + \int_t^1 \mathrm{Tr}\big(\Sigma\Sigma^\top F_s\big)\,\mathrm{d}s.$$

Table 7: OU problem parameters and training budgets. Epoch counts apply to PI-VM, AM, and VM.

	Linear	Quadratic easy	Quadratic hard
Dimension	10	20	20
A	$-I + \Xi$	$0.2I$	I
Σ	$I + \Xi$	I	I
Running cost	0	$0.2\|x\|^2$	$\|x\|^2$
Terminal cost	$\mathbf{1}^\top x$	$0.1\|x\|^2$	$0.5\|x\|^2$
Time step	0.01	0.02	0.02
Training epochs	60,000	60,000	80,000
VM trajectories/epoch	64	128	128
VM regression states/epoch	6,400	6,400	6,400

F.2 Training configuration

PI-VM, AM, and VM use a learning rate of 10^{-4} and the epoch counts in Table 7, with one update per epoch. The value networks are GELU MLPs with hidden-layer widths $(64, 128, 256, 128, 64)$ for $d \le 64$, $(128, 256, 256, 128, 64)$ for $64 < d \le 128$, and $(256, 256, 256, 128, 64)$ for $d > 128$.

PI-VM uses a batch of 6,400 anchors with $N = M = 8$: eight branches per anchor and a maximum branch length of eight steps. Its replay buffer has capacity 100,000 and is refreshed with 128 trajectories every 50 epochs. AM uses batches of 64 full trajectories. VM uses the trajectory and regression-state batch sizes in Table 7.

F.3 Evaluation

For a d-dimensional OU task, the control error is

$$\mathcal{L}_2(u, u^*) = \frac{1}{d}\mathbb{E}_{\mathbb{P}^*}\left[\int_0^1 \|u(t, X_t) - u^*(t, X_t)\|_2^2 \, \mathrm{d}t\right].$$

G Jump Benchmark Details

G.1 Problems and costs

Continuous-state jumps evolve on $[-1, 1]^4$ with $X_0 = 0$, $T = \lambda = 1$, and reference rate one per coordinate. At an event, Partial sets $x_i \leftarrow 0.5x_i + 0.5\xi$ and Refresh sets $x_i \leftarrow \xi$, where $\xi \sim \mathrm{Unif}[-1, 1]$. The objective uses path-KL and terminal cost $g = -\log h_T$, with no running or additional jump cost. For Refresh,

$$h_T(x) = 0.05 + 0.95\,\frac{0.6e^{1.5\sum_i x_i} + 0.4e^{-1.5\sum_i x_i}}{C^4}, \qquad C = \frac{\sinh(1.5)}{1.5}.$$

For Partial, with indices $i = 0, \ldots, 3$,

$$h_T(x) = 0.05 + 0.95 \sum_{s\in\{-1,1\}^4} \pi(s) \prod_{i=0}^{3} (1 + 0.7x_i s_i)^3,$$

$$\pi(s) \propto \exp\left(0.5\sum_{i=0}^{2} s_i s_{i+1} + 0.13\sum_{i=0}^{3}(-1)^i s_i\right).$$

G.2 Training configuration

PI-VM and DAM use GELU MLPs with hidden-layer widths $(64, 128, 256, 128, 64)$. Their input dimensions are five and ten, respectively, with scalar outputs. Table 8 lists the epoch counts, batch sizes, learning rates, and replay capacities; each epoch consists of one update. PI-VM uses $N = M = 8$, with eight branches per anchor and a maximum time window of $M \times 0.025 = 0.2$. Both methods prefill their replay buffers with 4,096 records and refresh 256 records every 16 epochs.

Table 8: Jump training configuration. Learning-rate arrows indicate initial and final values.

Task	Method	Epochs	Batch size	Replay capacity	Learning rate
Partial	PI-VM	24,000	512	4,096	$1.5 \times 10^{-3} \to 10^{-5}$
Refresh	PI-VM	24,000	512	4,096	$5.625 \times 10^{-4} \to 10^{-5}$
Partial	DAM	12,000	256	4,096	3×10^{-4}
Refresh	DAM	12,000	256	4,096	3×10^{-4}

G.3 Evaluation

We compare the induced drift $b_\nu(t,x) = \int (y-x)\nu_t(x,\mathrm{d}y)$ with its optimum using

$$\mathcal{L}_2(b_\nu, b^*) = \frac{1}{4}\mathbb{E}_{\mathbb{P}^*}\left[\int_0^1 \|b_\nu(t,X_t) - b^*(t,X_t)\|_2^2 \,\mathrm{d}t\right].$$

H CTMC Benchmark Details

H.1 Problems and costs

The CTMCs start at zero on $\mathcal{X} = \{-1, -0.875, \ldots, 0.875, 1\}^4$, containing 83,521 states, with $T = \lambda = 1$ and reference rate one per coordinate. For the one-dimensional grid $\{z_j\}$, let ϕ_j denote its piecewise-linear hat functions. Each coordinate has generator $Q = P - I$. Partial uses $P_{ij} = \mathbb{E}_{\xi\sim\mathrm{Unif}[-1,1]}[\phi_j(0.5z_i + 0.5\xi)]$. Refresh uses $P_{ij} = \mu_j$, where $\mu_j = 1/32$ at the two endpoints and $\mu_j = 1/16$ at interior grid points.

The objective uses path-KL and terminal cost $g = -\log h_T$, with no running or additional jump cost. For Refresh,

$$h_T(x) = 0.05 + 0.95\,\frac{0.6e^{1.5\sum_i x_i} + 0.4e^{-1.5\sum_i x_i}}{C^4}, \qquad C = \sum_j \mu_j e^{1.5 z_j}.$$

For Partial,

$$h_T(x) = 0.05 + 0.95 \sum_{s\in\{-1,1\}^4} \pi(s) \prod_{i=0}^{3} (1 + 0.7 x_i s_i)^3,$$

$$\pi(s) \propto \exp\left(0.5\sum_{i=0}^{2} s_i s_{i+1} + 0.13 \sum_{i=0}^{3} (-1)^i s_i\right).$$

H.2 Training configuration

PI-VM and DAM use GELU MLPs with hidden-layer widths $(64, 128, 256, 128, 64)$. Their input dimensions are five and ten, respectively, with scalar outputs. Table 9 gives their training configurations, with one update per epoch. PI-VM uses $N = M = 8$, with eight branches per anchor and a maximum time window of $M \times 0.025 = 0.2$.

PI-VM generates 16,400 anchors for the initial Partial buffer and 4,120 for Refresh, retaining the capacities in Table 9. Every 16 epochs, it inserts 1,040 new anchors for Partial and 280 for Refresh. DAM prefills 4,096 endpoint records and refreshes 256 records every 16 epochs.

H.3 Evaluation

For transition rates $\nu_t(x,y)$, define the induced drift by $b_\nu(t,x) = \sum_{y\in\mathcal{X}}(y-x)\nu_t(x,y)$. We evaluate

$$\mathcal{L}_2(b_\nu, b^*) = \frac{1}{4}\int_0^1 \sum_{x\in\mathcal{X}} p_t^*(x)\|b_\nu(t,x) - b^*(t,x)\|_2^2 \,\mathrm{d}t,$$

where p_t^* is the optimal occupancy distribution.

Table 9: CTMC training configuration. The learning-rate arrow indicates initial and final values.

Task	Method	Epochs	Batch size	Replay capacity	Learning rate
Partial	PI-VM	24,000	1,024	16,384	$10^{-4} \to 10^{-5}$
Refresh	PI-VM	24,000	2,048	4,096	10^{-4}
Partial	DAM	12,000	256	4,096	3×10^{-4}
Refresh	DAM	12,000	256	4,096	3×10^{-4}

Table 10: Manifold training epochs, batch sizes, and Hard-SOC networks.

Task	PI-VM epochs	Hard-SOC epochs	PI-VM batch	Hard-SOC batch	Hard-SOC network
Sphere	20,000	600	3,000	500	4–64–64–3
Stiefel, $\beta = 0.01$	10,000	1,000	3,000	600	9–256–256–8
Stiefel, $\beta = 0.5$	20,000	1,000	3,000	600	9–256–256–8

I Manifold Benchmark Details

I.1 Problems

All three tasks use $T = 1$, diffusion coefficient one, no running cost, and $J = \mathbb{E}[\frac{1}{2}\int_0^1 \|u_t\|^2 \mathrm{d}t + g(X_1)]$. On $\mathbb{S}^2 \subset \mathbb{R}^3$ (intrinsic dimension two), a normalized standard Gaussian gives the uniform initial distribution and $g(x) = 6(1 - x_3^2)$. Dynamics use tangent projection and normalization retraction. On $\mathrm{St}(4, 2)$ (ambient dimension eight, intrinsic dimension five), Gaussian QR initializes the Haar distribution. The cost is $g(X) = \beta \operatorname{Tr}(X^\top H X)$, $\beta \in \{0.01, 0.5\}$, with

$$H = \begin{pmatrix} 4 & 0.5 & 2.5 & 1 \\ 0.5 & 4 & 1 & 2.5 \\ 2.5 & 1 & 4 & 0.5 \\ 1 & 2.5 & 0.5 & 4 \end{pmatrix}.$$

We use tangent projection $Z - X \operatorname{sym}(X^\top Z)$ and QR retraction.

I.2 Training configuration

PI-VM uses GELU MLPs with hidden-layer widths $(64, 128, 256, 128, 64)$, and a learning rate of 10^{-4}. We set $N = M = 8$: eight branches per anchor and a maximum branch length of eight steps. The replay capacity is 100,000. Initialization generates 30,720 records; an initial refresh brings the buffer to 32,768 records. Subsequent refreshes use 32 trajectories to add up to 2,048 anchors every 50 epochs.

Hard-SOC uses Tanh activations and learning rates 2×10^{-3} on the sphere and 10^{-3} on Stiefel. See Table 10 for training epochs, batch sizes, and Hard-SOC network widths, including input and output layers.

I.3 Evaluation

The control error is

$$\mathcal{L}_2(u, u^*) = \frac{1}{d_{\mathrm{int}}} \mathbb{E}_{\mathbb{P}^*}\left[\int_0^1 \|u(t, X_t) - u^*(t, X_t)\|^2 \,\mathrm{d}t\right],$$

where $d_{\mathrm{int}} = 2$ on the sphere and $d_{\mathrm{int}} = 5$ on $\mathrm{St}(4, 2)$. Stiefel uses the Frobenius norm.

J Ablation Studies

We study branch count and trajectory length, followed by network depth and anchor/branch allocation. Let B denote replay anchors and N branches per anchor; the final three studies use Linear OU.

Table 11: Branch count N and length M: control error (top) and update time in ms (bottom).

	Branch steps M		
Branches N	4	8	16
4	0.00013 7 ms	0.00011 7 ms	0.00010 8 ms
8	0.00012 7 ms	**0.00010** **7 ms**	0.00010 9 ms
16	0.00011 9 ms	0.00010 13 ms	0.00009 18 ms

J.1 Branch count and trajectory length

We vary $N, M \in \{4, 8, 16\}$, where M is the maximum branch length. Table 11 reports control error and per-update time, with the default $(N, M) = (8, 8)$ in bold.

The default $(N, M) = (8, 8)$ attains an error of 0.00010 at 7 ms per update. At $N = 8$, extending M from 4 to 8 reduces error from 0.00012 to 0.00010 at the same displayed update time; extending it to 16 leaves the error unchanged at the reported precision. Increasing both N and M to 16 reduces error only to 0.00009 while raising update time to 18 ms, making $(8, 8)$ a useful accuracy–cost compromise in this comparison.

J.2 Network depth

We compare GELU hidden widths $(64, 128, 64)$ (shallow), $(64, 128, 256, 128, 64)$ (default), and $(64, 128, 256, 512, 256, 128, 64)$ (deep). The deep network has the lowest early error, while the default is slightly better at 9,000 updates.

Table 12: Control error by network depth and training update.

Iterations	Shallow	Default	Deep
3000	0.000818	0.000542	0.000434
5000	0.000289	0.000210	0.000192
7000	0.000157	0.000125	0.000123
9000	0.000094	**0.000082**	0.000089

J.3 Anchor and branch allocation

All three configurations in Table 13 use $BN = 51,200$ stored branch contributions per regression update, varying their allocation across anchors. This fixes the number of endpoint contributions, not total simulation work or wall-clock cost; replay updates reuse branches collected during buffer initialization and refresh. More distinct anchors give lower error at every reported checkpoint.

Table 13: Control error at fixed $BN = 51,200$.

Iterations	B=6400, N=8	B=3200, N=16	B=1600, N=32
3000	0.000542	0.000571	0.000722
5000	0.000210	0.000244	0.000299
9000	**0.000082**	0.000093	0.000114

J.4 Branch count at fixed anchor batch

Here $B = 6,400$ is fixed. Increasing N raises endpoint-evaluation work per update and the cost of generating new branch groups during collection. Larger branch counts improve early accuracy, with nearly equal errors at 9,000 updates.

Table 14: Control error with a fixed anchor batch and varying branch count.

Iterations	B=6400, N=8	B=6400, N=16	B=6400, N=32
3000	0.000542	0.000483	0.000477
9000	0.000082	0.000082	**0.000081**

Table 15: Diffusion image-training budgets per task and method.

Model	Task	VM NFE (M)	PI-VM NFE (M)	VM hours	PI-VM hours
CIFAR	Compression	12.800	13.624	4.42	4.17
CIFAR	Incompression	12.800	13.726	3.05	5.96
CIFAR	Aesthetic	12.800	13.726	3.60	6.76
DiT	Compression	32.000	25.969	24.00	22.08
DiT	Incompression	32.000	25.969	26.17	22.17
DiT	Aesthetic	32.000	25.969	43.97	39.68

K Image-Generation Settings and Selected Samples

K.1 Models, rewards, and value networks

Frozen generators are `google/ddpm-cifar10-32` at 32×32 and DiT-XL/2 with its VAE at 256×256 (latents $4 \times 32 \times 32$). Diffusion uses 100 memoryless-SDE steps over $[0.02, 1]$. JPEG quality is 85; compression/incompression rewards are negative/positive bits per pixel. Aesthetics uses fixed OpenAI CLIP ViT-L/14 and a LAION linear head. Terminal cost is $g = -\eta r$, with reward multiplier η distinct from path-KL coefficient λ: $\eta = 10$ for CIFAR and DiT aesthetics, and 25 for DiT JPEG tasks.

VM and diffusion PI-VM share a scalar CNN with widths $(64, 128, 256)$, time embeddings, BlurPool, and zero output weights with an estimated terminal-cost bias. The DiT value model uses eight-group normalization and a $(128, 256, 256)$ time MLP, without class embeddings. DiT generator inference uses fp16 and microbatch 32; other computations use fp32, with TF32 disabled. CIFAR uses fp32.

K.2 Diffusion training and compute

Training uses 1,000 updates for CIFAR and 2,500 for DiT. Adam holds 10^{-4} for 100 updates, then cosine-decays to 10^{-6}; gradient clipping is one. VM and PI-VM share time weights within each task: the paper convention for CIFAR and released VM weights for DiT.

VM collects 128 fresh complete trajectories per update, regresses all 101 states per path, and evaluates the online value. Diffusion PI-VM prefills 100,000 anchors at the same replay capacity, samples 12,800 without replacement per update, and uses regression microbatch 128. Each anchor has eight branches of up to eight steps. EMA decay is 0.9; the target network supplies collection, branching, continuation, and evaluation. After update 100, every 50 updates add 6,400 entries from 128 paths and 50 paired-half time points per path. DiT training uses CFG zero (conditional prediction without extrapolation); evaluation uses CFG two, $\epsilon_{\text{uncond}} + 2(\epsilon_{\text{cond}} - \epsilon_{\text{uncond}})$.

Table 15 counts per-sample generator evaluations (NFE) for training, replay prefill, and refresh.

K.3 CIFAR jump and hybrid adaptation

The jump reference is a Poissonized frozen-DDPM generator with rate $\nu = 400/0.98$ and whole-image Gaussian landing proposal

$$Y = x + b(t, x)/\nu + \sigma(t)\varepsilon/\sqrt{\nu}, \qquad \varepsilon \sim \mathcal{N}(0, I).$$

Coefficients are frozen on the 100-cell grid. Thinning uses rate multiplier $\exp(\text{clip}(V(t, x) - V(t, y), -\log 2, \log 2))$, a bounded Doob approximation. The hybrid reference $\frac{1}{2}\mathcal{L}_D + \frac{1}{2}\mathcal{L}_J$ scales diffusion drift/noise by $1/2$ and $1/\sqrt{2}$, combining Euler diffusion and frozen-time jumps by Lie splitting.

Both variants use 1,000 updates, 6,400 fixed anchors from 64 reference paths, batch/microbatch 256/64, eight fresh reference branches per anchor, and horizon 0.0784. Adam uses 10^{-4}, clipping one, and EMA 0.9. Diffusion time weights are reused heuristically, without diffusion replay. Jump compression resets the optimizer after update 50. These budgets differ from diffusion PI-VM.

K.4 Evaluation and figure conventions

CIFAR uses 1,000 images, seed 10001, batch 16, and CLIP ViT-B/32 Vendi. Figure 3 reports reward gain and $100\ \mathrm{Vendi}(\text{method})/\ \mathrm{Vendi}(\text{base})$ against each process's own base. Horizontal bars are paired-bootstrap 95% reward intervals from 10,000 resamples, conditional on the checkpoint. Processes share initial Gaussians by sample index, not subsequent paths.

DiT uses 1,024 samples per method/task, CFG two, and batch 32, with paired labels, initial latents, and SDE noises. Reward and Vendi use the same samples. Vendi uses OpenAI CLIP ViT-L/14 fp32 embeddings from uint8 RGB, bicubic resize/center crop to 224, and CLIP normalization. Order-one Vendi is computed over the full normalized embedding set with a cosine kernel. It measures feature diversity, not image quality or target-distribution fit; comparisons are within each model/task protocol.

Qualitative panels show selected samples; metrics use full unfiltered sets. DiT columns share labels and noises across methods, and CIFAR has a base row per process. All diffusion PI-VM points/panels use 100,000-anchor replay, with sample indices shared across methods within each task.

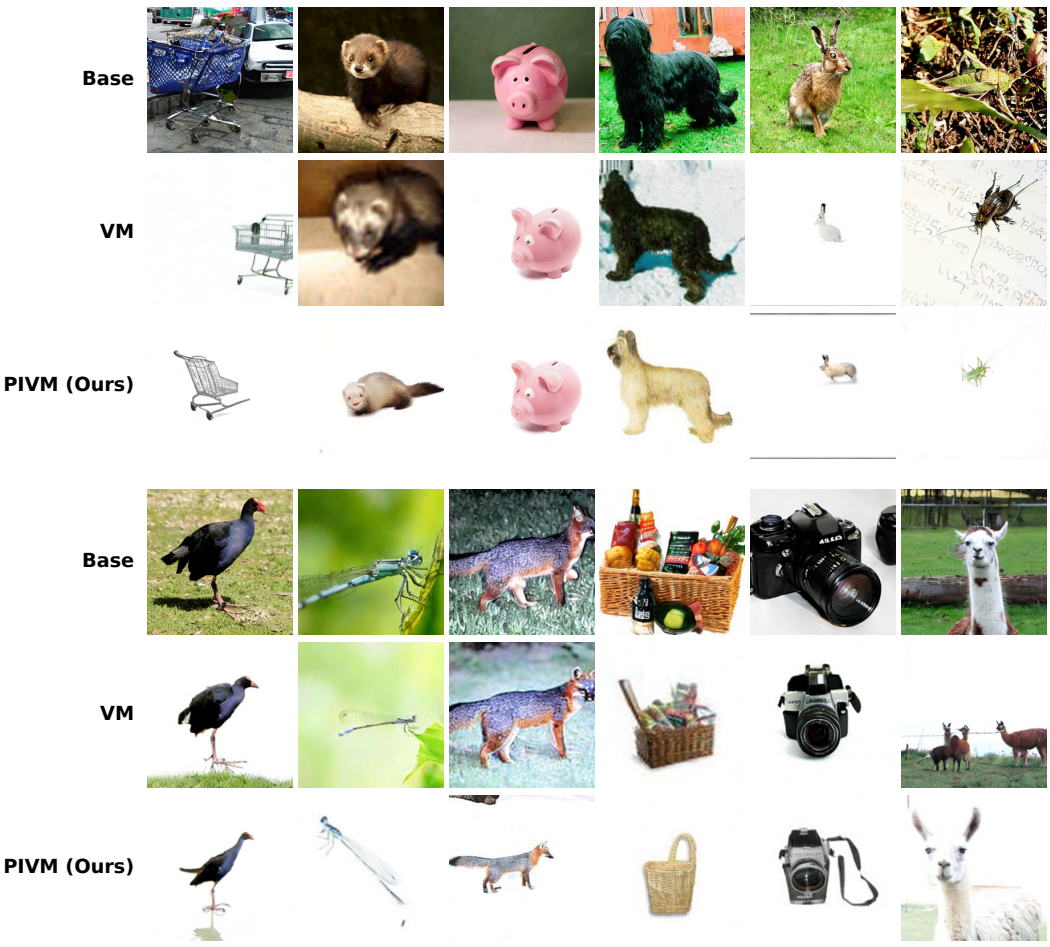

Figure 5: Selected DiT compression samples: Base, VM, and PI-VM.

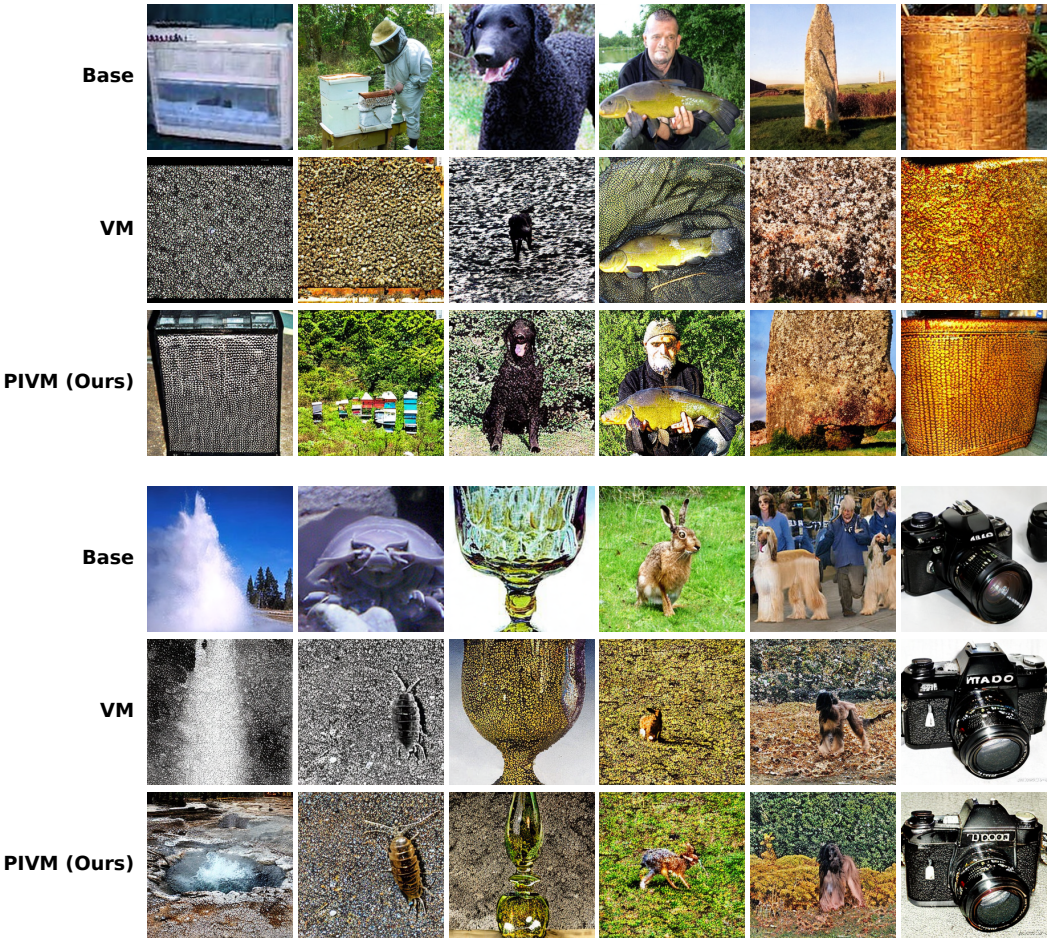

Figure 6: Selected DiT incompression samples: Base, VM, and PI-VM.

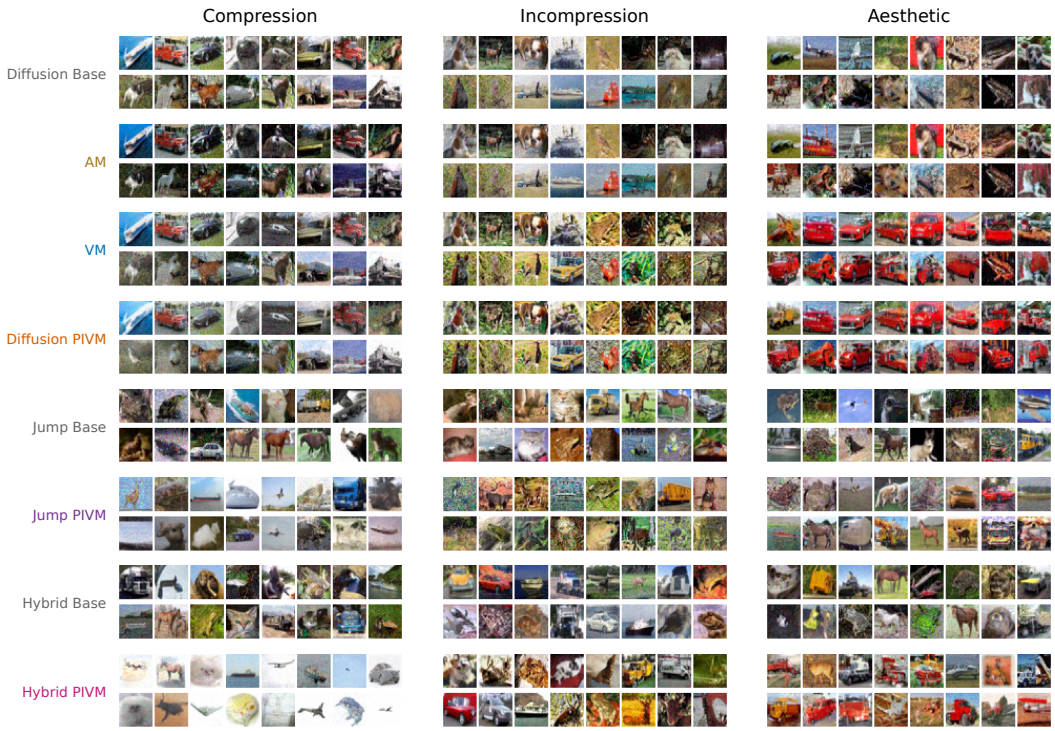

Figure 7: Selected CIFAR-10 samples for compression, incompression, and aesthetics.