Reward Score Matching: Unifying Reward-based Fine-tuning for Flow and Diffusion Models

Jeongjae Lee* **Jinho Chang*** **Jeongsol Kim†** **Jong Chul Ye†**
Graduate School of AI
KAIST, Korea
{jaylee2000, jinhojsk515, jeongsol, jong.ye}@kaist.ac.kr
*First authors †Corresponding authors

Abstract

Reward-based fine-tuning aims to steer a pretrained diffusion or flow-based generative model toward higher-reward samples while remaining close to the pretrained model. Although existing methods are motivated by different perspectives such as Soft RL, GFlowNets, etc., we show that many can be written under a common framework, which we call *reward score matching* (RSM). Under this view, alignment becomes score matching toward a reward-guided target, and the main differences across methods reduce to the construction of the *value-guidance estimator* and the *effective optimization strength* across timesteps. This unification clarifies the bias–variance–compute tradeoffs of existing designs and distinguishes core optimization components from auxiliary mechanisms that add complexity without clear benefit. Guided by this perspective, we develop simpler redesigns that improve alignment effectiveness and compute efficiency across representative settings with differentiable and black-box rewards. Overall, RSM turns a seemingly fragmented collection of reward-based fine-tuning methods into a smaller, more interpretable, and more actionable design space.

1 Introduction

Reward-based fine-tuning seeks to steer a pretrained diffusion or flow-based generative model toward higher-reward samples without deviating too far from the pretrained model. This problem has recently attracted a wide range of methods, including entropy-regularized reinforcement learning (Soft RL) [3, 9, 12, 20, 21, 24, 50], optimal control [26], and GFlowNets [25, 27, 51]. As a result, the literature appears fragmented: methods are presented with different objectives, derivations, and stabilization heuristics, making it difficult to tell which differences are fundamental and which are merely implementational.

In this paper, we show that many such methods reduce to a single underlying objective, which we term *Reward Score Matching* (RSM; see Table 1). At a high level, reward alignment can be viewed as score matching toward an ideal reward-guided target distribution which is composed of a reference term and an effective value-guidance term, together with timestep-dependent weighting or trust-region control. This unified view also clarifies the key distinction across methods.

In practice, this target is intractable to compute in high dimensions, so existing methods replace it with tractable value-guidance surrogates. Accordingly, the primary difference lies in *how the value-guidance surrogate is constructed*: some methods directly compute reward gradients, while others rely solely on reward evaluations, estimating the same signal through sampling. Applying Stein's identity [38], one of the most important contributions of this work is to show that these two strategies correspond precisely to first-order and zeroth-order estimations of a common underlying target, respectively.

Preprint.

Table 1: **Various methods admit specific choices within common RSM loss.** The common RSM loss $\mathcal{L}(\theta)$ is given by $\mathbb{E}_{t_i,\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}}\left[C_1(t_i)\Big(\|\boldsymbol{s}^{\theta}_{t_i} - (\boldsymbol{s}^{\text{ref}}_{t_i} + \boldsymbol{\Psi}_{t_i})\|^2 + C_2(t_i)\|\boldsymbol{s}^{\theta}_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i}\|^2\Big)\right]$. Each method specifies the value-guidance estimator for $\boldsymbol{\Psi}_{t_i}$, lookahead depth j, branching ($K_i > 1$), and temporal weights $\gamma(t_i)$, $C_1(t_i)$, and $C_2(t_i)$. The resulting *Normalized Influence Metric* $h(t_i)$ summarizes optimization strength.

Method	**j**	**Branching**	$\gamma(t_i)$	$C_1(t_i)$	$C_2(t_i)$	$h(t_i)^\dagger$
First-order estimators						
VGG-Flow‡	i‡	No	$(1-t_i)^2\delta(t_i)$	$\frac{1}{d}\frac{1}{\delta(t_i)^2}$	0	$\frac{1}{\alpha d}(1-t_i)^2$
SQDF	$i-1$	No	$\tilde{\gamma}^{Nt_i}_{\text{SQDF}}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	0	$\tilde{\gamma}^{Nt_i}_{\text{SQDF}}\frac{\sigma_{t_i}w(t_i)}{2}$
Residual ∇-DB‡	$i-1$	No	$\bar{\alpha}_{t_{i-1}}$	$\frac{1}{d}\frac{\Omega(t_i)^2}{\sigma_{t_i}^4}$	$\frac{(1-\bar{\alpha}_{t_i})\sigma_{t_i}^4}{\Omega(t_i)^2}\frac{w_R}{w_F}$	$\frac{1}{\alpha d}\bar{\alpha}_{t_{i-1}}\frac{w(t_i)}{\sigma_{t_i}}$
Zeroth-order estimators						
REINFORCE + KL reg	0	No	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	0	$\frac{\sigma_{t_i}w(t_i)}{2}$
PPO / GRPO / PCPO-base	0	No	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma_{t_i}w(t_i)}{2}$
PCPO-reweight (diffusion)	0	No	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma'_{t_i}w'(t_i)}{2}$
PCPO-reweight (flow)	0	No	$\frac{w'(t_i)}{w(t_i)}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\gamma(t_i)\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma_{t_i}w'(t_i)}{2}$
Branch-GRPO	0	Yes	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma_{t_i}w(t_i)}{2}$
TempFlow-GRPO	0	Yes	$\frac{9}{4}\sigma_{t_i}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\gamma(t_i)\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{9\sigma_{t_i}^2w(t_i)}{8}$
GRPO-Guard	0	No	$\frac{\sigma_{t_i}\Omega(t_i)}{\Delta t_i}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	0	$\frac{\sigma_{t_i}^3w(t_i)^2}{2(\Delta t_i)\delta(t_i)}$

† $w(t_i)$ is equal to $\Omega(t_i)\delta(t_i)/\sigma_{t_i}$ (see Appendix D.3).
‡ The reduction becomes explicit after pruning redundant auxiliary components (see Appendix G.1).
‡ $j = i$ denotes usage of a *current-state* estimator.

This reduction is especially revealing for policy-gradient methods: using PCPO's clipped log-ratio surrogate [20], we show that the PPO-clip objective also admits the same score-matching L_2 form. Furthermore, methods derived from seemingly different viewpoints such as GFlowNet and direct regression objectives often do not optimize fundamentally different targets, but rather different tractable surrogates of the same reward-guided score-matching problem.

Once this distinction is isolated, the remaining design space can be studied on common ground: *estimator design* determines guidance quality, *temporal weighting* scales its magnitude across timesteps, and *trust-region realization* determines how much of it survives regularization. This decomposition also helps separate core optimization ingredients from auxiliary mechanisms that add theoretical structure or implementation complexity without clearly improving the underlying update.

A second contribution of this paper is to use this unified framework as a practical diagnostic tool. RSM reveals shared bias–variance–compute tradeoffs across methods that would otherwise appear unrelated. For example, first-order Tweedie-based guidance can be simple but biased, especially at low-SNR timesteps, whereas zeroth-order full-rollout guidance can be unbiased but noisy and compute-intensive. Likewise, several recently proposed components can be reinterpreted as secondary coefficient-level modifications rather than distinct algorithmic principles. This yields a more systematic lens on the design space and explains why some inherited choices are more consequential than others.

Guided by this perspective, we show that several seemingly important design choices are secondary, while a few principled changes drive most of the practical improvements. Specifically, by comparing their value-guidance estimators through shared bias–variance–compute tradeoffs, we show that some auxiliary components can be removed without hurting performance. These insights lead to simple redesigns that improve alignment quality while reducing compute cost across representative differentiable-reward and black-box-reward settings, including a $5\times$ wall-clock speedup over TempFlow-GRPO [12] on the GenEval reward [11].

2 Preliminaries

2.1 Flow-based Models

Let $p_0(\boldsymbol{x}_0)$ and $p_1(\boldsymbol{x}_1)$ denote the target and source distributions, respectively. Flow-based models transport samples from $\boldsymbol{x}_1 \sim p_1$ to $\boldsymbol{x}_0 \sim p_0$ along a probability path $p_t(\boldsymbol{x}_t)$, induced by a flow map $\psi_t : \mathbb{R}^d \to \mathbb{R}^d$ over $t \in [0, 1]$, with boundary conditions $\psi_1(\boldsymbol{x}_1) = \boldsymbol{x}_1$ and $\psi_0(\boldsymbol{x}_1) = \boldsymbol{x}_0$. The corresponding trajectory $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_1)$ is governed by the flow ODE

$$d\boldsymbol{x}_t = \boldsymbol{v}_t(\boldsymbol{x}_t)dt, \tag{1}$$

where $\boldsymbol{v}_t(\boldsymbol{x}) = \dot{\psi}_t(\psi_t^{-1}(\boldsymbol{x}))$ is the velocity field. Prior work [1, 7, 24, 37] identified a reverse-time flow SDE that preserves the marginal distribution as:

$$d\boldsymbol{x}_t = \Big(\boldsymbol{v}_t(\boldsymbol{x}_t) - \frac{\tilde{\sigma}_t^2}{2}\nabla \log p_t(\boldsymbol{x}_t)\Big)dt + \tilde{\sigma}_t d\boldsymbol{w}. \tag{2}$$

We denote the diffusion coefficient in the SDE by $\tilde{\sigma}_t$, and its discretized counterpart by $\sigma_t = \tilde{\sigma}_t\sqrt{\Delta t}$. In conditional flow matching [23], one defines a conditional flow $\psi_t(\cdot \mid \boldsymbol{x}_0) : \mathbb{R}^d \to \mathbb{R}^d$ with conditional velocity field $\boldsymbol{v}_t(\boldsymbol{x} \mid \boldsymbol{x}_0)$, and marginal velocity $\boldsymbol{v}_t(\boldsymbol{x}) = \int \boldsymbol{v}_t(\boldsymbol{x} \mid \boldsymbol{x}_0)\, p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)\, d\boldsymbol{x}_0$.

2.2 Affine Conditional Flow

A widely used family of conditional flows is the affine conditional flow, defined as

$$\psi_t(\boldsymbol{x}_1|\boldsymbol{x}_0) = a_t\boldsymbol{x}_0 + b_t\boldsymbol{x}_1 \triangleq \boldsymbol{x}_t \tag{3}$$

where $a_t \in \mathbb{R}$ and $b_t \in \mathbb{R}$ denote time-dependent variables. From the definition of marginal velocity field and $p_1 = \mathcal{N}(0, \mathbf{I})$, the flow-ODE constructed with the affine flow is [1]

$$d\boldsymbol{x}_t = \frac{\dot{a}_t}{a_t}\boldsymbol{x}_t dt + \left(\frac{\dot{a}_t b_t^2 - a_t\dot{b}_t b_t}{a_t}\right)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)dt. \tag{4}$$

We establish notation generally using the affine conditional flow ODE, since it subsumes major generative paradigms such as diffusion models with VP-SDE, VE-SDE, etc.

VP-SDE. We define the coefficients $a_t = \sqrt{\bar{\alpha}_t}$ and $b_t = \sqrt{1 - \bar{\alpha}_t}$, where $\bar{\alpha}_t = \exp\left(-\int_0^t \beta(s)ds\right)$ is derived from a pre-defined noise scheduler $\beta(s)$ satisfying $\beta(0) = 0$ and strictly increasing $\beta(t)$. Consequently, the flow ODE becomes

$$d\boldsymbol{x}_t = -\frac{1}{2}\beta(t)\boldsymbol{x}_t dt - \frac{1}{2}\beta(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)dt. \tag{5}$$

VE-SDE. Let $a_t = 1$ and $b_t = \sigma_t$ where σ_t denotes a pre-defined noise variance schedule satisfying $\sigma_0 = 0$ and $\sigma_t < \sigma_{t+1}$. The corresponding flow ODE is

$$d\boldsymbol{x}_t = -\dot{\sigma}_t\sigma_t\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)dt. \tag{6}$$

Rectified Flow Let $a_t = 1 - t$ and $b_t = t$ for $t \in [0, 1]$. Then, the flow ODE becomes

$$d\boldsymbol{x}_t = \frac{\boldsymbol{x}_t - \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]}{t}dt. \tag{7}$$

For brevity, we denote the score function as $\boldsymbol{s}_t \equiv \boldsymbol{s}(\boldsymbol{x}_t, t) := \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ for the remainder of the paper. Then, the transition density of the discrete-time affine flow SDE takes the general form:

$$p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) = \mathcal{N}(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}_{t_i}, \sigma(t_i)^2\mathbf{I}), \tag{8}$$

where $\kappa(t_i)$ is fully determined by the definition of affine flow, and $\Omega(t_i), \sigma(t_i)$ are also affected by the stochastic noise schedule. Proof is in Appendix B.1 [2]. See also Appendix D.1 for the derivation of $\Omega(t_i)$ for common samplers.

[1] For brevity, we write $\nabla_{\boldsymbol{x}_t} f(\boldsymbol{x}_t)$ as shorthand notation of $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_t}$ for the remainder of this paper.

[2] The network may instead parameterize a time-dependent affine transform of the score. We encapsulate such affine transforms via $\delta(t_i)$ (see Appendix D.2).

2.3 Training-based Reward Alignment

Soft RL formulates alignment as an entropy-regularized Markov decision process over discretized reverse-time transitions [2, 3, 9, 41]. With timesteps $0 = t_0 < t_1 < \cdots < t_N = 1$, the state is $\boldsymbol{x}_t \in \mathbb{R}^d$, the learnable reverse transition $p^\theta(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})$ is the time-indexed policy, and reward is received only at the terminal state. This yields the objective

$$\max_\theta \mathbb{E}_{\boldsymbol{x}_0 \sim p_0^\theta}[r(\boldsymbol{x}_0)] - \alpha \sum_i \mathcal{D}_{\mathrm{KL}}(p^\theta(\cdot|\boldsymbol{x}_{t_i}) \| p^{\mathrm{ref}}(\cdot|\boldsymbol{x}_{t_i})). \tag{9}$$

A key object is the *soft value function* $V_t(\boldsymbol{x}_t)$, which combines future reward and entropy regularization. The corresponding optimal marginal satisfies

$$p_t^\star(\boldsymbol{x}_t) = \frac{p_t^{\mathrm{ref}}(\boldsymbol{x}_t)}{Z} \exp\left(\frac{V_t(\boldsymbol{x}_t)}{\alpha}\right), \tag{10}$$

where Z is a normalizing constant independent of t [41]. Soft RL subsumes Residual ∇-DB [27], SQDF [18], and policy-gradient methods.

Several alignment methods are not fully captured by Soft RL. VGG-Flow [26] and DiffusionNFT [52] directly regress the model velocity field toward a reward-guided target. Several recent methods also introduce timestep-dependent weighting [4, 12, 20, 45] to improve stability and convergence, mirroring the use of reweighting in diffusion pretraining [14]. In the next section, we show that many of these methods can be unified under a common RSM formulation[3], which also provides a systematic lens on temporal weighting.

3 Reward Score Matching: A Unified Framework

3.1 Optimal Guidance and Common Objective

Eq. (10) implies that the optimal intermediate marginal $p_t^\star$ differs from the reference marginal p_t^{ref} by an exponential tilt of the soft value function V_t. Accordingly, the optimal score takes the form

$$\boldsymbol{s}_t^\star(\boldsymbol{x}_t) = \boldsymbol{s}_t^{\mathrm{ref}}(\boldsymbol{x}_t) + \frac{1}{\alpha}\nabla_{\boldsymbol{x}_t} V_t, \tag{11}$$

which leads to the following first order solver with the optimal valude guidance:

Proposition 3.1 (Optimal value guidance)**.** *Consider a first-order solver with transitional kernel:*

$$p_{t_i}^{\Psi}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \mathcal{N}\Big(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\big(\boldsymbol{s}_{t_i}^{ref}(\boldsymbol{x}_{t_i}) + \boldsymbol{\Psi}_{t_i}\big),\ \sigma_{t_i}^2 \mathbf{I}\Big).$$

Then, the guidance term $\boldsymbol{\Psi}_{t_i}$ that matches the optimal transitional kernel $p_{t_i}^\star(\cdot \mid \boldsymbol{x}_{t_i})$ is

$$\boldsymbol{\Psi}_{t_i}^\star(\boldsymbol{x}_{t_i}) \triangleq \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\, \mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p_{t_i}^\star(\cdot|\boldsymbol{x}_{t_i})}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\right], \tag{12}$$

which is called the optimal value guidance.

The optimal value guidance is generally intractable due to the integration with respect to the posterior. Therefore, practical methods use an *optimal value guidance estimate* $\hat{\boldsymbol{\Psi}}_{t_i}(\boldsymbol{x}_{t_i}) \approx \boldsymbol{\Psi}_{t_i}^\star(\boldsymbol{x}_{t_i})$ and often apply a heuristic temporal weight $\gamma(t_i)$, yielding the *effective guidance* $\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i}) := \gamma(t_i)\hat{\boldsymbol{\Psi}}_{t_i}(\boldsymbol{x}_{t_i})$.

Since score networks are trained successfully by regression to target scores, the score function is parameterized by a score network and trained to satisfy $\boldsymbol{s}_t^\theta \approx \boldsymbol{s}_t$. In particular, Theorem 3.2 shows that existing methods can be unified by the regression problem of $\boldsymbol{s}_t^\theta$ toward the optimal value guidance estimate with $\boldsymbol{\Psi}_{t_i}$, yielding a common RSM objective. See Appendix C for proofs.

Theorem 3.2 (Reward Score Matching)**.** *For Soft RL methods and VGG-Flow, the loss $\mathcal{L}(\theta)$ admits the common form*

$$\mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}}\left[C_1(t_i)\Big(\|\boldsymbol{s}_{t_i}^\theta - (\boldsymbol{s}_{t_i}^{ref} + \boldsymbol{\Psi}_{t_i})\|^2 + C_2(t_i)\|\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger}\|^2\Big)\right] \tag{13}$$

where C_1, C_2 are weights, and $\boldsymbol{s}^\theta, \boldsymbol{s}^{\theta^\dagger}, \boldsymbol{s}^{ref}$ are the current, previous, and reference score functions [4].

[3] DiffusionNFT remains closely related but does not reduce exactly to the same formulation; see Appendix H.3.

[4] When $C_2(t) = 0$, Eq. (13) reduces to entropy-regularized optimal control (Appendix H.1).

Remark 3.3. In the first on-policy update[5], the old-policy anchor is inactive. In presence of PPO clipping, the update is suppressed by setting $\boldsymbol{\Psi}_{t_i} \leftarrow \mathbf{0}$ and $C_2(t_i) \leftarrow 0$.

The gradient of Eq. (13) reveals update dynamics. We derive a *Canonical Gradient Form*:

$$\nabla_\theta \mathcal{L}(\theta) = 2\mathbb{E}[\mathbf{J}_\theta(\boldsymbol{s}_{t_i}^\theta)^\top \mathcal{G}(\boldsymbol{x}_{t_i})], \tag{14}$$

where $\mathbf{J}_\theta$ denotes the Jacobian and

$$\mathcal{G}(\boldsymbol{x}_{t_i}) = C_1(t_i)\Bigg(-\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i}) + \underbrace{\left(\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\text{ref}}\right)}_{\text{KL Regularization}} + C_2(t_i)\underbrace{\left(\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger}\right)}_{\text{Trust Region}}\Bigg). \tag{15}$$

This decomposition highlights three competing forces: *Effective Guidance* steers generation toward high rewards, while *KL Regularization* and the *Trust Region* term prevent model collapse by anchoring the model to the reference and previous policies, respectively. Section 3.1 is written in the entropy-regularized setting for generality, but Eq. (15) also reconciles the $\alpha = 0$ case for SQDF [18] and policy-gradient methods; see Appendix C for details.

Eq. (15) also clarifies the hierarchy of RSM's design dimensions. The primary source of variation between methods lies in the way $\hat{\boldsymbol{\Psi}}_{t_i}$ is constructed. By contrast, temporal weighting and trust-region realization act only as secondary coefficient-level modifiers via $\gamma(t_i), C_1(t_i), C_2(t_i)$ and clipping. We therefore focus first on estimator design, and defer these coefficient-level choices to Section 4.

3.2 Estimating Optimal Value Guidance $\hat{\boldsymbol{\Psi}}_{t_i}$

This section, which is one of the most important contributions of this paper, shows that the seemingly different forms of reward-based finetuning methods can be unified as the zeroth and first order estimate of the current or lookahead estimator of the optimal value guidance estimate. Specifically, Eq. (12) suggests two practical families of value-guidance estimators: *current-state* and *lookahead*.

Current-state Estimator. Recall the first equality in Eq. (12):

$$\boldsymbol{\Psi}_{t_i}^\star = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) \tag{16}$$

This suggests a *current-state* estimator, available only in the first-order setting. VGG-Flow [26] falls in this family. Specifically, since we do not have access to the value function V but only to the terminal reward r, we approximate $\boldsymbol{\Psi}_{t_i}^\star$ directly at $\boldsymbol{x}_{t_i}$ via a Tweedie estimate.

Definition 3.4 (Current-state Estimator)**.** The first-order current-state estimator is

$$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{CS},1} = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}) \;\approx\; \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \tag{17}$$

This avoids the stochastic rollout but incurs substantial bias at low-SNR timesteps, as will be discussed later. It is therefore best viewed as a simpler but more biased alternative to lookahead estimators explained below, which subsumes most existing methods and exposes the main design tradeoffs.

Lookahead Estimator. Lookahead estimators roll out from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_j}$ with $j < i$ and extract reward information there. First-order estimators require differentiable rewards, whereas zeroth-order estimators apply to arbitrary black-box rewards.

Definition 3.5 (Lookahead Estimators)**.** For *lookahead depth* $j < i$ and branching width K_i, the first-order and zeroth-order lookahead estimators are

$$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},1} = \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\frac{1}{K_i}\sum_{k=1}^{K_i}\nabla_{\boldsymbol{x}_{t_{i-1}}^{(k)}} r(\hat{\boldsymbol{x}}_{0|t_j}^{(k)}) \approx \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1}:t_j}}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_j})\right], \tag{18}$$

$$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0} = \frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\frac{1}{K_i}\sum_{k=1}^{K_i} r(\hat{\boldsymbol{x}}_{0|t_j}^{(k)})\,\boldsymbol{\epsilon}_{t_i}^{(k)} \quad \approx \frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1}:t_j}}\left[r(\hat{\boldsymbol{x}}_{0|t_j})\,\boldsymbol{\epsilon}_{t_i}\right], \tag{19}$$

where $\hat{\boldsymbol{x}}_{0|t_j}$ denotes the Tweedie estimate of the posterior mean:

$$\hat{\boldsymbol{x}}_{0|t_j} = \mathbb{E}[\boldsymbol{x}_0 \mid \boldsymbol{x}_{t_j}] := \mathbb{E}_{\boldsymbol{x}_0 \sim p_{t_j}(\boldsymbol{x}_0|\boldsymbol{x}_{t_j})}[\boldsymbol{x}_0]\,. \tag{20}$$

[5]Throughout this paper, we use the relaxed notion of *on-policy* common in online RL algorithms, where data is collected under the old policy $\theta^\dagger$ and reused for one or more updates (e.g., DDPO [3]). When we need to refer specifically to the case $\theta = \theta^\dagger$, we distinguish it explicitly, e.g., by referring to it as the *first on-policy update* or *strictly on-policy*.

Existing first-order lookahead methods (Residual ∇-DB [27], SQDF [18]) use one-step lookahead ($j = i - 1$), and therefore inherit substantial Tweedie bias from $\hat{\boldsymbol{x}}_{0|t_j}$ at low-SNR timesteps, akin to current-state methods ($j = i$).

Several first-order methods (Residual ∇-DB, VGG-Flow) attempt to decompose the soft value gradient into a Tweedie approximation, and learn the residual Jensen gap via a small neural network g_ϕ, which can be explained by Proposition 3.6. See Appendix B.3 for the proof.

Proposition 3.6 (Conditional Consistency of First-Order Estimators)**.** *Let $j \in \{i, i-1\}$. Suppose that the soft value gradient at timestep t_j admits the exact decomposition*

$$\nabla_{\boldsymbol{x}_{t_j}} V_{t_j}(\boldsymbol{x}_{t_j}) = \gamma_{t_i} \nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j}). \tag{21}$$

Define the residual-corrected first-order estimator

$$\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) := \left(\frac{1}{\alpha}\iota[j = i] + \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\iota[j = i-1]\right)\Big(\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j})\Big), \tag{22}$$

where $\iota[\,\cdot\,]$ denotes the indicator function, and for $j = i-1$ we take $\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$, while for $j = i$ we simply set $\boldsymbol{x}_{t_j} = \boldsymbol{x}_{t_i}$. Then

$$\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) \,\middle|\, \boldsymbol{x}_{t_i}\right] = \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}), \tag{23}$$

where the expectation is trivial in the current-state case $j = i$. That is, under an exact residual parameterization g_ϕ, both the current-state first-order estimator and the one-step first-order estimator are unbiased for the optimal value guidance.

Under an exact residual parameterization, the resulting estimator is unbiased. However, as we later verify in Appendix G.1, auxiliary corrections g_ϕ turns out to be effectively negligible in the existing methods, so the resulting first-order estimators remain biased.

Existing zeroth-order (policy-gradient) methods typically use full rollout ($j = 0$)[6]. This full-rollout zeroth-order estimator is unbiased for the soft value gradient. Notably, the following theorem, originated from Stein's identity [38], completes the theoretical unification of zeroth-order and first-order value-guidance methods. The proof can be found in Appendix B.4.

Theorem 3.7 (Consistency of Policy Gradient)**.** *For the full-rollout $j = 0$, the lookahead zeroth-order estimator is an unbiased estimator of the soft value gradient:*

$$\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i} \approx \frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{\epsilon}_{t_i:t_0} \overset{\mathrm{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}] = \mathbb{E}_{\boldsymbol{\epsilon}_{t_i}}[\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})] = \boldsymbol{\Psi}^\star_{t_i}. \tag{24}$$

Approximating the lookahead expectation with sampled descendants introduces two further design choices to reduce variance. *Branching* specifies per-step branch widths K_i, with $K_i = 1$ denoting a point estimate at t_i; larger K_i reduce variance at higher compute cost [12, 22]. *Stochasticity localization* changes the underlying stochastic control problem by keeping only selected reverse steps stochastic [12, 21]. This reduces the accumulation of variance, giving a more cost-effective estimate of $\boldsymbol{\Psi}^\star_{t_i}$.

4 Practical Design Choices of RSM from Optimization Perspectives

Section 3 established exact reductions: many existing methods can be written under a common RSM objective. We now shift from this probabilistic view to a practical optimization view, asking which design choices matter most under finite compute and noisy gradient estimates. From this perspective, performance is governed by three coupled design dimensions: *estimator design*, *temporal weighting*, and *trust-region realization*. Estimator design controls the quality of the value-guidance estimate, while temporal weighting and trust-region realization determine how strongly each timestep is optimized and regularized. Existing methods therefore differ mainly in how they allocate compute, optimization mass, and regularization across timesteps.

[6] In principle, first-order consistency can also be achieved by full rollout, i.e. by selecting $j = 0$ and differentiating the terminal reward through the entire denoising trajectory. However, this requires the full Jacobian chain and is therefore computationally impractical and potentially unstable in high dimension.

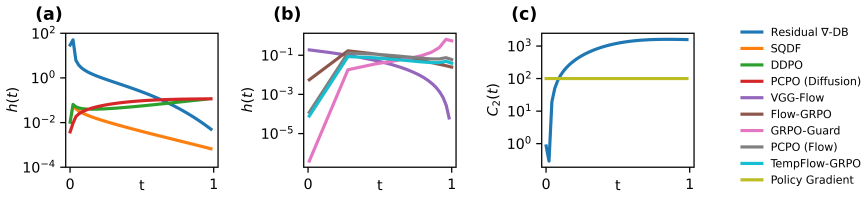

Figure 1: **Temporal Optimization Strength.** (a) Successful first-order methods reduce value guidance at low-SNR timesteps. (b) Improved zeroth-order methods reduce value guidance at high-SNR timesteps. (c) Residual ∇-DB enforces stronger trust-region constraints for low-SNR timesteps. Policy Gradient's $C_2(t)$ is depicted for constant $r(\boldsymbol{x}_0) = 1$ and $\alpha = 10^{-2}$.

Estimator design determines the quality of $\hat{\boldsymbol{\Psi}}_{t_i}$. Its main practical knobs are lookahead depth j, branching width $\{K_i\}$, and stochasticity localization. Deeper lookahead reduces bias at higher compute cost, while larger branching widths reduce finite-sample variance at higher cost. Stochasticity localization can likewise improve estimator quality, but unlike branching, it also changes the underlying stochastic control problem. Under fixed compute, estimator design therefore asks how much budget to spend on deeper rollouts, more descendants, and where to place stochastic steps. This induces two allocation problems: how to divide compute within each timestep to obtain a useful guidance estimate, and how to distribute total estimator budget across timesteps.

An important example is the zeroth-order estimator, whose exact conditional variance under independent descendant rollouts is $\frac{1}{K_i}\left(\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\right)^2 \mathrm{Cov}(r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i} \mid \boldsymbol{x}_{t_i})$. Accordingly, conditioned on $\boldsymbol{x}_{t_i}$, the covariance of the full-rollout zeroth-order estimator $\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i}$ scales as:

$$\mathrm{Cov}\left(\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i} \mid \boldsymbol{x}_{t_i}\right) \propto \frac{1}{K_i}\left(\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\right)^2. \tag{25}$$

Note that this scaling should be read as leading-order scaling. This scaling shows that, under a fixed branching width K_i, some timesteps are intrinsically noisier than others and may therefore require larger K_i to reduce variance, at the cost of additional compute.

Estimator design also includes reward centering; for instance, subtracting the sample mean acts as a control variate that reduces estimator variance without changing the underlying target guidance.

Temporal weighting. From an optimization perspective, the key question is not only how $\hat{\boldsymbol{\Psi}}_{t_i}$ is estimated, but how strongly its reward signal enters the actual update at each timestep. Recall from the Canonical Gradient Form in Eq. (15) that the effective guidance term appears through $\boldsymbol{\Psi}_{t_i}$, and that practical methods define $\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i}) := \gamma(t_i)\hat{\boldsymbol{\Psi}}_{t_i}(\boldsymbol{x}_{t_i})$. We therefore seek a common coefficient that measures the timestep-wise optimization strength of the underlying reward signal.

For current-state estimators, Eq. (17) shows that $\hat{\boldsymbol{\Psi}}_{t_i}$ is proportional to a reward gradient at the current state. For lookahead estimators, combining Theorem 3.7 with Eqs. (18)–(19) yields the common reward-gradient view

$$\widehat{\nabla r(\boldsymbol{x}_{t_i})} = \frac{1}{\sigma_{t_i}}\mathbb{E}_{\boldsymbol{x}_{t_i:t_0}}\left[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}\right] = \mathbb{E}_{\boldsymbol{x}_{t_i:t_0}}\left[\nabla_{\boldsymbol{x}_{t_i-1}} r(\boldsymbol{x}_0)\right]. \tag{26}$$

Thus, both differentiable-reward and black-box-reward methods can be compared through a common reward-gradient signal. Accordingly, the canonical gradient form in Eq. (14) can be represented with respect to the common reward-gradient Eq. (26).

More specifically, by using additional conversion factor $\delta(t)$ between score guidance and actual optimization dynamics (see Appendix D.2) such that $-\delta(t)(\boldsymbol{\epsilon}^\theta_t - \boldsymbol{\epsilon}^{\mathrm{ref}}_t) = \boldsymbol{s}^\theta_t$ for $\boldsymbol{\epsilon}$-prediction or $-\delta(t)(\boldsymbol{v}^\theta_t - \boldsymbol{v}^{\mathrm{ref}}_t) = \boldsymbol{s}^\theta_t$ for $\boldsymbol{v}$-prediction, respectively, the canonical gradient can be represented by

$$\nabla_\theta \mathcal{L}(\theta) = 2\mathbb{E}[\mathbf{J}_\theta(\boldsymbol{g}^\theta_{t_i})^\top \tilde{\mathcal{G}}(\boldsymbol{x}_{t_i})], \tag{27}$$

where

$$\tilde{\mathcal{G}}(\boldsymbol{x}_{t_i}) = h(t_i)\widehat{\nabla r(\boldsymbol{x}_{t_i})} + \delta(t_i)^2 C_1(t_i)\Bigg(\underbrace{(\boldsymbol{g}^\theta_{t_i} - \boldsymbol{g}^{\mathrm{ref}}_{t_i})}_{\text{KL Regularization}} + C_2(t_i)\underbrace{(\boldsymbol{g}^\theta_{t_i} - \boldsymbol{g}^{\theta^\dagger}_{t_i})}_{\text{Trust Region}}\Bigg). \tag{28}$$

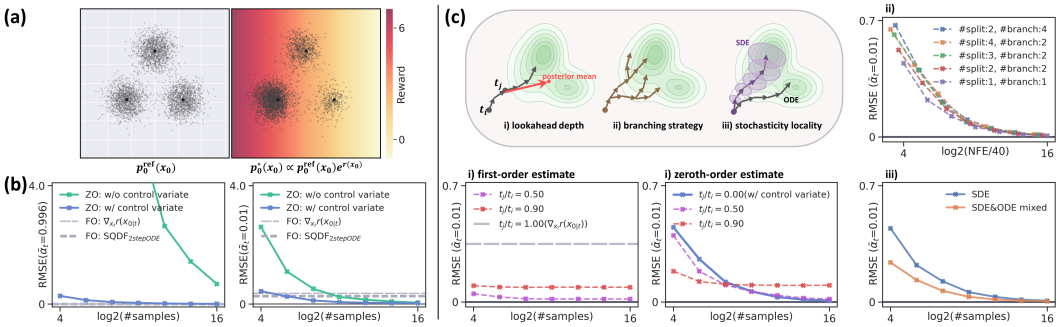

Figure 2: **Toy analysis of estimator quality under fixed compute.** (a) Reference distribution and its reward-tilted target. (b) RMSE of representative first-order (FO) and zeroth-order (ZO) estimators[7] at two timesteps. (c) RMSE of various estimators by sample size, for different lookahead depths, branching strategies, and stochasticity localizations. *#split* is the number of recursive branching stages, and *#branch* is the number of child trajectories spawned at each stage. Combining SDE and ODE for sampling is done in a same manner of Tempflow-GRPO [12].

where $\boldsymbol{g}^\theta$ denotes the neural network parameterization (e.g., $\boldsymbol{g}^\theta = \boldsymbol{\epsilon}^\theta$ for $\boldsymbol{\epsilon}$-prediction or $\boldsymbol{g}^\theta = \boldsymbol{v}^\theta$ for $\boldsymbol{v}$-prediction, respectively), and $h(t_i)$ refers to the combined timestep-wise coefficient of the reward-gradient signal in the update as the *Normalized Influence Metric*:

$$h(t_i) := \delta(t_i) C_1(t_i) \times \begin{cases} \gamma(t_i)/\alpha, & \text{(current-state)}, \\ \gamma(t_i)\sigma_{t_i}^2/(\alpha\Omega(t_i)), & \text{(lookahead)}. \end{cases} \tag{29}$$

In other words, $h(t)$ measures how strongly the reward gradient enters optimization dynamics after accounting for both temporal weighting and parameterization. Plots for representative methods are shown in Fig. 1.

This view reveals a clear empirical contrast. Successful first-order methods place most of the mass of $h(t)$ on high-SNR timesteps, suppressing low-SNR timesteps where its estimators are most biased. For zeroth-order methods, the situation is reversed: the estimator is unbiased, but high-SNR timesteps have higher variance under fixed branching width (Eq. (25)). Thus, improved methods place more mass of $h(t)$ on low-SNR timesteps.

Temporal weighting also includes any normalization that rescales rewards or advantages. Unlike centering, which acts as a control variate, dividing by a sample-dependent scale changes the effective guidance magnitude and therefore belongs conceptually with temporal weighting.

Trust-region realization determines how much of the update signal survives. In our formulation, it appears through $C_2(t)$ and clipping. Even with the same estimator quality and temporal weighting, different trust-region realizations can produce very different effective updates across timesteps. Timestep emphasis therefore depends not only on how strongly a timestep is weighted, but also on how much of its proposed update survives regularization and clipping. See Table 1 for the existing choices of this parameter.

5 Experiments

Inspired by our theoretical findings, we study the practical design space of RSM in two stages. First, we isolate estimator design at a single timestep under fixed compute, asking which local allocations estimate $\boldsymbol{\Psi}_{t_i}^\star$ most effectively. Second, we study end-to-end design under fixed total compute, asking how estimator design, temporal weighting, and trust-region realization should be chosen jointly across timesteps.

5.1 Isolated Estimator Analysis

Setup. We construct a two-dimensional toy problem with analytically tractable $\boldsymbol{\Psi}_{t_i}^\star$, where a Gaussian-mixture reference distribution $p_0^{\text{ref}}(\boldsymbol{x}_0)$ is reward-tilted into $p_0^\star(\boldsymbol{x}_0)$. We then compare

[7] $\text{SQDF}_{2stepODE}$ refers to the 2-step ODE estimator proposed in Kang et al. [18].

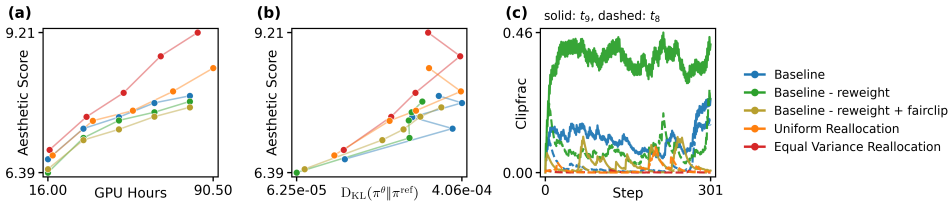

Figure 3: **Improving high-SNR timesteps is better than merely suppressing them.** Making clipping timestep-fair and reallocating budget improves reward efficiency under matched compute. (a) Aesthetic Score vs. GPU hours. (b) Aesthetic Score vs. KL divergence. (c) Clip fraction for t_9 (solid) and t_8 (dashed).

estimators $\hat{\Psi}_{t_i}^{\mathrm{LA}}$ from Eqs. (18)–(19) against this ground truth by RMSE under matched compute. See Appendix E.1 for details.

Results. Fig. 2(b) shows that one-step lookahead first-order guidance is severely biased at low SNR. By contrast, full-rollout zeroth-order guidance is unbiased, but requires many more samples for reliable estimates. Reward centering further reduces zeroth-order error at fixed sample count.

Fig. 2(c-i) shows that lookahead depth j induces a three-way tradeoff among bias, variance, and compute. Deeper lookahead reduces bias, but under limited samples shallower lookahead can be both more accurate and more compute-efficient, since it averages over posterior uncertainty and therefore has lower variance than full rollout; this is especially clear in the zeroth-order results. At the same lookahead depth, the first-order estimator typically improves faster with sample count than the zeroth-order estimator[8].

Fig. 2(c-ii) shows that, under fixed total sample budget, different branching patterns yield similar error–NFE trends[9]. Fig. 2(c-iii) shows that stochasticity localization improves estimator quality by making the induced value function easier to predict under fixed compute [10].

Findings. Under finite compute, the best local estimator need not be the least biased one: shallower lookahead estimators may outperform unbiased full-rollout estimators when variance and cost dominate. Our results also suggest that point estimates are too noisy, but once branching is used at all, the specific (#split, #branch) choice matters relatively little under matched sample budget. We also find that subtracting the sample mean is crucial to reduce estimate variance.

5.2 End-to-End Design

We now test whether aforementioned design principles improve end-to-end optimization under matched compute. Experiment details are in Appendix E; note that our proposed improvements require comparable or significantly less compute per epoch compared to baselines. More results are in Appendices F.1 and I.

Mechanistic Analysis. We begin with a diagnostic case study on zeroth-order flow matching, using TempFlow-GRPO as the baseline. By Eq (25), we can conjecture that zeroth-order guidance at high-SNR timesteps is intrinsically noisy under fixed branching width. TempFlow-GRPO nevertheless assigns substantial budget to these timesteps, while relying on strong clipping and downweighting to contain the resulting instability. This suggests a mismatch between where compute is spent and where guidance is actually reliable.

We therefore redesign the update in three steps. First, we make clipping comparable across timesteps by moving the timestep-dependent scale outside the clipping operator:

$$\mathrm{clip}_{\xi}\left(\frac{\|\mu_{t_i}^{\theta}-\mu_{t_i}^{\theta^{\dagger}}\|^2}{2\tilde{\sigma}_{t_i}^2\Delta t_i}\right) \longrightarrow \frac{1}{\tilde{\sigma}_{t_i}^2\Delta t_i}\mathrm{clip}_{\xi}\left(\frac{\|\mu_{t_i}^{\theta}-\mu_{t_i}^{\theta^{\dagger}}\|^2}{2}\right). \tag{30}$$

[8] In this toy analysis, first-order curves use the full Jacobian chain to compute the exact first-order signal.

[9] Note that branching improves the conditional estimate for a given parent latent, but under fixed sample budget it reduces the number of distinct parent latents explored at that timestep. This tradeoff is not fully captured by our toy analysis, but is important in practice.

[10] As discussed in Section 3.2, this benefit comes partly from changing the underlying stochastic control problem rather than merely reducing variance for the same target.

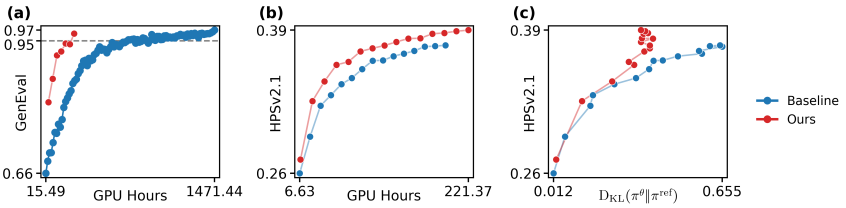

Figure 4: **Validation: Zeroth-order methods.** Principled budget allocation and temporal weighting improve performance on (a) GenEval with SD3.5-M[13] and (b, c) HPSv2.1 with SD1.5.

Second, we reallocate branching budget toward the high-SNR region to reduce estimator variance where it is largest. Third, once this redistribution is applied, we find that t_9 remains too noisy and too heavily clipped to justify further investment, so we deactivate it and reallocate its budget to the remaining steps.

Fig. 3 shows that uniform redistribution already improves over TempFlow-GRPO, and a variance-aware redistribution performs best. Under matched compute, this redesign improves reward more quickly while maintaining a comparable reward–KL tradeoff. The resulting lesson is simple: budget should be concentrated on timesteps whose guidance can be made reliable, rather than on timesteps whose updates are later suppressed.

Validation Across Settings. In zeroth-order flow matching, the baseline bottleneck is uniform budget allocation across timesteps, even though the useful region of the trajectory depends on the reward. For GenEval [11], the reward is primarily semantic, so investing heavily in high-SNR timesteps is unnecessary; those steps are more tied to fine details than to the semantic structure the reward emphasizes. We therefore concentrate branching budget on the first few low-SNR timesteps, where semantic guidance is most useful. This redesign reaches GenEval $= 0.97$ with a $5\times$ wall-clock speedup over TempFlow-GRPO (Fig. 4(a)).

In zeroth-order diffusion, the baseline bottleneck is the absence of branching: PCPO [20] uses a single sampled continuation at every reverse step, so each timestep receives a noisy point estimate of the reward signal. Ideally, one would branch at every reverse step, but doing so over the full trajectory ($N = 50$) is prohibitively expensive. We therefore introduce branching only at a sparse subset of steps and leave the rest unbranched, concentrating reward-bearing updates where they are most useful while keeping total compute manageable. This sparse-allocation design improves reward substantially faster while achieving a better reward–KL tradeoff (Fig. 4(b, c)).

In first-order methods, the baseline bottleneck is local Tweedie-based guidance, but this is partly obscured in the original evaluation settings, which are mostly aesthetic-heavy. Prior first-order methods [18, 26, 27] are evaluated either on purely aesthetic rewards or on HPSv2.1 [47] with prompt subsets such as `hpd_photo_painting`, where semantic demands are relatively weak. When we instead fine-tune on HPSv2.1 using GenEval training prompts, so that both aesthetic quality and semantic fidelity matter, performance drops sharply under original designs. In this setting, low-SNR timesteps can no longer be ignored: local Tweedie-based gradients become increasingly biased there, yet meaningful learning still has to occur at those steps. We therefore replace these local gradients with terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$, effectively moving to $j = 0$, which yields a practical *reward score distillation* update while reversing the inherited low-SNR downweighting[11]. Across both flow-matching and diffusion backbones, this yields substantially faster reward improvement while remaining competitive in the reward–KL tradeoff (Fig. 5)[12].

[11] Although the exact quantity $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$ would provide an unbiased first-order guidance signal, computing it requires a high-order Jacobian chain through the full denoising trajectory, which is both computationally expensive and potentially destabilizing, as noted in Poole et al. [31]. We therefore use the practical surrogate $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$; see Appendix E for details.

[12] We display *transition drift* for first-order flow matching, since KL divergence is undefined for the VGG-Flow baseline [26]; see Appendices E.2 and F.1.

[13] Due to resource constraints, we did not rerun the GenEval baseline; instead, we plot reverse-engineered data from Figure 3 of He et al. [12]. As a result, reward–KL tradeoff comparisons are unavailable for GenEval.

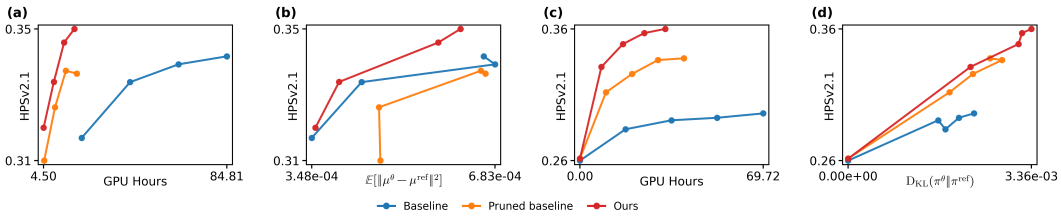

Figure 5: **Validation: First-order methods.** Improved reward guidance for low-SNR timesteps yields faster reward gains, while maintaining a competitive reward–KL tradeoff on (a, b) SD3.5-M and (c, d) SD1.5. See Appendix F.2 for more results.

6 Discussion

Broadening the framework. RSM covers most affine flow-based RL fine-tuning methods, but several related objectives lie slightly outside its most direct formulation. Reward-weighted MLE methods such as ReFL [48] also fit within the broader Soft RL picture, although we did not study them here because they currently underperform the methods considered in this paper. Our method is also closely related to entropy-regularized optimal control in the absence of trust-region regularization (Appendix H.1), implying further unification with methods based on stochastic optimal control [7, 16, 43]. Related ideas may also extend beyond training-time alignment: Appendix H.2 shows that inference-time methods such as DNO [40] rely on closely related zeroth-order principles, but update the initial noise rather than the reverse-generation policy. DiffusionNFT is likewise closely related, as a form of reward-weighted regression connected to Soft RL; Appendix H.3 discusses this relation in more detail. Within first-order methods, another promising extension is to learn stronger value-gradient critics: existing auxiliary networks g_ϕ often collapse under weak supervision, and better bootstrapped or consistency-based targets may provide lower-variance guidance without the bias of shallow lookahead.

Better tuning of the design space. Our results show that the practical design space remains far from exhausted. In this paper, we did not perform exhaustive tuning or aggressive engineering; rather, we tested a small number of principled modifications, and even these already produced large gains. This suggests substantial headroom for more systematic tuning of RSM's design space. Further simplification is also possible, for example by absorbing the trust-region term C_2 into temporal weighting, as in GRPO-Guard [45] (see Eq. (106)). In addition, we optimized these choices under fixed samplers, even though sampler coefficients directly affect estimator variance and native timestep weights. Joint optimization of the sampler and training objective, as in PCPO (diffusion) [20], is therefore another promising direction.

Limitations. Although our redesign improves upon existing reward-based fine-tuning methods, it does not eliminate reward hacking. Our auxiliary evaluations suggest that the gains reported here are not driven by *more* aggressive reward hacking, but they do not establish emergent robustness to reward misspecification. More broadly, RSM does not unify *all* RL fine-tuning methods for flow-based models. It covers a broad class of score-matching-based approaches, but several important objectives remain outside its current scope, including offline preference-based methods such as Diffusion-DPO [44].

Broader Impact. Improving alignment of flow-based models enhances their ability to generate data with desired properties, hence better accommodating human needs throughout various application domains.

7 Conclusion

We introduced Reward Score Matching (RSM), a unified view of reward-based fine-tuning for flow and diffusion models. Under this view, many methods that are motivated from different perspectives reduce to the same underlying score-matching problem, with their main differences arising from how value guidance is estimated, weighted, and regularized across timesteps.

This perspective also clarifies the practical structure of the design space. In particular, it puts first-order and zeroth-order methods on common ground as different estimators of the same target,

exposing shared bias–variance–compute tradeoffs that are otherwise obscured by method-specific derivations. It further shows that much of the remaining variation can be understood through three coupled design dimensions: estimator design, temporal weighting, and trust-region realization.

Guided by this framework, we identified several simplifications and principled redesigns that improve reward-based fine-tuning across representative differentiable-reward and black-box-reward settings, often with substantially better compute efficiency. Taken together, these results suggest that progress in this area depends less on introducing increasingly specialized objectives and more on understanding how to estimate, weight, and preserve reward-guided updates effectively. We aspire RSM to provide a common language for comparing existing methods, and a practical foundation for designing stronger ones.

References

[1] Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209): 1–80, 2025. URL http://jmlr.org/papers/v26/23-1605.html.

[2] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957.

[3] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

[4] Jaemoo Choi, Yuchen Zhu, Wei Guo, Petr Molodyk, Bo Yuan, Jinbin Bai, Yi Xin, Molei Tao, and Yongxin Chen. Rethinking the design space of reinforcement learning for diffusion models: On the importance of likelihood estimation beyond loss design, 2026. URL https://arxiv.org/abs/2602.04663.

[5] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11472–11481, June 2022.

[6] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

[7] 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.

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024. URL https://arxiv.org/abs/2403.03206.

[9] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=8OTPepXzeh.

[10] Stephanie Fu, Netanel Yakir Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=DEiNSfh1k7.

[11] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=Wbr51vK331.

[12] Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7mCo3R3Wyn.

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[15] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[16] Jian Huang, Yuling Jiao, Lican Kang, Xu Liao, Jin Liu, and Yanyan Liu. Schrödinger-föllmer sampler. *IEEE Transactions on Information Theory*, 71(2):1283–1299, 2025. doi: 10.1109/TIT.2024.3522494.

[17] Nai-Chieh Huang, Ping-Chun Hsieh, Kuo-Hao Ho, and I-Chen Wu. PPO-Clip attains global optimality: Towards deeper understandings of clipping. In *AAAI*, pages 12600–12607, 2024.

[18] Hyeongyu Kang, Jaewoo Lee, Woocheol Shin, Kiyoung Om, and Jinkyoo Park. Diffusion fine-tuning via reparameterized policy gradient of the soft q-function, 2026. URL https://openreview.net/forum?id=8zoxC9e23q.

[19] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: an open dataset of user preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[20] Jeongjae Lee and Jong Chul Ye. PCPO: Proportionate credit policy optimization for preference alignment of image generation models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=alY08iknli.

[21] Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde, 2025. URL https://arxiv.org/abs/2507.21802.

[22] Yuming Li, Yikai Wang, Yuying zhu, Zhongyu Zhao, Ming Lu, Qi She, and Shanghang Zhang. BranchGRPO: Stable and efficient GRPO with structured branching in diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T2nP2IQasd.

[23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[24] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oCBKGw5HNf.

[25] Qingming Liu, Zhen Liu, Dinghuai Zhang, and Kui Jia. Nabla-r2d3: Effective and efficient 3d diffusion alignment with 2d rewards. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Dk2qprCnu8.

[26] Zhen Liu, Tim Z. Xiao, Carles Domingo-Enrich, Weiyang Liu, and Dinghuai Zhang. Value gradient guidance for flow matching alignment. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=6MmOy2Ji8V.

[27] Zhen Liu, Tim Z. Xiao, Weiyang Liu, Yoshua Bengio, and Dinghuai Zhang. Efficient diversity-preserving diffusion alignment via gradient-informed GFlownets. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Aye5wL6TCn.

[28] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, June 2025. ISSN 2731-5398. doi: 10.1007/s11633-025-1562-4. URL http://dx.doi.org/10.1007/s11633-025-1562-4.

[29] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *CoRR*, abs/1705.07798, 2017. URL http://arxiv.org/abs/1705.07798.

[30] Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, 2023. URL https://openreview.net/forum?id=beHp3L9KXc.

[31] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FjNys5c7VyY.

[32] Jannes Quer and Enric Ribera Borrell. Connecting stochastic optimal control and reinforcement learning. *Journal of Mathematical Physics*, 65(8), 2024.

[33] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[35] Chrisoph Schuhmann. Laion aesthetics, Aug 2022. URL https://laion.ai/blog/laion-aesthetics/.

[36] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[38] Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981.

[39] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[40] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=JpbqiD7n9r.

[41] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review, 2024. URL https://arxiv.org/abs/2407.13734.

[42] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

[43] Francisco Vargas, Andrius Ovsianas, David Lopes Fernandes, Mark Girolami, Neil D Lawrence, and Nikolas Nüsken. Bayesian learning via neural schrödinger-föllmer flows. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2022. URL https://openreview.net/forum?id=1Fqd10N5yTF.

[44] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8228–8238, June 2024.

[45] Jing Wang, Jiajun Liang, Jie Liu, Henglin Liu, Gongye Liu, Jun Zheng, Wanyuan Pang, Ao Ma, Zhenyu Xie, Xintao Wang, Meng Wang, Pengfei Wan, and Xiaodan Liang. Grpo-guard: Mitigating implicit over-optimization in flow matching via regulated clipping, 2025. URL https://arxiv.org/abs/2510.22319.

[46] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[47] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Lai. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023. URL https://arxiv.org/abs/2306.09341.

[48] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15903–15935, 2023.

[49] Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models, 2025. URL https://arxiv.org/abs/2509.25050.

[50] Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. Dancegrpo: Unleashing grpo on visual generation, 2025. URL https://arxiv.org/abs/2505.07818.

[51] Dinghuai Zhang, Yizhe Zhang, Jiatao Gu, Ruixiang Zhang, Joshua M. Susskind, Navdeep Jaitly, and Shuangfei Zhai. Improving GFlownets for text-to-image diffusion alignment. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=XDbY3qhM42.

[52] Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VJZ477R89F.

Table 2: **Notation guide.** Main symbols used in the unified RSM framework.

Symbol	Meaning	Main role
$\boldsymbol{\Psi}^{\star}_{t_i}$	Optimal value guidance, $\frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}$	Ideal reward-guided correction to the reference score
$\hat{\boldsymbol{\Psi}}_{t_i}$	Practical estimate of $\boldsymbol{\Psi}^{\star}_{t_i}$	Guidance estimate before temporal reweighting
$\boldsymbol{\Psi}_{t_i}$	Effective guidance, $\gamma(t_i)\hat{\boldsymbol{\Psi}}_{t_i}$	Guidance actually used in the loss
$\gamma(t_i)$	Heuristic temporal weight	Controls timestep-wise guidance strength
$C_1(t_i)$	Main timestep weight in Eq. (13)	Scales the score-matching update at t_i
$C_2(t_i)$	Trust-region coefficient in Eq. (13)	Scales anchoring to the previous policy / score
$h(t_i)$	Normalized Influence Metric	Effective timestep-wise optimization strength
$\Omega(t_i)$	Score coefficient in the reverse transition kernel	Converts score guidance into mean shift
$\delta(t_i)$	Parameterization-conversion factor	Maps native outputs to score differences
j	Lookahead depth	Sets how far rollout proceeds before reward extraction
K_i	Branching width at timestep t_i	Number of descendants used for conditional estimation

A Notation Guide

Table 2 summarizes the main symbols used throughout the paper.

B Main Proofs

B.1 Proof of Discretized affine flow SDE

Proof. Consider a conditional affine flow $\boldsymbol{x}_t := \psi_t(\boldsymbol{x}_1|\boldsymbol{x}_0) = a_t\boldsymbol{x}_0 + b_t\boldsymbol{x}_1$ with $\boldsymbol{x}_1 \sim \mathcal{N}(0, \mathbf{I})$. The corresponding flow ODE is described as

$$\begin{aligned} d\boldsymbol{x}_t = \boldsymbol{v}_t(\boldsymbol{x}_t, t)dt &= \frac{\dot{a}_t}{a_t}\boldsymbol{x}_t dt + \left(\frac{\dot{a}_t b_t^2 - a_t\dot{b}_t b_t}{a_t}\right)\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)dt \\ &= A(t)\boldsymbol{x}_t dt + B(t)\boldsymbol{s}(\boldsymbol{x}_t, t). \end{aligned} \tag{31}$$

From Song et al. [37], the corresponding flow SDE that matches the same marginal density across time t is

$$d\boldsymbol{x}_t = [\boldsymbol{v}_t(\boldsymbol{x}_t, t) - \frac{1}{2}g_t\boldsymbol{s}(\boldsymbol{x}_t, t)]dt + \sqrt{g_t}d\boldsymbol{w}_t \tag{32}$$

$$= [A(t)\boldsymbol{x}_t + (B(t) - \frac{1}{2}g_t)\boldsymbol{s}(\boldsymbol{x}_t, t)]dt + \sqrt{g_t}d\boldsymbol{w}_t \tag{33}$$

$$= [A(t)\boldsymbol{x}_t + \tilde{B}(t)\boldsymbol{s}(\boldsymbol{x}_t, t)]dt + \sqrt{g_t}d\boldsymbol{w}_t. \tag{34}$$

where we set $\tilde{B}(t) = B(t) - 0.5g_t$. Note that $\tilde{B}(t) = B(t)$ when $g_t = 0$ (i.e. ODE).

Consider an integrating factor $\Phi(t) = \exp\left(\int A(\tau)d\tau\right)$. For $\boldsymbol{y}_t = \Phi(t)^{-1}\boldsymbol{x}_t$, we apply Ito derivative and obtain

$$d\boldsymbol{y}_t = \Phi(t)^{-1}d\boldsymbol{x}_t + \boldsymbol{x}_t d\Phi(t)^{-1} \tag{35}$$

$$= \Phi(t)^{-1}[A(t)\boldsymbol{x}_t dt + \tilde{B}(t)\boldsymbol{s}(\boldsymbol{x}_t, t)dt + \sqrt{g_t}d\boldsymbol{w}_t] - A(t)\Phi(t)^{-1}\boldsymbol{x}_t dt \tag{36}$$

$$= \Phi(t)^{-1}[\tilde{B}(t)\boldsymbol{s}(\boldsymbol{x}_t, t)dt + \sqrt{g_t}d\boldsymbol{w}_t], \tag{37}$$

where the first equality holds as $d\Phi(t)^{-1}d\boldsymbol{x}_t \approx 0$. Thus, by integrating both sides from s to t, we have

$$\boldsymbol{x}_t = \frac{\Phi(t)}{\Phi(s)}\boldsymbol{x}_s + \frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\tilde{B}(r)\boldsymbol{s}(\boldsymbol{x}_r, r)dr + \frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\sqrt{g_r}d\boldsymbol{w}_r \tag{38}$$

$$\approx \frac{\Phi(t)}{\Phi(s)}\boldsymbol{x}_s + \left[\frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\tilde{B}(r)dr\right]\boldsymbol{s}(\boldsymbol{x}_s, s) + \frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\sqrt{g_r}d\boldsymbol{w}_r, \tag{39}$$

where we assume $\boldsymbol{s}(\boldsymbol{x}_r, r) \approx \boldsymbol{s}(\boldsymbol{x}_s, s)$ which corresponds to the *first-order exponential integrator* underlying DPM-solver. Note that higher-order DPM-solvers use higher-order approximations here.

For conditional flow models, $A(t) = \dot{a}_t/a_t$ and $\Phi(t) = a_t$ by definition. Thus, the discretized SDE simplifies to:

$$\boldsymbol{x}_t = \frac{a_t}{a_s}\boldsymbol{x}_s + \left[\frac{a_t}{a_s}\int_s^t \frac{a_s}{a_r}\tilde{B}(r)dr\right]\boldsymbol{s}(\boldsymbol{x}_s, s) + a_t\int_s^t \frac{1}{a_r}\sqrt{g_r}d\boldsymbol{w}_r, \tag{40}$$

where the first term denotes a scaled previous sample, the second term denotes a score function, and the third term denotes the Gaussian noise with zero mean and covariance $\sigma_s^2\mathbf{I}$ where $\sigma_s^2 = a_t^2\int_s^t \frac{g_r}{a_r^2}dr$. Hence, by setting $s = t_i$ and $t = t_{i-1}$, we can express the general form of transition density $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i})$ as

$$p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) = \mathcal{N}(\underbrace{\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}(\boldsymbol{x}_{t_i})}_{=:\boldsymbol{\mu}_{t_i}(\boldsymbol{x}_{t_i})}, \sigma_{t_i}^2\mathbf{I}). \tag{41}$$

□

B.2 Proof of Proposition 3.1

We first need the following lemma for the zero-mean score identity.

Lemma B.1 (Zero-mean score identity under tail regularity)**.** *Fix $\boldsymbol{x} \in \mathbb{R}^d$ and let $p(\cdot \mid \boldsymbol{x})$ be a conditional density on $\mathbb{R}^d$ (with respect to Lebesgue measure) such that:*

1. *$\int_{\mathbb{R}^d} p(\boldsymbol{u} \mid \boldsymbol{x})\, d\boldsymbol{u} = 1$;*
2. *$p(\cdot \mid \boldsymbol{x})$ is differentiable in $\boldsymbol{u}$, and $\int_{\mathbb{R}^d} \left|\partial_{u_j} p(\boldsymbol{u} \mid x)\right| d\boldsymbol{u} < \infty$ for $\forall j \in \{1, \ldots, d\}$;*
3. *For $\forall j$, for almost every $\boldsymbol{u}_{-j} \in \mathbb{R}^{d-1}$, $\lim_{\boldsymbol{u}_j \to \pm\infty} p(u_j, \boldsymbol{u}_{-j} \mid \boldsymbol{x}) = 0$.*

Then the conditional score has zero mean:

$$\mathbb{E}_{\boldsymbol{u}\sim p(\cdot|\boldsymbol{x})}\left[\nabla_{\boldsymbol{u}} \log p(\boldsymbol{u} \mid \boldsymbol{x})\right] = \mathbf{0}. \tag{42}$$

Proof. For each j,

$$\mathbb{E}_{p(\cdot|\boldsymbol{x})}[\partial_{u_j} \log p(\boldsymbol{u} \mid \boldsymbol{x})] = \int_{\mathbb{R}^d} p(\boldsymbol{u} \mid \boldsymbol{x})\, \frac{\partial_{u_j} p(\boldsymbol{u} \mid \boldsymbol{x})}{p(\boldsymbol{u} \mid \boldsymbol{x})}\, d\boldsymbol{u} = \int_{\mathbb{R}^d} \partial_{u_j} p(\boldsymbol{u} \mid \boldsymbol{x})\, d\boldsymbol{u},$$

where the equality is valid wherever $p(\boldsymbol{u} \mid \boldsymbol{x}) > 0$, and condition 2 justifies the integral. Write $\boldsymbol{u} = (u_j, \boldsymbol{u}_{-j})$ and apply Fubini:

$$\int_{\mathbb{R}^d} \partial_{u_j} p(\boldsymbol{u} \mid \boldsymbol{x})\, d\boldsymbol{u} = \int_{\mathbb{R}^{d-1}} \left(\int_{\mathbb{R}} \partial_{u_j} p(u_j, \boldsymbol{u}_{-j} \mid x)\, du_j\right) d\boldsymbol{u}_{-j}.$$

For fixed $\boldsymbol{u}_{-j}$, the inner integral is a 1D integral of a derivative, hence by the fundamental theorem of calculus,

$$\int_{\mathbb{R}} \partial_{u_j} p(u_j, \boldsymbol{u}_{-j} \mid x)\, du_j = \Big[p(u_j, \boldsymbol{u}_{-j} \mid x)\Big]_{u_j=-\infty}^{u_j=+\infty}.$$

By condition 3, this boundary term is 0 for almost every $\boldsymbol{u}_{-j}$, so the outer integral is 0. Since this holds for each j, we conclude Eq. (42). □

We are now ready to prove the main result for the first-order score matching kernel.

Proposition 3.1 (Optimal value guidance)**.** *Consider a first-order solver with transitional kernel:*

$$p_{t_i}^{\Psi}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \mathcal{N}\Big(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\big(\boldsymbol{s}_{t_i}^{ref}(\boldsymbol{x}_{t_i}) + \boldsymbol{\Psi}_{t_i}\big),\ \sigma_{t_i}^2\mathbf{I}\Big).$$

Then, the guidance term $\boldsymbol{\Psi}_{t_i}$ that matches the optimal transitional kernel $p_{t_i}^{\star}(\cdot \mid \boldsymbol{x}_{t_i})$ is

$$\boldsymbol{\Psi}_{t_i}^{\star}(\boldsymbol{x}_{t_i}) \triangleq \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\, \mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p_{t_i}^{\star}(\cdot|\boldsymbol{x}_{t_i})}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\right], \tag{12}$$

which is called the optimal value guidance.

Proof. First, we will prove the first part, i.e. the current-state characterization:

$$\boldsymbol{\Psi}^{\star}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \tag{43}$$

From Eq. (10), we derive the optimal Value Guidance Vector. Take the logarithm of both sides:

$$\log p^{\star}_{t_i}(\boldsymbol{x}_{t_i}) = \log p^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{1}{\alpha} V_{t_i}(\boldsymbol{x}_{t_i}) - \log Z. \tag{44}$$

Then, differentiate with respect to $\boldsymbol{x}$ at $t = t_i$:

$$\begin{aligned} \nabla_{\boldsymbol{x}_{t_i}} \log p^{\star}_{t_i}(\boldsymbol{x}_{t_i}) &= \nabla_{\boldsymbol{x}_{t_i}} \log p^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) \\ \boldsymbol{s}^{\star}_{t_i}(\boldsymbol{x}_{t_i}) &= \boldsymbol{s}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \end{aligned} \tag{45}$$

Finally, by comparing this to the definition $\boldsymbol{s}^{\star}_{t_i}(\boldsymbol{x}_{t_i}) = \boldsymbol{s}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \boldsymbol{\Psi}^{\star}_{t_i}(\boldsymbol{x}_{t_i})$, we conclude that:

$$\boldsymbol{\Psi}^{\star}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \tag{46}$$

Next, we will express the optimal Value Guidance Vector in relation to $\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})$. Using Eq. (8), the *reference* one-step kernel at time t_i can be represented by

$$p^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \mathcal{N}\Big(\underbrace{\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})}_{=:\, \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})}\ ,\ \sigma^2_{t_i}\mathbf{I}\Big),$$

Using the definition of soft Bellman equations and soft-optimal policies, we obtain [41]:

$$p^{\star}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \frac{\exp\big(V^{\star}_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})/\alpha\big)\ p^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})}{\exp\big(V^{\star}_{t_i}(\boldsymbol{x}_{t_i})/\alpha\big)}. \tag{47}$$

Now plug in the Gaussian reference kernel. Up to an additive constant independent of $\boldsymbol{x}_{t_{i-1}}$,

$$\log p^{\star}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \frac{V^{\star}_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})}{\alpha} - \frac{1}{2\sigma^2_{t_i}}\big(\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})\big)^{\top}\big(\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})\big).$$

Differentiate w.r.t. $\boldsymbol{x}_{t_{i-1}}$:

$$\nabla_{\boldsymbol{x}_{t_{i-1}}} \log p^{\star}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_{i-1}}} V^{\star}_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}) - \frac{1}{\sigma^2_{t_i}}\big(\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})\big).$$

Taking conditional expectation under $p^{\star}_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$ and using Lemma B.1, we obtain

$$\mathbf{0} = \frac{1}{\alpha}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^{\star}_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})}\Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^{\star}_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\Big] - \frac{1}{\sigma^2_{t_i}}\Big(\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^{\star}_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})}[\boldsymbol{x}_{t_{i-1}}] - \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})\Big).$$

Rearranging yields the conditional mean identity

$$\mathbb{E}_{p^{\star}_{t_i}}[\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}] = \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{\sigma^2_{t_i}}{\alpha}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^{\star}_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})}\Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^{\star}_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\Big] \tag{48}$$

Matching the mean of $p^{\Psi}(\cdot \mid \boldsymbol{x}_{t_i})$, which is $\boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \Omega(t_i)\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i})$, to $\mathbb{E}_{p^{\star}_{t_i}}[\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}]$ gives

$$\boldsymbol{\Psi}^{\star}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^{\star}_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})}\Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^{\star}_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\Big].$$

□

B.3 Proof of Proposition 3.6

Proposition 3.6 (Conditional Consistency of First-Order Estimators)**.** *Let $j \in \{i, i-1\}$. Suppose that the soft value gradient at timestep t_j admits the exact decomposition*

$$\nabla_{\boldsymbol{x}_{t_j}} V_{t_j}(\boldsymbol{x}_{t_j}) = \gamma_{t_i} \nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j}). \tag{21}$$

Define the residual-corrected first-order estimator

$$\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) := \left(\frac{1}{\alpha}\iota[j=i] + \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\iota[j=i-1]\right)\left(\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j})\right), \tag{22}$$

where $\iota[\cdot]$ denotes the indicator function, and for $j = i-1$ we take $\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$, while for $j = i$ we simply set $\boldsymbol{x}_{t_j} = \boldsymbol{x}_{t_i}$. Then

$$\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) \;\middle|\; \boldsymbol{x}_{t_i}\right] = \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}), \tag{23}$$

where the expectation is trivial in the current-state case $j = i$. That is, under an exact residual parameterization g_ϕ, both the current-state first-order estimator and the one-step first-order estimator are unbiased for the optimal value guidance.

Proof. For $j = i$, Eq. (22) reduces to

$$\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(i) = \frac{1}{\alpha}\left(\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}) + g_\phi(\boldsymbol{x}_{t_i})\right) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) = \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}), \tag{49}$$

where the second equality follows from Eq. (21), and the third from Eq. (12).

For $j = i-1$, by linearity of expectation and Eq. (21),

$$\begin{aligned}\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(i-1) \;\middle|\; \boldsymbol{x}_{t_i}\right] &= \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot|\boldsymbol{x}_{t_i})}\left[\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}}) + g_\phi(\boldsymbol{x}_{t_{i-1}})\right] \\ &= \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot|\boldsymbol{x}_{t_i})}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\right].\end{aligned} \tag{50}$$

Applying Lemma 3.1 yields

$$\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(i-1) \;\middle|\; \boldsymbol{x}_{t_i}\right] = \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}), \tag{51}$$

which proves the claim. □

Proposition 3.6 formalizes the ideal case underlying residual-corrected first-order methods: if g_ϕ exactly captures the Jensen-gap residual induced by Tweedie's formula, then both current-state and one-step first-order guidance are unbiased. In practice, however, Appendix G.1 shows that the learned residual term is negligible, so this consistency condition is not realized empirically.

Thus, for existing methods, the Jensen gap

$$V_t(\boldsymbol{x}_t) - r(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)) = \alpha \log\left(\int \exp\left(\frac{r(\boldsymbol{x}_0)}{\alpha}\right) p^{\mathrm{ref}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) d\boldsymbol{x}_0\right) - \alpha \log\left(\exp\left(\frac{r(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t))}{\alpha}\right)\right) \tag{52}$$

is not meaningfully closed.

Under exact residual correction, Eq. (21) recovers the intended $\mathcal{L}_{\text{forward}}$ of Residual ∇-DB. Choosing $g_\phi(\cdot) = \mathbf{0}$ yields SQDF. $\gamma_t = 1, g_\phi(\cdot) = \mathbf{0}$ returns to the case where we approximate the value function with only Tweedie's formula, as in DPS [6].

B.4 Proof of Theorem 3.7

We prove that the REINFORCE-based estimator is an unbiased estimator of the gradient of the soft value function. This aligns with the practical reality that existing implementations of Soft PPO and GRPO [9, 24] treat the KL penalty as additive intermediate rewards rather than solving the exact log-sum-exp Soft Bellman equation.

Definition B.2 (Soft Value Functions of Diffusion MDP)**.** Let $r : \mathbb{R}^d \to \mathbb{R}$ be the terminal reward at $t = t_0$. Given a transition function $f : \mathbb{R}^d \times \mathbb{Z}_{\geq 0} \to \mathbb{R}^d$ and a noise schedule $\sigma : \mathbb{Z}_{\geq 0} \to \mathbb{R}$, the definition of the soft value function $V : \mathbb{R}^d \times \mathbb{Z}_{\geq 0} \to \mathbb{R}$ under the expectation over the optimal policy is stated as:

$$V(\boldsymbol{x}_{t_{i-1}}, t_{i-1}) := \mathbb{E}_{p^\star_{t_{1:i-1}}, \boldsymbol{\epsilon}_{t_{1:i-1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_{t_0}) - \alpha \sum_{k=1}^{i-1} \mathcal{D}_{\mathrm{KL}}(p^\star_{t_k} \| p^{\mathrm{ref}}_{t_k})] \tag{53}$$

$$:= \max_{p_{t_{1:i-1}}} \left[\mathbb{E}_{p_{t_{1:i-1}}, \boldsymbol{\epsilon}_{t_{1:i-1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_{t_0}) - \alpha \sum_{k=1}^{i-1} \mathcal{D}_{\mathrm{KL}}(p_{t_k} \| p^{\mathrm{ref}}_{t_k})] \right] \tag{54}$$

Theorem 3.7 (Consistency of Policy Gradient)**.** *For the full-rollout $j = 0$, the lookahead zeroth-order estimator is an unbiased estimator of the soft value gradient:*

$$\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i} \approx \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \mathbb{E}_{\boldsymbol{\epsilon}_{t_i:t_0} \overset{\mathrm{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}] = \mathbb{E}_{\boldsymbol{\epsilon}_{t_i}}[\frac{\sigma^2_{t_i}}{\alpha \Omega(t_i)} \nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})] = \boldsymbol{\Psi}^\star_{t_i}. \tag{24}$$

Proof. Since the soft value function is defined as a maximum of the cost functional with optimal policy, Envelope Theorem states that the gradient of the soft value function $\nabla_{\boldsymbol{x}_{t_{i-1}}} V(\boldsymbol{x}_{t_{i-1}}, t_{i-1})$ is

$$\mathbb{E}_{p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V(\boldsymbol{x}_{t_{i-1}}, t_{i-1}) \right] \tag{55}$$

$$= \mathbb{E}_{p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{\partial V(\boldsymbol{x}_{t_{i-1}}, t_{i-1})}{\partial \boldsymbol{x}_{t_{i-1}}} \Big|_{p_{t_{1:i-1}} = p^\star_{t_{1:i-1}}} \right] \tag{56}$$

$$= \mathbb{E}_{p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{x}_{t_{i-1}}} \left(\mathbb{E}_{p^\star_{t_{1:i-1}}, \boldsymbol{\epsilon}_{t_{1:i-1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_{t_0})] \right) \right] \tag{57}$$

$$= \mathbb{E}_{p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{1}{\sigma_t} \nabla_{\boldsymbol{\epsilon}_{t_i}} \left(\mathbb{E}_{p^\star_{t_{1:i-1}}, \boldsymbol{\epsilon}_{t_{1:i-1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_{t_0})] \right) \right] \tag{58}$$

$$= \mathbb{E}_{p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{1}{\sigma_t} \left(\mathbb{E}_{p^\star_{t_{1:i-1}}, \boldsymbol{\epsilon}_{t_{1:i-1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_{t_0}) \boldsymbol{\epsilon}_{t_i}] \right) \right] \tag{59}$$

where the last line came from a special case of Stein's identity, which states that for a differentiable function h,

$$\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[h(\boldsymbol{\epsilon}) \boldsymbol{\epsilon}] = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})}[\nabla_{\boldsymbol{\epsilon}} h(\boldsymbol{\epsilon})]. \tag{60}$$

Therefore, the optimal value guidance term $\boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i})$ by Lemma 3.1 is identical to the value guidance derived by REINFORCE $\boldsymbol{\Psi}_{t_i}$ derived in Appendix C.2.1. This confirms that REINFORCE and other zeroth-order methods utilize an unbiased estimator of the soft value gradient.

$$\mathbb{E}_{\boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}}[\frac{\sigma^2_{t_i}}{\alpha \Omega(t_i)} \nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})] = \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \mathbb{E}_{\boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i:t_0} \overset{\mathrm{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})}[r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}] \tag{61}$$

□

B.5 Convergence of Policy Iteration

Section 3 and Appendix B characterize the reward-guided target and its associated optimal guidance as fixed-point identities of the optimum $p^\star$. In practice, however, RL-based fine-tuning methods sample from the current policy p^θ and update that policy using the RSM loss. We therefore rely on prior policy optimization theory for convergence guarantees. TRPO [36] was originally derived from a monotonic improvement perspective, and subsequent theory established that exact TRPO converges to the optimal policy in entropy-regularized MDPs [29]. More recent results established global convergence guarantees for PPO-clip and its variants under neural function approximation [17]. Accordingly, our core theory specifies the fixed point that the reward score matching update is designed to target, while convergence of the resulting iterative optimization procedure is justified by prior works rather than proven in this paper.

C Proof of Theorem 3.2

Theorem 3.2 (Reward Score Matching)**.** *For Soft RL methods and VGG-Flow, the loss $\mathcal{L}(\theta)$ admits the common form*

$$\mathbb{E}_{t_i,\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}}\left[C_1(t_i)\Big(\|\boldsymbol{s}^{\theta}_{t_i}-(\boldsymbol{s}^{ref}_{t_i}+\boldsymbol{\Psi}_{t_i})\|^2+C_2(t_i)\|\boldsymbol{s}^{\theta}_{t_i}-\boldsymbol{s}^{\theta^\dagger}_{t_i}\|^2\Big)\right] \tag{13}$$

where C_1, C_2 are weights, and $\boldsymbol{s}^{\theta}$, $\boldsymbol{s}^{\theta^\dagger}$, $\boldsymbol{s}^{ref}$ are the current, previous, and reference score functions [14].

Proof. The proof for each method is laid throughout Appendix C. □

Technical remark ($\alpha = 0$). SQDF and policy-gradient methods are well-defined even when $\alpha = 0$. $\boldsymbol{\Psi}_{t_i}$ and $C_2(t_i)$ appear singular due to $1/\alpha$, but these quantities appear only through products $C_1(t_i)\boldsymbol{\Psi}_{t_i}$ and $C_1(t_i)C_2(t_i)$ in Eq. (15). The factor of α in $C_1(t_i)$ cancels the apparent singularity. Therefore, when $\alpha = 0$, the canonical gradient remains the correct object for interpreting optimization dynamics, although the literal decomposition from Eq. (11) no longer strictly applies.

C.1 First-order Methods

C.1.1 Residual ∇-DB

The relationship between Residual ∇-DB and Soft RL has bene briefly discussed in Appendix D of Liu et al. [27]. We rewrite the objective by mapping forward and backward GFlowNet processes (P_F, P_B) to reverse sampling and forward noising, respectively.

Residual ∇-DB minimizes a weighted sum of four terms:

$$\mathcal{L}_{\text{total}}(\theta,\phi) = w_F\mathcal{L}_{\text{forward}}(\theta,\phi) + w_B\mathcal{L}_{\text{backward}}(\theta,\phi) + \mathcal{L}_{\text{terminal}}(\phi) + w_R\mathcal{L}_{\text{reg}}(\theta), \tag{62}$$

$$\mathcal{L}_{\text{forward}}(\theta,\phi) = \mathbb{E}_{t_i,\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}_{t_i}}\left[\left\|\nabla_{\boldsymbol{x}_{t_{i-1}}}\ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}},\boldsymbol{x}_{t_i}) - \mathcal{G}_{\text{GFN\_F}}(\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}_{t_i})\right\|^2\right], \tag{63}$$

$$\mathcal{L}_{\text{backward}}(\theta,\phi) = \mathbb{E}_{t_i,\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}_{t_i}}\left[\left\|\nabla_{\boldsymbol{x}_{t_i}}\ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}},\boldsymbol{x}_{t_i}) + \mathcal{G}_{\text{GFN\_B}}(\boldsymbol{x}_{t_i})\right\|^2\right], \tag{64}$$

$$\mathcal{L}_{\text{terminal}}(\phi) = \mathbb{E}_{\boldsymbol{x}_0}[\|g_\phi(\boldsymbol{x}_0)\|^2], \tag{65}$$

$$\mathcal{L}_{\text{reg}}(\theta) = \mathbb{E}_{t_i,\boldsymbol{x}_{t_i}}[\|\boldsymbol{\epsilon}^{\theta}_{t_i}-\boldsymbol{\epsilon}^{\theta^\dagger}_{t_i}\|^2] = \mathbb{E}_{t_i,\boldsymbol{x}_{t_i}}[(1-\bar{\alpha}_{t_i})\|\boldsymbol{s}^{\theta}_{t_i}-\boldsymbol{s}^{\theta^\dagger}_{t_i}\|^2], \tag{66}$$

where ℓ, gradient fields $\mathcal{G}$ and Gaussian reparameterization are:

$$\ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}},\boldsymbol{x}_{t_i}) := \log p^{\theta}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) - \log p^{\text{ref}}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}), \tag{67}$$

$$\mathcal{G}_{\text{GFN\_F}}(\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}_{t_i}) = \frac{1}{\alpha}\tilde{\gamma}_{t_i}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}}r(\hat{\boldsymbol{x}}_{0|t_{i-1}})) + g_\phi(\boldsymbol{x}_{t_{i-1}}), \tag{68}$$

$$\mathcal{G}_{\text{GFN\_B}}(\boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\tilde{\gamma}_{t_i}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_i}}r(\hat{\boldsymbol{x}}_{0|t_i})) + g_\phi(\boldsymbol{x}_{t_i}), \tag{69}$$

$$\boldsymbol{x}_{t_{i-1}} = \boldsymbol{\mu}^{\theta}_{t_i} + \sigma_{t_i}\boldsymbol{\epsilon}_{t_i}, \quad \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0},\mathbf{I}).$$

Pruning $\mathcal{L}_{\text{backward}}$ and $\mathcal{L}_{\text{terminal}}$. $\mathcal{L}_{\text{backward}}$ requires the gradient $\nabla_{\boldsymbol{x}_{t_i}}\ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}},\boldsymbol{x}_{t_i})$. Under the Gaussian reparameterization, $\boldsymbol{x}_{t_{i-1}}$ depends on $\boldsymbol{x}_{t_i}$ through the model output, which introduces Jacobian terms of the form $\partial\boldsymbol{\epsilon}^{\theta}_{t_i}/\partial\boldsymbol{x}_{t_i}$ into $\nabla_{\boldsymbol{x}_{t_i}}\ell^{\theta}_{t_i}$. Consequently, $\nabla_\theta\mathcal{L}_{\text{backward}}$ involves higher-order derivatives, including mixed θ-$\boldsymbol{x}$ terms, rendering optimization numerically unstable and empirically ineffective. This is consistent with our ablation results: removing $\mathcal{L}_{\text{backward}}$ does not degrade performance, whereas removing $\mathcal{L}_{\text{forward}}$ causes training to fail (Appendix G.1).

$\mathcal{L}_{\text{terminal}}$ (Eq. (65)) is not parameterized by θ and therefore contributes no gradient to the canonical loss term in Theorem 3.2. Moreover, as discussed in Appendix G.1, the outputs of g_ϕ are negligible in magnitude and do not affect the training dynamics.

[14] When $C_2(t) = 0$, Eq. (13) reduces to entropy-regularized optimal control (Appendix H.1).

Reduction to Reward Score Matching. $\nabla_{\boldsymbol{x}_{t_{i-1}}} \ell^\theta_{t_i}$ can be expressed as a scaled score difference:

$$\nabla_{\boldsymbol{x}_{t_{i-1}}} \ell^\theta_{t_i}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{x}_{t_i}) = \frac{1}{\sigma^2_{t_i}}(\boldsymbol{\mu}^\theta_{t_i} - \boldsymbol{\mu}^{\text{ref}}_{t_i}) = \frac{\Omega(t_i)}{\sigma^2_{t_i}}(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}). \tag{70}$$

$$\therefore \mathcal{L}_{\text{forward}}(\theta, \phi) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[\frac{\Omega(t_i)^2}{\sigma^4_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \left(\boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\sigma^2_{t_i}}{\Omega(t_i)} \mathcal{G}_{\text{GFN\_F}}(\boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}) \right) \right\|^2 \right]. \tag{71}$$

Thus, $w_F \mathcal{L}_{\text{forward}} + w_R \mathcal{L}_{\text{reg}}$ is a special case of the canonical loss, with

$$\boldsymbol{\Psi}_{t_i} = \tilde{\gamma}_{t_i} \frac{\sigma^2_{t_i}}{\alpha \Omega(t_i)} \operatorname{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})), \quad C_1(t_i) = \frac{1}{d} \frac{\Omega(t_i)^2}{\sigma^4_{t_i}}, \quad C_2(t_i) = \frac{(1 - \bar{\alpha}_{t_i})\sigma^4_{t_i}}{\Omega(t_i)^2} \frac{w_R}{w_F}, \tag{72}$$

where the factor $\frac{1}{d}$ arises implicitly from a `torch.mean()` operation over tensor dimensions.

C.1.2 SQDF

SQDF is motivated by the following objective, given in Eq. 11 of Kang et al. [18]:

$$\mathcal{L}(\theta) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \big[-r(\hat{\boldsymbol{x}}_{0|t_{i-1}}) + \alpha \mathcal{D}_{\text{KL}}(p^\theta(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) \| p^{\text{ref}}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})) \big]. \tag{73}$$

In the actual implementation, however, the transition mean $\boldsymbol{\mu}^\theta_{t_{i-1}}$ is reparameterized, and a stopgrad operation is applied to the Tweedie estimate inside the reward term. In addition, an attenuation factor $\tilde{\gamma}_{t_i} = \tilde{\gamma}^{Nt_i}_{\text{SQDF}}$ is introduced in the reward term for temporal credit assignment. To analyze the resulting optimization dynamics, we adopt the parameterization in Lemma 3.1.

$$\mathcal{L}(\theta) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[-\tilde{\gamma}_{t_i} \ \operatorname{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})) \cdot \boldsymbol{\mu}^\theta_{t_i} + \frac{\alpha}{2\sigma^2_{t_i}} \left\| \boldsymbol{\mu}^\theta_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}} \right\|^2 \right]. \tag{74}$$

Substituting $\boldsymbol{\mu}^\theta_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}} = \Omega(t_i)(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i})$ yields

$$\mathbb{E}\left[-\tilde{\gamma}_{t_i} \ \operatorname{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})) \cdot \left(\kappa(t_i) \boldsymbol{x}_{t_i} + \Omega(t_i) \boldsymbol{s}^\theta_{t_i} \right) + \frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} \right\|^2 \right], \tag{75}$$

Up to additive terms that do not contribute to the gradient, this is equivalent to:

$$\mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[\frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \left(\boldsymbol{s}^{\text{ref}}_{t_i} + \tilde{\gamma}_{t_i} \frac{\sigma^2_{t_i}}{\alpha \Omega(t_i)} \ \operatorname{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})) \right) \right\|^2 \right]. \tag{76}$$

Therefore, SQDF is a special case of the canonical loss, with

$$\boldsymbol{\Psi}_{t_i} = \tilde{\gamma}_{t_i} \frac{\sigma^2_{t_i}}{\alpha \Omega(t_i)} \ \operatorname{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})), \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma^2_{t_i}}, \quad C_2(t_i) = 0. \tag{77}$$

C.1.3 VGG-Flow

To facilitate comparison, we rewrite the VGG-Flow objectives using the parameterization $G^\phi(\boldsymbol{x}_{t_i}) \triangleq -\tilde{\gamma}_{t_i} \operatorname{sg}\big(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i})\big) - g^\phi(\boldsymbol{x}_{t_i})$.

$$\mathcal{L}_{\text{consistency}}(\phi) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}} \left\| \frac{\partial}{\partial t} G^\phi(\boldsymbol{x}_t) \Big|_{t=t_i} + [\nabla G^\phi(\boldsymbol{x}_{t_i})]^\top (\boldsymbol{v}^{\text{ref}}_{t_i} + \frac{1}{\lambda} G^\phi(\boldsymbol{x}_{t_i})) + [\nabla_{\boldsymbol{x}_{t_i}} \boldsymbol{v}^{\text{ref}}_{t_i}]^\top G^\phi(\boldsymbol{x}_{t_i}) \right\|^2, \tag{78}$$

$$\mathcal{L}_{\text{boundary}}(\phi) = \mathbb{E}_{\boldsymbol{x}_0} \| g^\phi(\boldsymbol{x}_0) \|^2, \tag{79}$$

$$\mathcal{L}_{\text{matching}}(\theta) = \mathbb{E}_{\boldsymbol{x}_{t_i}} \left[\| \boldsymbol{v}^\theta_{t_i} - \boldsymbol{v}^{\text{ref}}_{t_i} - \frac{1}{\alpha} [-\tilde{\gamma}_{t_i} \operatorname{sg}((\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i})) - g^\phi(\boldsymbol{x}_{t_i})] \|^2 \right]. \tag{80}$$

Here, $\mathcal{L}_{\text{total}}(\theta, \phi) = \mathcal{L}_{\text{matching}}(\theta) + \mathcal{L}_{\text{consistency}}(\phi) + w_b \mathcal{L}_{\text{boundary}}(\phi)$.[15]

[15] The original paper and our manuscript use opposite time conventions ($0 \to 1$ vs. $1 \to 0$). To match our notation, we flip the signs highlighted in red relative to the original formulation.

As with Residual ∇-DB, and as discussed in the main text, $\mathcal{L}_{\text{boundary}}$ effectively enforces $g^\phi(\boldsymbol{x}_{t_i}) \approx \mathbf{0}$ for all t_i. After dropping g^ϕ, we analyze the dominant term, $\mathcal{L}_{\text{matching}}(\theta)$, by mapping the velocity field $\boldsymbol{v}_{t_i}$ to our unified score parameterization $\boldsymbol{s}_{t_i}$ under the general SDE formulation, specifically using the $\boldsymbol{s}_{t_i}$ form without $\Omega(t_i)$.

Using $\delta(t_i)$ for $\boldsymbol{v}$-parameterized Rectified Flow (Eq. (121)), $\mathcal{L}_{\text{matching}}$ can be written as:

$$\begin{aligned}\mathcal{L}_{\text{matching}}(\theta) &= \mathbb{E}_{\boldsymbol{x}_{t_i}}\left[\left\| \frac{1}{\delta(t_i)}(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}) - \frac{1}{\alpha}\tilde{\gamma}_{t_i}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i}))\right\|^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}_{t_i}}\left[\frac{1}{\delta(t_i)^2}\left\|\boldsymbol{s}^\theta_{t_i} - \left(\boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\delta(t_i)}{\alpha}\gamma_{t_i}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i}))\right)\right\|^2\right].\end{aligned} \tag{81}$$

Therefore, VGG-Flow is a special case of the canonical loss, with

$$\boldsymbol{\Psi}_{t_i} = \frac{\tilde{\gamma}_{t_i}}{\alpha}\delta(t_i)\ \text{sg}((\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i})), \quad C_1(t_i) = \frac{1}{d}\frac{1}{\delta(t_i)^2}, \quad C_2(t_i) = 0, \tag{82}$$

where the factor $\frac{1}{d}$ arises implicitly from a `torch.mean()` operation over tensor dimensions, rather than from an explicit scalar normalization.

C.2 Zeroth-order Methods

We now turn to policy-gradient-based methods. As the canonical zeroth-order base case, we begin with a KL-regularized REINFORCE objective, which is the natural policy-gradient analogue of the Soft RL formulation under our unified transition parameterization. Note that this is a *strictly* on-policy algorithm. This yields the fundamental zeroth-order effective value guidance, after which all variants can be understood as straightforward modifications within the design space of RSM.

C.2.1 KL-regularized REINFORCE

We formulate REINFORCE [39, 46] in the KL-regularized form used by Algorithm 1 (Soft PPO) of Uehara et al. [41]. Using our general parameterization, the objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^\theta, \boldsymbol{\epsilon}_{t_{i:1}}}\left[-r(\boldsymbol{x}_0) + \alpha\sum_{i=1}^{N}\mathcal{D}_{\text{KL}}(p^\theta(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i})\|p^{\text{ref}}(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}))\right]. \tag{83}$$

For Gaussian transitions with fixed covariance $\sigma^2_{t_i}\mathbf{I}$, each KL term reduces to

$$\frac{\alpha}{2}\|\frac{1}{\sigma_{t_i}}(\boldsymbol{\mu}^\theta_{t_{i-1}}(\boldsymbol{x}_{t_i}) - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}}(\boldsymbol{x}_{t_i}))\|^2. \tag{84}$$

The REINFORCE gradient of the reward term is

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^\theta, \boldsymbol{\epsilon}_{t_{i:1}}}[-r(\boldsymbol{x}_0)] = \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^\theta, \boldsymbol{\epsilon}_{t_{i:1}}}\left[-r(\boldsymbol{x}_0)\nabla_\theta \log p_\theta(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i})\right]. \tag{85}$$

Since

$$\nabla_\theta \log p_\theta(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) = (\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^\theta_{t_{i-1}})^\top \frac{1}{\sigma^2_{t_i}}\nabla_\theta \boldsymbol{\mu}^\theta_{t_{i-1}} = (\frac{1}{\sigma_{t_i}}\boldsymbol{\epsilon}_{t_i})^\top \nabla_\theta \boldsymbol{\mu}^\theta_{t_{i-1}}, \tag{86}$$

combining the reward and KL terms yields

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_\tau\left[\left(-r(\boldsymbol{x}_0)\frac{1}{\sigma_{t_i}}\boldsymbol{\epsilon}_{t_i} + \alpha\frac{1}{\sigma^2_{t_i}}(\boldsymbol{\mu}^\theta_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}})\right)^\top \nabla_\theta \boldsymbol{\mu}^\theta_{t_{i-1}}\right], \tag{87}$$

where $\tau = \{\boldsymbol{x}_{t_i}\}_{i=0}^N$ denotes the trajectory induced by θ.

Substituting the unified parameterization $\boldsymbol{\mu}^\theta_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^\theta_{t_i}$ from Lemma 3.1, we obtain

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_\tau\left[\left(-r(\boldsymbol{x}_0)\frac{1}{\sigma_{t_i}}\boldsymbol{\epsilon}_{t_i} + \frac{\alpha\Omega(t_i)}{\sigma^2_{t_i}}(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i})\right)^\top \Omega(t_i)\nabla_\theta \boldsymbol{s}^\theta_{t_i}\right] \tag{88}$$

$$= \mathbb{E}_\tau\left[\frac{\alpha\Omega(t_i)^2}{\sigma^2_{t_i}}\left(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} - \frac{\sigma_{t_i}}{\alpha\Omega(t_i)}r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}\right)^\top \nabla_\theta \boldsymbol{s}^\theta_{t_i}\right]. \tag{89}$$

This is precisely the gradient of the weighted squared loss

$$\mathcal{L}(\theta) = \mathbb{E}_\tau \left[\frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \| (\boldsymbol{s}_{t_i}^\theta - (\boldsymbol{s}_{t_i}^{\text{ref}} + \boldsymbol{\Psi}_{t_i})) \|^2 \right], \tag{90}$$

with

$$\boldsymbol{\Psi}_{t_i} = \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = 0. \tag{91}$$

In relation to Lemma 3.1, REINFORCE corresponds to the zeroth-order estimation of the lookahead value gradient,

$$\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{\boldsymbol{x}_{t_i}}} \left[\nabla_{\boldsymbol{x}_{t_{i-1}}} \boldsymbol{V}^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}) \right] = \frac{\mathbb{E}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}]}{\sigma_{t_i}}, \tag{92}$$

using a point estimation for the expectation.

C.2.2 Clipped log-ratio surrogate

We next consider the clipped log-ratio surrogate used by PCPO-base [20]. Denoting the old-policy ratio by

$$\rho_{t_i} = \frac{p_\theta(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})}{p_{\theta^\dagger}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})}, \tag{93}$$

the clipped objective is

$$\mathcal{L}_{\text{clip-log}}(\theta) = \begin{cases} \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^{\theta^\dagger}, \boldsymbol{\epsilon}_{t_i}} \left[-r(\boldsymbol{x}_0) \log \rho_{t_i} + \frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\text{ref}} \right\|^2 \right] & \text{if } \big(-\xi \le \log \rho_{t_i} \le \xi \big) \\ & \quad \text{or } \big(\log \rho_{t_i} < -\xi \wedge r(\boldsymbol{x}_0) \ge 0 \big) \\ & \quad \text{or } \big(\log \rho_{t_i} > \xi \wedge r(\boldsymbol{x}_0) \le 0 \big), \\ \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^{\theta^\dagger}, \boldsymbol{\epsilon}_{t_i}} \left[\frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\text{ref}} \right\|^2 \right] & \text{otherwise.} \end{cases} \tag{94}$$

where

$$-\log \rho_{t_i} = -\frac{\Omega(t_i)}{\sigma_{t_i}} (\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger})^\top \boldsymbol{\epsilon}_{t_i} + \frac{\Omega(t_i)^2}{2\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger} \right\|^2. \tag{95}$$

We group terms by $\boldsymbol{s}_{t_i}^\theta$, and perform a square completion equivalent to the previous section:

$$\begin{aligned} & r(\boldsymbol{x}_0)(-\log \rho_{t_i}) + \frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\text{ref}} \right\|^2 \\ &= -\frac{r(\boldsymbol{x}_0)\Omega(t_i)}{\sigma_{t_i}} (\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger} - \boldsymbol{s}_{t_i}^{\text{ref}} + \boldsymbol{s}_{t_i}^{\text{ref}})^\top \boldsymbol{\epsilon}_{t_i} + \frac{r(\boldsymbol{x}_0)}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2} \| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger} \|^2 + \frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\text{ref}} \|^2 \\ &= \frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \| \boldsymbol{s}_{t_i}^\theta - \big(\boldsymbol{s}_{t_i}^{\text{ref}} + \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i} \big) \|^2 + \frac{r(\boldsymbol{x}_0)}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2} \| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger} \|^2 + C \end{aligned} \tag{96}$$

for a constant C independent of θ. Thus,

$$\mathcal{L}(\theta) = \begin{cases} \mathbb{E} \left[\frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \big(\boldsymbol{s}_{t_i}^{\text{ref}} + \boldsymbol{\Psi}_{t_i} \big) \right\|^2 + \frac{r(\boldsymbol{x}_0)}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger} \right\|^2 \right] & \text{if } \big(-\xi \le \log \rho_{t_i} \le \xi \big) \\ & \quad \text{or } \big(\log \rho_{t_i} < -\xi \wedge r(\boldsymbol{x}_0) \ge 0 \big) \\ & \quad \text{or } \big(\log \rho_{t_i} > \xi \wedge r(\boldsymbol{x}_0) \le 0 \big), \\ \mathbb{E} \left[\frac{\alpha \Omega(t_i)^2}{2\sigma_{t_i}^2} \left\| \boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\text{ref}} \right\|^2 \right] & \text{otherwise.} \end{cases} \tag{97}$$

Therefore, the clipped log-ratio surrogate admits the canonical form with

$$\boldsymbol{\Psi}_{t_i} = \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{98}$$

C.2.3 Variants

EPG. EPG [4] is a variant of REINFORCE, which extends it to the *loose* on-policy setting, and uses a control variate for variance reduction. Its loss can be written as

$$\mathcal{L}_{\mathrm{EPG}}(\theta) = \mathbb{E}_{\tau\sim p^{\theta^\dagger}}\Big[\,\mathrm{sg}(\rho_{t_i})\,\frac{\alpha\Omega(t_i)^2}{2\sigma(t_i)^2}\big\|\boldsymbol{s}^\theta_{t_i} - \big(\boldsymbol{s}^{\mathrm{ref}}_{t_i} + \boldsymbol{\Psi}_{t_i}\big)\big\|^2\Big], \tag{99}$$

with the same $\boldsymbol{\Psi}_{t_i}$ as KL-regularized REINFORCE. Compared to REINFORCE, the only modification is the multiplicative stop-gradient importance ratio, which does not produce gradients itself.

PPO, GRPO. Huang et al. [17] proved that the KL-regularized PPO loss

$$\mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^{\theta^\dagger},\boldsymbol{\epsilon}_{t_i}}[\max\big(-\rho_{t_i}r(\boldsymbol{x}_0), -\mathrm{clip}_\xi(\rho_{t_i})r(\boldsymbol{x}_0)\big) + \frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\big\|\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\mathrm{ref}}_{t_i}\big\|^2] \tag{100}$$

is equivalent to

$$\mathcal{L}_{\mathrm{clip}}(\theta) = \begin{cases} \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^{\theta^\dagger},\boldsymbol{\epsilon}_{t_i}}\left[-r(\boldsymbol{x}_0)(\rho_{t_i}-1) + \frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\big\|\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\mathrm{ref}}_{t_i}\big\|^2\right] & \text{if } \big(1-\xi\le\rho_{t_i}\le 1+\xi\big) \\ & \quad \text{or } \big(\rho_{t_i} < 1-\xi \wedge r(\boldsymbol{x}_0)\ge 0\big) \\ & \quad \text{or } \big(\rho_{t_i} > 1+\xi \wedge r(\boldsymbol{x}_0)\le 0\big), \\ \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^{\theta^\dagger},\boldsymbol{\epsilon}_{t_i}}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\big\|\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\mathrm{ref}}_{t_i}\big\|^2\right] & \text{otherwise.} \end{cases} \tag{101}$$

up to constants independent of θ. Therefore, PPO simply replaces the log-ratio term in Eq. (94) with $\rho_{t_i} - 1$. For Gaussian transitions,

$$\rho_{t_i} - 1 = \exp\left(-\frac{\Omega(t_i)}{\sigma_{t_i}}(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i})^\top\boldsymbol{\epsilon}_{t_i} - \frac{\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i}\right\|^2\right) - 1. \tag{102}$$

Directly optimizing this exponential is numerically unstable. In practice, diffusion/flow-based PPO and GRPO implementations [3, 9, 24, 50] use an extremely tight clipping threshold (e.g., $\xi = 10^{-4}$). Under such tight clipping, the first-order approximation $\exp(x) - 1 \approx x$ incurs error of order $\xi^2/2$ for the interior regime $1-\xi \le \rho_{t_i} \le 1+\xi$, as well as for clipped cases where the loss is already fixed at the clipping boundary[16]. Under this necessary approximation, PPO / GRPO reduce to the clipped log-ratio surrogate in Eq. (97), and therefore have the same unified terms as Eq. (98).

PCPO-reweight. For flow-matching models, PCPO *explicitly* reweights the policy loss by

$$\gamma_{t_i} = \frac{w_{\mathrm{new}}(t_i)}{w(t_i)}, \qquad w_{\mathrm{new}}(t_i) = \zeta\Delta t_i, \qquad \zeta = \sum_{i=1}^{N} w(t_i). \tag{103}$$

See Appendix D.3 for more details. This modifies only the effective guidance and anchor strength:

$$\boldsymbol{\Psi}_{t_i} = \gamma_{t_i}\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \gamma_{t_i}\frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{104}$$

For diffusion models, PCPO *implicitly* reweights $h(t_i)$ by modifying the variance schedule σ_{t_i} to σ'_{t_i}. Its effect is summarized in Table 1.

GRPO-Guard. GRPO-Guard observes that $\log\rho_{t_i}$ is negatively biased by the KL term between the current and old policies, which empirically shifts its mean below zero. To correct this, it centers and rescales the log-ratio as

$$\sigma_{t_i}\big(\log\rho_{t_i} - \mathbb{E}[\log\rho_{t_i}]\big),$$

and additionally reweights the policy term by $1/\Delta t_i$. Under the clipped log-ratio objective, this yields

$$\mathcal{L}(\theta) = \begin{cases} \mathbb{E}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\big\|\boldsymbol{s}^\theta_{t_i} - \big(\boldsymbol{s}^{\mathrm{ref}}_{t_i} + \boldsymbol{\Psi}_{t_i}\big)\big\|^2\right] & \text{if } -\xi \le \sigma_{t_i}\big(\log\rho_{t_i} - \mathbb{E}[\log\rho_{t_i}]\big) \le \xi, \\ \mathbb{E}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\big\|\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\mathrm{ref}}_{t_i}\big\|^2\right] & \text{otherwise,} \end{cases} \tag{105}$$

[16]The remaining non-clipped sign-consistent regimes, $(\rho_{t_i} < 1-\xi\wedge r(\boldsymbol{x}_0)\ge 0)$ and $(\rho_{t_i} > 1+\xi\wedge r(\boldsymbol{x}_0)\le 0)$, admit a looser bound. Empirically, however, Lee and Ye [20] observe that $|\rho_{t_i} - 1| < 0.01$ throughout training due to a low learning rate, so the approximation error remains small in practice.

Table 3: **Unified Notation for Prior Works.**

Parameter Description	Notation in Literature	Unified
Entropy Regularization	α [41] $\frac{2\alpha}{d}$ [18] β [20, 24] $\frac{1}{\beta}$ [26, 27]	α
Downweighting Factor	η_t [26] γ_t [18, 27]	$\tilde{\gamma}_t$
Residual Parameterization	g_ϕ [27] $-\nu_\phi$ [26]	g_ϕ
Reward Representation	$\log R(\cdot)$ [27, 51] $r(\cdot)$ [18, 20, 24, 26, 41]	$r(\cdot)$

with

$$\boldsymbol{\Psi}_{t_i} = \gamma_{t_i} \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = 0, \tag{106}$$

where $\gamma_{t_i} = \frac{\sigma_{t_i} \Omega(t_i)}{\Delta t_i}$. Thus, GRPO-Guard explicitly removes the old-policy anchor term.

TempFlow-GRPO. TempFlow-GRPO introduces *trajectory branching*: starting from a deterministic ODE trajectory, it revisits each intermediate latent $\boldsymbol{x}_{t_i}$, injects one-step SDE noise to generate multiple descendants, and then deterministically denoises each descendant to the terminal state for reward evaluation[17]. This is precisely a multi-particle lookahead estimator in our framework. Note that TempFlow-GRPO does not explicitly average value-guidance vectors; instead, it applies an L_2 regression loss to multiple reward-weighted point estimates. However, because the gradient of a sum of squared errors is centered at the sample mean, this optimization is functionally equivalent to regressing toward the average of those particle-wise targets. Hence, under RSM, TempFlow-GRPO implicitly realizes a zeroth-order lookahead value-guidance estimator with branching width $K_i > 1$, together with heuristic temporal reweighting $\gamma(t_i) = \frac{9}{4}\sigma_{t_i}$. This yields

$$\boldsymbol{\Psi}_{t_i} = \gamma_{t_i} \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \Big(\frac{1}{K_i} \sum_{k=1}^{K_i} r(\boldsymbol{x}_0^{(k)}) \boldsymbol{\epsilon}_{t_i}^{(k)} \Big), \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \gamma_{t_i} \frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{107}$$

BranchGRPO. BranchGRPO likewise uses multiple descendants to estimate a zeroth-order lookahead guidance term, but differs from TempFlow-GRPO in that its branching is *recursive* rather than one-step from each revisited parent latent. Under the same RSM view, it therefore realizes a multi-particle estimator form:

$$\boldsymbol{\Psi}_{t_i} = \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \Big(\frac{1}{K_i} \sum_{k=1}^{K_i} r(\boldsymbol{x}_0^{(k)}) \boldsymbol{\epsilon}_{t_i}^{(k)} \Big), \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{108}$$

C.3 Remark on parameters.

To address the discrepancies in naming conventions across Soft RL and GFlowNet literature, we have unified the notation for components that serve identical functions. Table 3 maps the various notations found in prior works to the single, unified notation used in this text.

Entropy Regularization (α): While often denoted as β (inverse temperature) or $1/\beta$ in Soft RL contexts, we adopt α to strictly control the degree of entropy regularization[18] .

Downweighting Factor ($\tilde{\gamma}_t$): We rename VGG-Flow's one-step reward estimation factor from η_t to $\tilde{\gamma}_t$ to align with the conventions established by Residual ∇-DB and SQDF.

Reward Value ($r(\cdot)$): In diffusion fine-tuning GFlowNet literature, the reward is often interpreted as the log of the GFlowNet reward. We use $r(\cdot)$ directly to refer to this value.

[17] Although TempFlow-GRPO presents the sampling order differently, it is functionally equivalent to this interpretation.

[18] SQDF [18] apply mean reduction over the latent dimension d for *only* entropy regularization, and omit the $\frac{1}{2}$ coefficient for the KL divergence. Consequently, their reported α corresponds to an effective regularization weight of $2\alpha/d$.

D Parameterizations

D.1 Ω Derivations for Popular Samplers

In this section, we provide step-by-step derivations of the scaling factor $\Omega(t_i)$ for two widely used sampling algorithms: DDIM applied to VP-SDE (e.g., Stable Diffusion 1.5) and Euler Discrete applied to Rectified Flow (e.g., Stable Diffusion 3).

From Proposition 3.1, the relationship between the optimal score $\boldsymbol{s}^\star_{t_i}$, the reference score $\boldsymbol{s}^{\text{ref}}_{t_i}$, and the value gradient is established via the conditional transition mean $\boldsymbol{\mu}_{t_{i-1}}$. When $\boldsymbol{\mu}_{t_{i-1}}$ can be expressed in the form $\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}_{t_i}$, the guidance term simplifies to:

$$\boldsymbol{s}^\star_{t_i} = \boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}). \tag{109}$$

The specific value of $\Omega(t_i)$ depends on the choice of noise schedule and numerical solver.

D.1.1 DDIM Sampler (VP-SDE)

For the DDIM sampler applied to a VP-SDE, the transition from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_{i-1}}$ is defined as:

$$\boldsymbol{x}_{t_{i-1}} = \sqrt{\bar{\alpha}_{t_{i-1}}}\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) + \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i} + \sigma_{t_i}\boldsymbol{\epsilon}_{t_i} \tag{110}$$

where $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) = \frac{\boldsymbol{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i}}{\sqrt{\bar{\alpha}_{t_i}}}$ is the Tweedie estimate of the clean data, and $\boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the injected noise.

We can rewrite the conditional transition mean $\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i})$ as:

$$\begin{aligned}
\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) &= \sqrt{\bar{\alpha}_{t_{i-1}}}\left(\frac{\boldsymbol{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i}}{\sqrt{\bar{\alpha}_{t_i}}}\right) + \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i} \\
&= \sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\boldsymbol{x}_{t_i} - \left(\sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\sqrt{1-\bar{\alpha}_{t_i}} - \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\right)\boldsymbol{\epsilon}^\theta_{t_i} \\
&= \sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\boldsymbol{x}_{t_i} + \underbrace{\left(\sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\sqrt{1-\bar{\alpha}_{t_i}} - \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\right)\sqrt{1-\bar{\alpha}_{t_i}}}_{\Omega(t_i)}\boldsymbol{s}^\theta_{t_i}.
\end{aligned} \tag{111}$$

D.1.2 SDE-DPM-Solver++ (VP-SDE)

To further demonstrate the generality of our framework, we derive $\Omega(t_i)$ for SDE-DPM-Solver++ [28], a higher-order solver supported by Residual ∇-DB.

Let $a_t = \sqrt{\bar{\alpha}_t}$ and $b^2_t = 1 - a^2_t$ for the brevity. Let $\lambda_t = \log(a_t/b_t)$ be log-SNR and $h = \lambda_{t_{i-1}} - \lambda_{t_i}$. For SDE-DPM-Solver++ 1 sampler applied to a VP-SDE, the transition from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_{i-1}}$ is defined as:

$$\boldsymbol{x}_{t_{i-1}} = \frac{b_{t_{i-1}}}{b_{t_i}}e^{-h}\boldsymbol{x}_{t_i} + a_{t_{i-1}}(1-e^{-2h})\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) + b_{t_{i-1}}\sqrt{1-e^{-2h}}\boldsymbol{\epsilon}_{t_i} \tag{112}$$

where $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) = \frac{\boldsymbol{x}_{t_i} - b_{t_i}\boldsymbol{\epsilon}^\theta_{t_i}}{a_{t_i}} = \frac{\boldsymbol{x}_{t_i} + b^2_{t_i}\boldsymbol{s}^\theta_{t_i}}{a_{t_i}}$ denotes the Tweedie estimate of the clean data and $\boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(0, \mathbf{I})$ is the injected noise. Thus,

$$\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) = \left(\frac{b_{t_{i-1}}}{b_{t_i}}e^{-h} + \frac{a_{t_{i-1}}}{a_{t_i}}(1-e^{-2h})\right)\boldsymbol{x}_{t_i} + \frac{a_{t_{i-1}}}{a_{t_i}}(1-e^{-2h})b^2_{t_i}\boldsymbol{s}^\theta_{t_i}. \tag{113}$$

The covariance for this step is $\boldsymbol{\Sigma}_{t_i} = b^2_{t_{i-1}}(1-e^{-2h})\mathbf{I} := \sigma_{t_i}\mathbf{I}$. By comparing this result with the form $\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$, we have

$$\Omega(t_i) = \frac{a_{t_{i-1}}}{a_{t_i}}\left(1-e^{-2h}\right)b^2_{t_i} = \frac{a_{t_{i-1}}}{a_{t_i}}\frac{b^2_{t_i}}{b^2_{t_{i-1}}}\left(1-e^{-2h}\right)b^2_{t_{i-1}} = \sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\frac{1-\bar{\alpha}_{t_i}}{1-\bar{\alpha}_{t_{i-1}}}\sigma_{t_i}. \tag{114}$$

D.1.3 Euler Discrete Sampler (Flow-SDE)

For the Euler Discrete sampler applied to Rectified Flow models (e.g., Liu et al. [24]), we consider the reverse SDE that shares the same marginal densities as the probability flow ODE. The reverse SDE step from t_i to t_{i-1} (where $t_{i-1} < t_i$, and we define $\Delta t_i = t_i - t_{i-1}$) takes the form:

$$\boldsymbol{x}_{t_{i-1}} = \underbrace{\boldsymbol{x}_{t_i} - \Delta t_i \boldsymbol{v}^\theta_{t_i}(\boldsymbol{x}_{t_i}) + \frac{\sigma^2_{t_i}}{2}\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})}_{\boldsymbol{\mu}_{t_{i-1}}} + \sigma_{t_i}\boldsymbol{\epsilon}_{t_i}, \tag{115}$$

where $\boldsymbol{v}^\theta_{t_i}(\boldsymbol{x}_{t_i})$ is the velocity field prediction, $\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$ is the score prediction, and $\boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the injected noise[19].

In the rectified flow formulation, the velocity field and the marginal score are related by:

$$\boldsymbol{v}_{t_i}(\boldsymbol{x}_{t_i}) = -\frac{1}{1-t_i}\boldsymbol{x}_{t_i} - \frac{t_i}{1-t_i}\boldsymbol{s}_{t_i}(\boldsymbol{x}_{t_i}). \tag{116}$$

This is consistent with the Tweedie estimate $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) = \frac{\boldsymbol{x}_{t_i} + \boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})t_i^2}{1-t_i} = \boldsymbol{x}_{t_i} - \boldsymbol{v}^\theta_{t_i}(\boldsymbol{x}_{t_i})t_i$.

Substituting the expression for $\boldsymbol{v}^\theta_{t_i}$ into the conditional transition mean $\boldsymbol{\mu}_{t_{i-1}}$, we get:

$$\begin{aligned}
\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) &= \boldsymbol{x}_{t_i} - \Delta t_i \left(-\frac{1}{1-t_i}\boldsymbol{x}_{t_i} - \frac{t_i}{1-t_i}\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})\right) + \frac{\sigma^2_{t_i}}{2}\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i}) \\
&= \left(1 + \frac{\Delta t_i}{1-t_i}\right)\boldsymbol{x}_{t_i} + \left(\frac{t_i}{1-t_i}\Delta t_i + \frac{\sigma^2_{t_i}}{2}\right)\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i}).
\end{aligned} \tag{117}$$

We aim to express the guidance term via the transition mean in the form $\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$.

Equating the coefficients of the score term $\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$, we have:

$$\Omega(t_i) = \frac{t_i}{1-t_i}\Delta t_i + \frac{\sigma^2_{t_i}}{2}. \tag{118}$$

Note that $\sigma_{t_i} = \tilde{\sigma}_{t_i}\sqrt{\Delta t_i}$, where $\tilde{\sigma}_{t_i} = a\sqrt{\frac{t_i}{1-t_i}}$ for Flow-GRPO SDE and $\tilde{\sigma}_{t_i} = a$ for Dance-GRPO SDE.

This derivation clarifies how the intrinsic parameters of the flow model (t_i) and the chosen noise scale (σ_{t_i}) jointly determine the magnitude of value guidance $\Omega(t_i)$ required to navigate the reverse SDE towards the soft-optimal policy.

D.2 δ Derivations for Common Parameterizations

We derive $\delta(t_i)$, the signed scalar relating the parameterization difference to the score difference, for cases used in practice in this paper.

$\boldsymbol{\epsilon}$-prediction, VP-SDE. Under the forward marginal, the score satisfies

$$\boldsymbol{s}_{t_i} = -\frac{1}{\sqrt{1-\bar{\alpha}_{t_i}}}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} = -\sqrt{1-\bar{\alpha}_{t_i}}\,\boldsymbol{s}_{t_i}. \tag{119}$$

Therefore,

$$\boldsymbol{\epsilon}^\theta_{t_i} - \boldsymbol{\epsilon}^{\text{ref}}_{t_i} = -\sqrt{1-\bar{\alpha}_{t_i}}\big(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}\big).$$

Comparing this with

$$\boldsymbol{\epsilon}^\theta_{t_i} - \boldsymbol{\epsilon}^{\text{ref}}_{t_i} = -\frac{1}{\delta(t_i)}\big(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}\big),$$

we obtain, for VP-SDE diffusion models,

$$\delta(t_i) = \frac{1}{\sqrt{1-\bar{\alpha}_{t_i}}}. \tag{120}$$

[19] Here, σ_t follows the definition in Lemma 3.1. Note that $\sigma_t = \tilde{\sigma}_t(\Delta t)$.

v-prediction, Rectified Flow. Consider the general forward marginal $p_t(\boldsymbol{x}_{t_i} \mid \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t_i}; a_{t_i}\boldsymbol{x}_0, b_{t_i}^2\mathbf{I})$, which induces the trajectory $\boldsymbol{x}_{t_i} = a_{t_i}\boldsymbol{x}_0 + b_{t_i}\boldsymbol{\epsilon}$. Differentiating with respect to time gives $\boldsymbol{v}_{t_i} \triangleq \frac{d\boldsymbol{x}_{t_i}}{dt} = \dot{a}_{t_i}\boldsymbol{x}_0 + \dot{b}_{t_i}\boldsymbol{\epsilon}$. Substituting $\boldsymbol{x}_0 = \frac{1}{a_{t_i}}(\boldsymbol{x}_{t_i} - b_{t_i}\boldsymbol{\epsilon})$ to eliminate the clean data yields

$$\boldsymbol{v}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\dot{a}_{t_i}}{a_{t_i}}\boldsymbol{x}_{t_i} + \left(\dot{b}_{t_i} - \frac{\dot{a}_{t_i}b_{t_i}}{a_{t_i}}\right)\boldsymbol{\epsilon}.$$

Using the score identity (Eq. (119)), this becomes

$$\boldsymbol{v}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\dot{a}_{t_i}}{a_{t_i}}\boldsymbol{x}_{t_i} + \left(\frac{\dot{a}_{t_i}b_{t_i}^2 - a_{t_i}\dot{b}_{t_i}b_{t_i}}{a_{t_i}}\right)\boldsymbol{s}_{t_i}(\boldsymbol{x}_{t_i}).$$

The coefficient of $\boldsymbol{x}_{t_i}$ depends only on the fixed forward schedule and therefore contains no learned parameters. Hence, when taking the difference between the learned policy and the reference policy, the $\boldsymbol{x}_{t_i}$ terms cancel exactly, giving

$$\boldsymbol{v}_{t_i}^{\theta} - \boldsymbol{v}_{t_i}^{\text{ref}} = \left(\frac{\dot{a}_{t_i}b_{t_i}^2 - a_{t_i}\dot{b}_{t_i}b_{t_i}}{a_{t_i}}\right)\left(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right).$$

Combining this with

$$\boldsymbol{v}_{t_i}^{\theta} - \boldsymbol{v}_{t_i}^{\text{ref}} = -\frac{1}{\delta(t_i)}\left(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right),$$

we obtain

$$\delta(t_i) = -\frac{a_{t_i}}{\dot{a}_{t_i}b_{t_i}^2 - a_{t_i}\dot{b}_{t_i}b_{t_i}}.$$

For Rectified Flow, where $a_{t_i} = 1 - t_i$ and $b_{t_i} = t_i$, this reduces to

$$\delta(t_i) = \frac{1 - t_i}{t_i}. \tag{121}$$

D.3 Sampler-dependent weights w

Table 1 shows $h(t_i)$, which summarizes timestep-wise optimization strength. For lookahead methods, a convenient bridge to this quantity is the sampler-dependent weight

$$w(t_i) := \frac{\Omega(t_i)\delta(t_i)}{\sigma_{t_i}} \quad \Longleftrightarrow \quad \Omega(t_i)\delta(t_i) = w(t_i)\sigma_{t_i}. \tag{122}$$

Substituting Eq. (122) into the lookahead form of $h(t_i)$ yields, for $C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$,[20]

$$h(t_i) = \delta(t_i)C_1(t_i)\gamma(t_i)\frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)} = \frac{\gamma(t_i)}{2}\delta(t_i)\Omega(t_i) = \frac{\gamma(t_i)}{2}w(t_i)\sigma_{t_i} \propto w(t_i)\sigma_{t_i}. \tag{123}$$

Thus, $w(t_i)$ provides a convenient sampler-level lens for understanding how different design choices reschedule effective optimization strength across timesteps. See Figure 2 of Lee and Ye [20] for the shapes of $w(t_i)$ under different samplers.

We now use PCPO-reweight as a case study. Its key modification is to replace the native schedule induced by $w(t_i)$ with a flatter alternative $w'(t_i)$, thereby reshaping $h(t_i)$.

DDIM sampler, ϵ-prediction (SD1.5). PCPO defines

$$w(t_i) = \frac{\frac{\sqrt{1-\bar{\alpha}_{t_i}}}{\sqrt{\alpha_{t_i}}} - \sqrt{1 - \bar{\alpha}_{t_{i-1}} - \sigma_{t_i}^2}}{\sigma_{t_i}}, \tag{124}$$

which satisfies Eq. (122) together with Ω from Eq. (111) and δ from Eq. (120). PCPO then modifies the noise schedule from σ_{t_i} to σ'_{t_i} so that the resulting weights become approximately constant,

[20] Residual ∇-DB is slightly different: since $C_1(t_i) = \frac{1}{d}\frac{\Omega(t_i)^2}{\sigma_{t_i}^4}$, the same substitution yields $h(t_i) \propto w(t_i)\sigma_{t_i}^{-1}$ rather than $w(t_i)\sigma_{t_i}$. The role of $w(t_i)$ as a useful sampler-level lens remains analogous.

Table 4: **Summary of experiment settings for Section 5.2.**

Experiment setting	Baseline	Base model	Reward model	Prompts	Max epoch
Mechanistic Analysis	TempFlow-GRPO	SD3.5-M	Aesthetic Score	Pick-a-Pic	150
Validation: ZO, Flow	TempFlow-GRPO	SD3.5-M	GenEval	GenEval	420
Validation: ZO, Diffusion	PCPO	SD1.5	HPSv2.1	HPDv2	300
Validation: FO, Flow	VGG-Flow	SD3.5-M	HPSv2.1	GenEval	200
Validation: FO, Diffusion	Residual ∇-DB	SD1.5	HPSv2.1	GenEval	100

$w'(t_i) \approx w_{\text{const}}$. Accordingly, the induced schedule changes from $\frac{1}{2}\sigma_{t_i} w(t_i)$ to $\frac{1}{2}\sigma'_{t_i} w'(t_i)$. To avoid confounding this reweighting with a global change in loss scale, PCPO normalizes the modified schedule so that $\text{avg}(w') = \text{avg}(w)$.

Euler discrete sampler, v-prediction (SD3). Here PCPO keeps the sampler noise schedule fixed and instead reweights the native timestep weights directly. Writing $\sigma_{t_i} = \tilde{\sigma}_{t_i}\sqrt{\Delta t_i}$, where $\tilde{\sigma}_{t_i}$ is the instantaneous diffusion coefficient in the Itô SDE, the standard weight is

$$w(t_i) = \frac{\sqrt{\Delta t_i}}{\tilde{\sigma}_{t_i}} \left(1 + \frac{(1-t_i)\tilde{\sigma}_{t_i}^2}{2t_i}\right)^2, \tag{125}$$

which again satisfies Eq. (122) with Ω from Eq. (118) and δ from Eq. (121). PCPO replaces this native schedule by $w'(t_i) = w_{\text{const}}\Delta t_i$, so the induced weighting changes from $\frac{1}{2}\sigma_{t_i} w(t_i)$ to $\frac{1}{2}\sigma_{t_i} w'(t_i)$. As in the DDIM case, PCPO matches the average scale by enforcing $\text{avg}(w') = \text{avg}(w)$.

E Experiment Details

E.1 Toy experiments.

To conduct the experiments in Section 5.1, we designed a base data distribution on 2 dimensional space $p_0^{\text{ref}}(\boldsymbol{x}_0)$ as mixture of three Gaussian nodes with unit covariance:

$$p_0^{\text{ref}}(\boldsymbol{x}_0) := \frac{1}{3}\left\{\mathcal{N}\left(\begin{bmatrix}3\\-\sqrt{3}\end{bmatrix}, \mathbf{I}\right) + \mathcal{N}\left(\begin{bmatrix}-3\\-\sqrt{3}\end{bmatrix}, \mathbf{I}\right) + \mathcal{N}\left(\begin{bmatrix}0\\2\sqrt{3}\end{bmatrix}, \mathbf{I}\right)\right\} \tag{126}$$

We defined our reward $r(\boldsymbol{x}_0) := \boldsymbol{x}_0[0]/2 + 3$ as a linear function, which allows the reward-tilted distribution $p_0^*(\boldsymbol{x}_0) \propto p_0^{\text{ref}}(\boldsymbol{x}_0)e^{r(\boldsymbol{x}_0)}$ to be another mixture of Gaussians. The noised marginal distributions $p_t^{\text{ref}}(\boldsymbol{x}_t)$ and $p_t^*(\boldsymbol{x}_t)$ can be obtained in a closed form as an interpolation between data Guassians and noise prior $\mathcal{N}(0, \mathbf{I})$. Their score function difference is used as a ground truth for the soft value function gradient according to Eq. (11).

We trained diffusion models for $p_0^{\text{ref}}(\boldsymbol{x}_0)$ and $p_0^*(\boldsymbol{x}_0)$ with ϵ-prediction on 500 timesteps. The training was done with batch size 4096 for 2000 iterations, using Adam optimizer with learning rate 10^{-3}. To simulate the value gradient estimator for different reward alignment training methods, DDIM sampling was done with timestep interval 10 with Markovian stochasticity.

E.2 End-to-end Design.

Table 4 summarizes experiment settings used in Section 5.2. Below we record the method-specific choices omitted from the main text. Unless otherwise specified, we use the default configurations of the corresponding baseline. Although we used H200 GPUs for some experiments to reduce wall-clock time, all experiments are reproducible on GPUs with 24GB VRAM except for the non-pruned VGG-Flow baseline.

Mechanistic analysis (ZO, flow). We follow TempFlow-GRPO on SD3.5-M [8] with Aesthetic Score [35] reward and Pick-a-Pic [19] prompts. Each epoch uses 16 prompts with 4 seeds per prompt. For entropy regularization, we use $\alpha = 0.01$ and follow the heuristic in the Flow-GRPO repository, which temporally weights the entropy term by Δt_i. The baseline branching profile is $K = [6, 6, 6, 6, 6, 6, 6, 6, 6]$. For the redesigned variants, the Uniform Reallocation profile is $K = [7, 7, 7, 7, 7, 7, 6, 6, 0]$, and the Equal Variance Reallocation profile is $K = [4, 5, 5, 6, 7, 8, 9, 10, 0]$.

Following the policy-gradient setting, we use the group-normalized advantage $\hat{A}$ in place of raw reward r; see Appendix G.3. Evaluation was performed every 30th epoch on a fixed set of prompts. Wall-clock time is reported in H200 hours; 2 H200 GPUs were used for experiments.

Validation: ZO, flow. We follow the batch setting of TempFlow-GRPO, using 48 prompts per epoch with 4 seeds per prompt. For entropy regularization, we use $\alpha = 0.004$ and do *not* apply the Flow-GRPO Δt_i heuristic. Because GenEval is primarily semantic, our redesign does not invest branching budget at high-SNR timesteps; instead, it concentrates compute on the first few low-SNR timesteps, where semantic guidance is most useful. Concretely, we use the branching profile $K = [6, 6, 8, 10, 0, 0, 0, 0, 0]$, together with GRPO-Guard temporal reweighting and the fair clipping rule in Eq. (30). For non-branching timesteps, we add a KL penalty anchoring to $\boldsymbol{s}^{\text{ref}}_{t_i}$ using ODE trajectories, which avoids additional sampling cost for anchor trajectories. We use the same α for these ODE anchors. As above, we use the group-normalized advantage $\hat{A}$ in place of reward r. Wall-clock time is measured excluding GenEval reward computation. For the baseline, wall-clock time is estimated from the per-step time required under the TempFlow-GRPO setting. Our method was evaluated every 60th epoch, while the TempFlow-GRPO baseline reported evaluations every 20th epoch. Evaluation was performed on the GenEval test prompt dataset, separate from the training set. Wall-clock time is reported in H200 hours; 4 H200 GPUs were used for experiments.

Validation: ZO, diffusion. The baseline PCPO can be viewed as using the uniform profile $K_i \equiv 1$, i.e., a single sampled continuation at every reverse step. Ideally, one would branch at every reverse step, but doing so over the full trajectory ($N = 50$) is prohibitively expensive. We therefore branch at only 10% of reverse steps, leaving the remaining steps unbranched. Specifically, for a roughly matched per-epoch compute budget, we use 8 prompts per epoch with 5 seeds per prompt, and set $K_i = 4$ at every 10th timestep. For non-branching timesteps, we add a small $\|\boldsymbol{s}^\theta - \boldsymbol{s}^{\text{ref}}\|^2$ regularizer on 10% of ODE rows to prevent uncontrolled drift from the reference model, with $\alpha = 8 \times 10^{-5}$. We fine-tune SD1.5 [34] with LoRA [15] rank 16. Evaluation was performed every 10th epoch on a fixed set of prompts. As above, we use the group-normalized advantage $\hat{A}$ in place of reward r. Wall-clock time is reported in RTX4090 hours; 4 RTX4090 GPUs were used for experiments.

Validation: FO, flow. We follow the batch setting of VGG-Flow, using 32 generation trajectories and one update per epoch. The baseline relies on local Tweedie-based reward gradients, whereas our redesign uses terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$, which effectively correspond to using $j = 0$ reward information. The exact unbiased quantity would be $\nabla_{\boldsymbol{x}_{t_j}} r(\boldsymbol{x}_0)$, but computing it requires an expensive high-order Jacobian chain. Following score distillation approaches, we therefore use $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ as a practical surrogate. To counteract inherited downweighting of low-SNR timesteps, we replace the baseline schedule $\tilde{\gamma}_{t_i} = (1 - t_i)^2$ with $\tilde{\gamma}_{t_i} = \frac{1-t_i}{2}$. We plot *transition mean drift*, i.e. $\mathbb{E}[\|\boldsymbol{\mu}^\theta - \boldsymbol{\mu}^{\text{ref}}\|^2]$, which is analogous to the KL divergence defined for lookahead methods, i.e. $\mathbb{E}[\|\boldsymbol{\mu}^\theta - \boldsymbol{\mu}^{\text{ref}}\|^2/2\sigma^2]$. Evaluation was perfomed every 50th epoch on the GenEval test dataset. As we observed substantial variability across random seeds in this setting, we report results averaged over three runs with different seeds. Wall-clock time is reported in H200 hours; 4 H200 GPUs were used for experiments.

Validation: FO, diffusion. Both the baseline and our method collect 256 generation trajectories. For training, we subsample 20% of transitions from each collected trajectory by uniformly splitting the trajectory into timestep intervals and sampling one transition from each interval, while always including the final transition, following Liu et al. [27]. The baseline again uses local Tweedie-based reward gradients, while our redesign uses terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ as a practical surrogate for the exact quantity $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$. This effectively corresponds to using $j = 0$ reward information without differentiating through the full Jacobian chain. To reverse the inherited low-SNR downweighting, we multiply $\ell^\theta_{t_i}$ by $3\sigma_{t_i}$ and set $\tilde{\gamma}_{t_i} = \frac{3}{2}\sigma_{t_i}$. Evaluation was perfomed every 50th epoch on the GenEval test dataset. Wall-clock time is reported in RTX4090 hours; 4 RTX4090 GPUs were used for experiments.

E.3 Additional Experiments

Figures 8–11 were obtained using the original codebases of Residual ∇-DB [27] and VGG-Flow [26], with Aesthetic Score as the reward.

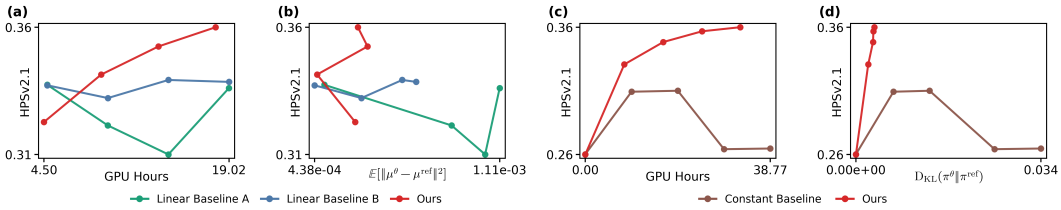

Figure 6: **Ablating the first-order estimator.** Replacing $\nabla_{\boldsymbol{x}_t} r(\hat{\boldsymbol{x}}_0)$ with $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ improves both reward efficiency and the reward–KL tradeoff. In flow matching, we compare against two linearized baselines that keep the original local Tweedie-based estimator but adopt milder temporal weighting: (a) reward vs. GPU hours; (b) reward vs. KL. In diffusion, we compare against the corresponding baseline with the original estimator under the modified weighting: (c) reward vs. GPU hours; (d) reward vs. KL.

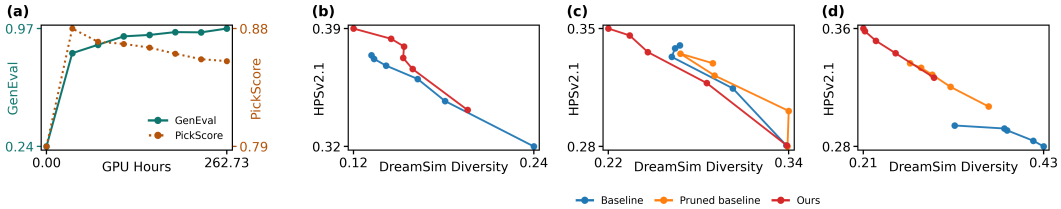

Figure 7: **Auxiliary metrics suggest no obvious reward hacking.** (a) PickScore remains stable throughout GenEval zeroth-order flow-matching fine-tuning. (b–d) DreamSim diversity on HPSv2.1 for zeroth-order diffusion, first-order flow matching, and first-order diffusion, respectively.

F Additional Results

F.1 Ablations for First-Order Experiments

To isolate the contribution of the modified value-guidance estimator $\boldsymbol{\Psi}_{t_i}$, we compare our full method against ablated variants that retain the baseline first-order estimator based on $\nabla_{\boldsymbol{x}_t} r(\hat{\boldsymbol{x}}_0)$ rather than $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$. In flow matching, our redesign changes both the estimator and the temporal weighting: it replaces the local Tweedie-based gradient with terminal-image reward gradients, and replaces the baseline weighting $\tilde{\gamma}(t_i) = (1 - t_i)^2$ with a milder low-SNR schedule. Since the original VGG-Flow paper already observed that a linear schedule can outperform $(1 - t_i)^2$ in some settings, Fig. 6(a, b) further compares against two linearized baselines to test whether our gains come solely from the weighting change. *Linear Baseline A* keeps the baseline estimator and uses $\tilde{\gamma}(t_i) = (1 - t_i)$, while *Linear Baseline B* keeps the baseline estimator and uses our scaled linear schedule. Our full method still performs substantially better, showing that the improvement is not explained by temporal weighting alone, but critically depends on the estimator change itself. In diffusion, Fig. 6(c, d) shows the analogous comparison against a baseline that keeps the original local Tweedie-based estimator under the same modified weighting. Overall, replacing $\nabla_{\boldsymbol{x}_t} r(\hat{\boldsymbol{x}}_0)$ with $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ consistently improves both reward efficiency and the reward–KL tradeoff in flow-matching and diffusion settings.

F.2 Reward Hacking

We next examine whether the redesigns introduced in Section 5.2 improve the target reward at the cost of substantially degraded sample quality or diversity. Overall, we do not observe evidence that our improvements are driven by excessive reward hacking relative to the corresponding baselines.

For the GenEval zeroth-order flow-matching experiments, direct qualitative comparison to the baseline is unavailable because the original baseline checkpoints were not retained. Instead, we track PickScore on the validation set throughout training. As shown in Figure 7(a), PickScore remains stable rather than collapsing during optimization, suggesting that the GenEval gains are not accompanied by an obvious degradation in this auxiliary preference-based metric.

For other settings, we evaluate the tradeoff between reward and diversity by plotting reward–DreamSim Diversity [10] Pareto frontiers. Fig. 7(b)–(d) show that our method is not worse than the baseline on this frontier, indicating that the reward gains do not come at the cost of a systematically

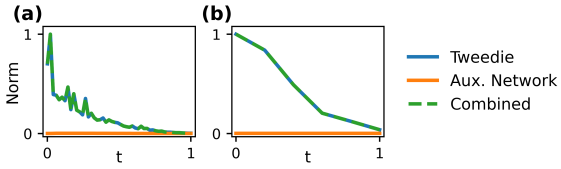

Figure 8: $\|g_\phi\|$ **is negligible.** The learned refinement term g_ϕ is negligible compared to the analytic reward gradient throughout the entire generation process for both (a) Residual ∇-DB, (b) VGG-Flow.

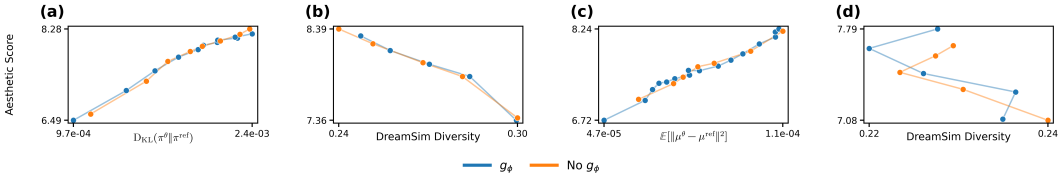

Figure 9: g_ϕ **is redundant for training.** Removing g_ϕ reduces wall-clock time, while maintaining optimality on the tradeoff between reward and prior/diversity preservation. Averaged across three consecutive random seeds. (a, b) Residual ∇-DB, (c, d) VGG-Flow[21].

worse diversity profile. Taken together, these results suggest that the improvements from our redesigns reflect better optimization of the intended objective rather than merely exploiting weaknesses of the reward model.

Remark on Fig. 5(b). Although the non-pruned VGG-Flow baseline attains a slightly better reward–transition-drift Pareto frontier in Fig. 5(b), we do not view this as evidence that the auxiliary network g_ϕ is essential. First, g_ϕ may induce nontrivial interactions in optimization dynamics that make the unpruned model slightly easier to stabilize, but this modest benefit comes at a substantial systems cost: the non-pruned baseline requires roughly twice the VRAM and significantly longer wall-clock time, making it a poor tradeoff in practice. In our view, comparable gains would be better pursued through simpler knobs such as a smaller learning rate or a larger effective batch size in the pruned baseline, rather than through an additional auxiliary network. Second, among the first-order baselines, VGG-Flow exhibited the largest seed-to-seed variability, so advantages at earlier stages of training should be interpreted cautiously. Third, auxiliary evaluations based on DreamSim Diversity in Fig. 7(d) show that the pruned and unpruned baselines are comparable, with the pruned baseline slightly better if anything. Taken together, these observations support our main claim: g_ϕ is not a core ingredient of effective optimization, but an auxiliary component that adds substantial complexity for at most modest and inconsistent gains.

G Auxiliary Designs of Prior Work

G.1 Redundancy of Auxiliary Components

Our unified framework posits that the core optimization signal is fully captured by Eq. (13). This implies that auxiliary components common in prior work—specifically refinement networks and backward losses—are theoretically redundant or even harmful. We empirically validate that stripping these components simplifies the algorithm and reduces computational overhead without compromising performance; note that our findings are consistent with Liu et al. [25].

Redundancy of Refinement Networks (g_ϕ). Residual ∇-DB and VGG-Flow employ a learnable network g_ϕ to compensate for Tweedie approximation errors (i.e., by learning the residual $\nabla V - \nabla r(\hat{\boldsymbol{x}}_0)$). However, since g_ϕ is trained primarily via a terminal boundary loss ($g_\phi(\boldsymbol{x}_0) \to 0$) without intermediate supervision, it collapses toward zero due to the spectral bias of neural networks [33]. Empirical measurements in Fig. 8 confirm that the learned signal is negligible relative to the Tweedie gradient: $\|g_\phi\| \ll \frac{\tilde{\gamma}_t}{\alpha}\|\nabla r(\hat{\boldsymbol{x}}_0)\|$. Consequently, ablations in Fig. 9 demonstrate that removing g_ϕ maintains the Pareto frontier between reward maximization and prior preservation while strictly

[21] Due to missing checkpoints, panels (c) and (d) are reported from different runs.

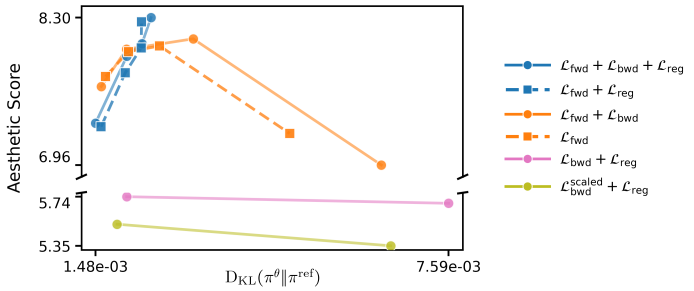

Figure 10: $\mathcal{L}_{\text{backward}}$ **of Residual ∇-DB does not contribute to effective training.**

reducing training time. This confirms g_ϕ contributes computational overhead without algorithmic benefit.

Instability of Backward Loss ($\mathcal{L}_{\text{backward}}$). Residual ∇-DB incorporates a backward loss $\mathcal{L}_{\text{backward}}$ derived from detailed balance conditions. As detailed in Appendix C.1.1, this term introduces high-order Jacobian dependencies that are analytically prone to variance explosion. Results in Fig. 10 confirm that $\mathcal{L}_{\text{backward}}$ induces training instability when its weight is non-negligible, whereas the forward gradient alone is sufficient for stable convergence. This suggests the backward consistency loss is practically unnecessary, adding volatility rather than value.

G.2 Revisiting Existing Justifications for Attenuation Coefficients $\tilde{\gamma}_t$

Beyond the limitations of soft value function approximation—which can collapse when the terminal loss drives $g_\phi \approx \mathbf{0}$ (Section G.1)—the specific attenuation schedules $\tilde{\gamma}_t$ used in prior work remain only partially justified.

Residual ∇-DB. Motivated by GFlowNet theory, this method uses the forward-looking (FL) trick [30] to decompose $\nabla_{x_t} \log \tilde{F}(x_t)$ into a single-step Tweedie estimate and a residual term g_ϕ. However, this derivation relies on assumptions that are difficult to satisfy for complex image-generation rewards, such as computable intermediate energy or additive energy structure. Even if one interprets the "energy" more broadly as expected future reward, the resulting $\tilde{\gamma}_t$ schedule still appears heuristic. A simpler interpretation is our RSM perspective; the method attenuates the score-matching objective where the Tweedie estimate is less reliable.

VGG-Flow. VGG-Flow likewise introduces $\tilde{\gamma}_t$ and decomposes the gradient into a Tweedie term and a learnable correction g_ϕ, but lacks theoretical justification for the weighting itself. At the same time, the paper explicitly draws inspiration from score distillation [31], which is compatible with our score-matching interpretation.

SQDF. SQDF, motivated by Soft RL, interprets $\tilde{\gamma}_t$ as a credit-assignment coefficient, assigning little weight to high-noise regimes under the intuition that these states have lower signal-to-noise ratio and therefore weaker influence on the final sample. However, this intuition is not well aligned with the standard understanding of diffusion models, where low-SNR stages are often responsible for global semantic structure, and early denoising decisions can strongly affect the final reward [5]. Empirically, their intuition is also inconsistent with the strong performance of Flow-GRPO-Fast [24], which trains only on low-to-mid-SNR timesteps, yet outperforms its full-trajectory counterpart.

Overall, these observations suggest that interpreting $\tilde{\gamma}_t$ primarily as a damping factor for approximation error is more consistent with both prior empirical results and the known behavior of diffusion models.

G.3 Additional Design Considerations

We adopt specific simplifications to isolate the effects of our proposed components.

Reward Normalization. $\eta(t)$ effectively controls the scale of the estimated reward gradient. Residual ∇-DB and VGG-Flow tune this quantity indirectly, through reward-specific choices of α or C_2, to improve stability. Motivated by the canonical gradient form (Eq. (15)), we suggest that first-order methods may reduce this hyperparameter sensitivity by explicitly scaling $\eta(t)$ with $\frac{1}{\|\nabla r\|}$.

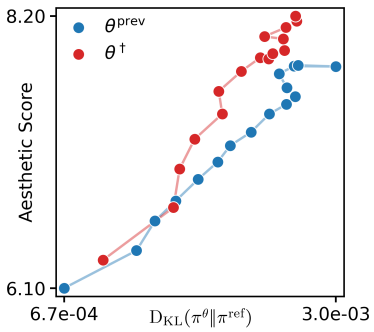

Figure 11: **Online samples suffice.** Including past rollouts (offline buffer) does not improve the Pareto frontier for Residual ∇-DB.

For zeroth-order methods, the reward-gradient estimator takes the form $\frac{1}{\sigma_{t_i}}\mathbb{E}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}]$. This perspective helps clarify why reward normalization can substantially improve optimization. First, subtracting the group mean acts as a control variate. Replacing r with $r-\hat{\mu}_G$ reduces estimator variance, while also changing the optimization geometry: instead of merely inducing stronger or weaker attraction toward high- and low-reward regions, it produces push–pull dynamics relative to the group average, which is often easier to optimize. Second, dividing by a scale term helps make the magnitude of the estimator more comparable across reward functions. In particular, replacing r with $\hat{A}=\frac{r-\hat{\mu}_G}{\hat{\sigma}_G}$ makes the scale of $\mathbb{E}[\hat{A}\boldsymbol{\epsilon}_{t_i}]$ roughly comparable across different rewards, and therefore eases hyperparameter tuning. This same scale-normalization principle also helps explain the empirical success of the Flow-GRPO implementation, which normalizes rewards to be approximately in $[0,1]$. More broadly, the results of Choi et al. [4] suggest that dividing by the average magnitude $\|r\|$, or by the sample mean $\|\hat{\mu}_G\|$ as a practical surrogate, can further stabilize training.

Trust Regions. Trust-region mechanisms, including the quadratic penalty C_2 and clipping, should be viewed as heuristics rather than essential components.

For example, GRPO-Guard effectively sets $C_2(t_i)=0$ while remaining stable. This suggests that the RSM objective can sometimes be simplified further without sacrificing performance, although doing so appears to require nontrivial engineering.

A similar picture emerges for clipping. EPG reports that, in some settings, removing clipping does not hurt performance. In contrast, we find that in certain Flow-GRPO settings, naively increasing the clip range ξ destabilizes training. Moreover, PCPO observes that standard PPO-style clipping can reduce the effective batch size and thereby harm performance [20]. This issue is particularly severe in Flow-SDEs, where the singular behavior near $t\approx 0$ causes nearly all samples to be clipped, as noted by Wang et al. [45].

Overall, these observations suggest that trust-region design remains an open heuristic space. Further simplification is possible in some regimes, but robust performance across settings likely depends on better-engineered heuristics.

Dataset $\mathcal{D}$: Focus on Online Samples. While prior work augments training with offline replay buffers [18, 27], empirical evidence suggests this offers minimal benefit. SQDF reports negligible difference in performance, and our ablations with Residual ∇-DB (Fig. 11) show that including past rollouts can actually degrade the Pareto frontier. Thus, we train exclusively on online samples.

Incorporation of CFG in Training Loss. Although we use CFG in all experiments, it is not a necessary component. Since CFG [13] amplifies the variance of score estimates, it can destabilize training [4, 24, 50]. In some settings, it may therefore be preferable to exclude CFG from the training loss.

H Extension of RSM Framework

H.1 Connection with Entropy-regularized Optimal Control Perspective [32, 42]

We begin from the soft RL objective defined over path measures:

$$V(\tau)=\max_{u,\nu}\left[\mathbb{E}_{\mathbb{P}_{u,\nu}}[r(\boldsymbol{x}_0)]-\alpha\mathcal{D}_{\text{KL}}(\mathbb{P}_{u,\nu}(\tau)\|\mathbb{P}_{\text{ref}}(\tau))\right]. \tag{127}$$

Here, $\mathbb{P}_{\text{ref}}$ denotes the path measure induced by the pretrained diffusion model, while $\mathbb{P}_{u,\nu}$ denotes the path measure induced by fine-tuned diffusion model and initial distribution ν.

Note that fine-tuning a diffusion model can be interpreted as modifying the reference dynamics through a drift perturbation. Such perturbation admit a natural control interpretation, which enables us to reformulate the soft RL objective as an entropy-regularized optimal control problem in continuous time.

To formalize this view, we consider a reverse-time SDE with control $u : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$,

$$d\boldsymbol{x}_t = [f(\boldsymbol{x}_t, t) + u(\boldsymbol{x}_t, t)]dt + g(t)dw_t, \tag{128}$$

where $f(\boldsymbol{x}_t, t)$ denotes the drift coefficient including score function and $g(t)$ is an invertible diffusion coefficient. Our goal is to characterize the optimal control u that solves the above entropy-regularized problem. To obtain an explicit expression of this path-space KL divergence, we invoke the following form of Girsanov's theorem.

Girsanov Theorem Consider the diffusion process with path measure $\mathbb{P}_{\text{ref}}$,

$$dX_t = f(X_t, t)dt + g(t)dW_t, \tag{129}$$

where $f(X_t, t)$ denotes the drift coefficient, $g(t)$ is an invertible diffusion coefficient, and W_t denotes the Brownian motion under $\mathbb{P}_{\text{ref}}$. Let $u(X_t, t)$ be an adapted process satisfying Novikov's condition. Define the stochastic exponential

$$Y_T = \exp\left(\int_0^T (g(t)^{-1}u(X_t, t))^\top dW_t - \frac{1}{2}\int_0^T \|g(t)^{-1}u(X_t, t)\|^2 dt\right). \tag{130}$$

As Y_T is a positive martingale with unit expectation, it defines a new probability measure $\mathbb{P}_u$ via

$$\frac{d\mathbb{P}_u}{d\mathbb{P}_{\text{ref}}} = Y_T. \tag{131}$$

Then, the process

$$W_t^u = W_t - \int_0^t g(s)^{-1}u(X_s, s)ds \tag{132}$$

is a Brownian motion under $\mathbb{P}_u$, and the dynamics of X_t become

$$dX_t = [f(X_t, t) + u(X_t, t)]dt + g(t)dW_t^u \tag{133}$$

Now, from the definition of path-space KL divergence, we get

$$\mathcal{D}_{\text{KL}}(\mathbb{P}_{u,\nu}||\mathbb{P}_{\text{ref}}) = \mathbb{E}_{\mathbb{P}_{u,\nu}}\left[\log\frac{\mathbb{P}_{u,\nu}}{\mathbb{P}_{\text{ref}}}\right] = \mathbb{E}_{\mathbb{P}_{u,\nu}}\left[\log\frac{\nu}{\nu_{\text{ref}}} + \log\frac{\mathbb{P}_{u,\nu}(\cdot|\boldsymbol{x}_T)}{\mathbb{P}_{\text{ref}}(\cdot|\boldsymbol{x}_T)}\right] \tag{134}$$

where the second equality follows from the factorization of the path measure into the initial distribution $\nu_{\text{ref}} = p_T^{\text{ref}}$ and the conditional dynamics. The first term corresponds to the discrepancy in the initial distributions, and the second term measures the deviation of the controlled dynamics from the reference dynamics conditioned on the initial state. With fixed initial sample $\boldsymbol{x}_T$, by the Girsanov theorem, we obtain

$$\begin{aligned}
\mathbb{E}_{u,\nu}\left[\log\frac{\mathbb{P}_{u,\nu}(\cdot|\boldsymbol{x}_T)}{\mathbb{P}_{\text{ref}}(\cdot|\boldsymbol{x}_T)}\right] &= \mathbb{E}_{u,\nu}\left[\int_0^T \frac{u(\boldsymbol{x}_t, t)^\top}{g(t)}d\boldsymbol{w}_t - \frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right] \\
&= \mathbb{E}_{u,\nu}\left[\int_0^T \frac{u(\boldsymbol{x}_t, t)^\top}{g(t)}\left(d\boldsymbol{w}^u + \frac{u(\boldsymbol{x}_t, t)}{g(t)}dt\right) - \frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right] \\
&= \mathbb{E}_{u,\nu}\left[\frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right],
\end{aligned} \tag{135}$$

where the second equality holds due to Eq. (132) and the third equality holds as $\boldsymbol{w}_t^u$ is $\mathbb{P}_{u,\nu}$-martingale.

In consequence, the soft RL objective function is rewritten as

$$\max_{u,\nu} \mathbb{E}_{\mathbb{P}_{u,\nu}}[r(\boldsymbol{x}_0)] - \alpha \mathbb{E}_{\mathbb{P}_{u,\nu}} \left[\frac{1}{2} \int_0^T \| \frac{u(\boldsymbol{x}_t, t)}{g(t)} \|^2 dt + \log \frac{\nu}{\nu_{\text{ref}}} \right]. \tag{136}$$

Under the perspective of optimal control problem, this is equivalent to

$$\min_{u,\nu} \frac{\alpha}{2} \mathbb{E}_{\mathbb{P}_{u,\nu}} \underbrace{\left[\int_0^T \| \frac{u(\boldsymbol{x}_t, t)}{g(t)} \|^2 dt \right]}_{\text{running cost}} + \mathbb{E}_{\mathbb{P}_{u,\nu}} \underbrace{[-r(\boldsymbol{x}_0)]}_{\text{terminal cost}} + \mathbb{E}_{\mathbb{P}_{u,\nu}} \underbrace{\left[\log \frac{\nu}{\nu_{\text{ref}}} \right]}_{\text{initial cost}}. \tag{137}$$

The initial cost appears because the initial distribution is treated as an optimization variable, which is not standard in classical stochastic optimal control formulation.

Uehara et al. [42] has shown that the marginal density induced by Eq. (128) with optimal control and optimal initial distribution obtained from Eq. (137) is

$$p_t^*(\cdot) = \frac{1}{Z_{\text{tar}}} p_t^{\text{ref}}(\boldsymbol{x}) \exp\left(\frac{V_t^*(\boldsymbol{x})}{\alpha} \right) \tag{138}$$

where the optimal value function satisfies

$$\exp\left(\frac{V_t^*(\boldsymbol{x})}{\alpha} \right) = \mathbb{E}_{p^{\text{ref}}} \left[\exp\left(\frac{r(\boldsymbol{x}_0)}{\alpha} \right) | \boldsymbol{x}_t = \boldsymbol{x} \right] \tag{139}$$

and the optimal control is defined by

$$u^*(\boldsymbol{x}, t) = \frac{g^2(t)}{\alpha} V_t^*(\boldsymbol{x}) = g^2(t) \nabla_{\boldsymbol{x}} \log \mathbb{E}_{p^{\text{ref}}} \left[\exp\left(\frac{r(\boldsymbol{x}_0)}{\alpha} \right) | \boldsymbol{x}_t = \boldsymbol{x} \right] \propto \boldsymbol{\Psi}_{t_i}^\star(\boldsymbol{x}_{t_i}). \tag{140}$$

We emphasize that fine-tuning a diffusion model can be interpreted as perturbing the drift term via a control input. In this view, the fine-tuning model effectively learns to predict $\boldsymbol{s}_t^{\text{ref}} + u^*$, which corresponds to the target score function in the RSM framework. The optimal control perspective further provides an alternative route to estimating the value function using classical tools from optimal control theory, such as adjoint sampling.

H.2 Inference-Time Alignment

While the Reward Score Matching framework primarily addresses weight updates during training, its core principles are isomorphic to inference-time alignment strategies. DNO [40] steers diffusion models without parameter fine-tuning by directly optimizing the intermediate noise variables $\boldsymbol{\epsilon}_t$ to maximize a reward. To handle non-differentiable rewards, DNO employs the *Hybrid2* method, estimating the gradient of the reward with respect to the noise via zeroth-order perturbations.

We demonstrate that this approach is functionally identical to applying the zeroth-order value guidance estimator to the noise space. Recall that zeroth-order methods update model parameters θ using the Monte Carlo estimator:

$$\frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \mathbb{E}_{\boldsymbol{\epsilon}_{t_i:t_1}}[r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}]. \tag{141}$$

In parallel, DNO updates the latent variable $\boldsymbol{z}$ using the estimator (Eqs. (20, 21) in Tang et al. [40]):

$$\mathbb{E}_{\boldsymbol{\epsilon}} \Big[\frac{1}{\sigma} \big(r(D_\theta(\boldsymbol{z} + \sigma\boldsymbol{\epsilon})) - r(D_\theta(\boldsymbol{z})) \big) \big(D_\theta(\boldsymbol{z} + \sigma\boldsymbol{\epsilon}) - D_\theta(\boldsymbol{z}) \big) \Big] \cdot \nabla_{\boldsymbol{z}} D_\theta(\boldsymbol{z}), \tag{142}$$

where D_θ denotes the denoising network and σ denotes the degree of perturbation (not to be confused with variance schedule σ_{t_i}). Note that the update above is approximately equal to (Eqs. (18, 19) in Tang et al. [40]):

$$\mathbb{E}_{\boldsymbol{\epsilon}} \Big[\frac{1}{\sigma} \big(r(\boldsymbol{x} + \sigma\boldsymbol{\epsilon}) - r(\boldsymbol{x}) \big) \boldsymbol{\epsilon} \Big] \cdot \nabla_{\boldsymbol{z}} D_\theta(\boldsymbol{z}) \tag{143}$$

up to a scaling factor. This simplified form reveals a striking structural equivalence: both methods rely on a reward-weighted noise term ($\approx r \cdot \boldsymbol{\epsilon}$) to estimate the descent direction. This implies that training techniques, such as branching and signal-aware scheduling, can be repurposed to improve inference-time alignment.

H.3 DiffusionNFT

$$\mathcal{L}_{\mathrm{NFT}}(\theta) = \mathbb{E}_{\boldsymbol{x}_0\sim p^{\theta^\dagger},\,\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\frac{1+c}{2}\left\|(1-\beta)\boldsymbol{v}_t^{\theta^\dagger} + \beta\boldsymbol{v}_t^{\theta} - \boldsymbol{v}_t^{\mathrm{fwd}}\right\|^2 + \frac{1-c}{2}\left\|(1+\beta)\boldsymbol{v}_t^{\theta^\dagger} - \beta\boldsymbol{v}_t^{\theta} - \boldsymbol{v}_t^{\mathrm{fwd}}\right\|^2], \tag{144}$$

where $c = \mathrm{clip}_1(\hat{A}(\boldsymbol{x}_0)) \in [-1, 1]$ and $\boldsymbol{v}_t^{\mathrm{fwd}} = \boldsymbol{\epsilon} - \boldsymbol{x}_0$.

Differentiating Eq. (144) w.r.t. θ and collecting terms yields

$$\nabla_\theta \mathcal{L}_{\mathrm{NFT}}(\theta) = \mathbb{E}\Big[2\beta\Big(\beta\boldsymbol{v}_t^{\theta} + (c-\beta)\boldsymbol{v}_t^{\theta^\dagger} - c\,\boldsymbol{v}_t^{\mathrm{fwd}}\Big)\cdot\nabla_\theta\boldsymbol{v}_t^{\theta}\Big]. \tag{145}$$

Therefore, up to non-gradient-producing constants, Eq. (144) is equivalent to

$$\mathcal{L}(\theta) = \mathbb{E}\left[\beta^2\left\|\boldsymbol{v}_t^{\theta} - \left(\boldsymbol{v}_t^{\theta^\dagger} + \frac{c}{\beta}(\boldsymbol{v}_t^{\mathrm{fwd}} - \boldsymbol{v}_t^{\theta^\dagger})\right)\right\|^2\right] \tag{146}$$

Note that

$$\|y - ((1-\lambda)a + \lambda b)\|^2 = \lambda\|y-b\|^2 + (1-\lambda)\|y-a\|^2 + C \tag{147}$$

where C is a constant independent of y. Applying the identity to Eq. (146) with $\lambda = \frac{c}{\beta}$, we see that up to θ-independent constants,

$$\mathcal{L}_{\mathrm{NFT}}(\theta) \;\equiv\; \mathbb{E}\Big[\beta c\big\|\boldsymbol{v}_t^{\theta} - \boldsymbol{v}_t^{\mathrm{fwd}}\big\|^2 + \beta(\beta-c)\big\|\boldsymbol{v}_t^{\theta} - \boldsymbol{v}_t^{\theta^\dagger}\big\|^2\Big], \tag{148}$$

where the first term is an *old-policy (trust-region) anchor* and the second term is a *reward weighted regression* toward the forward target $\boldsymbol{v}_t^{\mathrm{fwd}} = \boldsymbol{\epsilon} - \boldsymbol{x}_0$.

Connection to Reward-weighted MLE. Eq. (148) shows that DiffusionNFT consists of a reward-weighted regression term toward the forward target and an old-policy anchor. In parallel, Soft RL admits a reward-weighted MLE formulation, whose objective reduces to a reward-weighted denoising regression loss. This suggests viewing RWR as a practical approximation to reward-weighted MLE [41]. From this perspective, DiffusionNFT can be understood as a practical approximation to Soft RL, hence closely related to RSM.

H.4 AWM

Advantage Weighted Matching (AWM) [49] also connects diffusion alignment to score matching. However, its objective is fundamentally different from ours. AWM proposes a new advantage-weighted matching objective that brings RL post-training closer to the pretraining loss. By contrast, our goal is not to reinterpret RL fine-tuning as pretraining, but to show that a broad class of *existing* reward-based fine-tuning methods already admit a common Reward Score Matching (RSM) form.

This distinction is central. AWM introduces a new training direction, whereas our contribution is a unifying analysis of prior methods. In particular, we show that methods derived from seemingly different perspectives reduce to a shared RSM structure, and that their primary differences are algorithmic—most notably, how value guidance is estimated and incorporated. To the best of our knowledge, this is the first work to provide such a unification of existing reward-based fine-tuning methods. Accordingly, AWM is best understood as an orthogonal and complementary direction, not a precursor to the perspective developed in this paper.

I Qualitative Results

This section presents qualitative comparisons between baseline methods and our improved variants from the **Validation Across Settings** experiments (Section 5.2). The baselines—Residual ∇-DB, VGG-Flow, PCPO, and TempFlow-GRPO—correspond to Figs. 12–15, respectively. Additional experimental details are provided in Appendix E.2 and summarized in Table 4.

Examples are not cherry-picked to highlight severe reward hacking in the baselines, nor do we claim that our methods exhibit emergent safeguards against reward hacking. Instead, these results serve to illustrate that the observed reward improvements are not achieved through more aggressive exploitation of the reward function. For Fig. 15, we report only our method's outputs, as the corresponding baseline checkpoints trained on GenEval are not publicly available and were not rerun.