
Greed is Good: A Unifying Perspective on Guided Generation

Zander W. Blasingame
Clarkson University
blasinzw@clarkson.edu

Chen Liu
Clarkson University
cliu@clarkson.edu

Abstract

Training-free guided generation is a widely used and powerful technique that allows the end user to exert further control over the generative process of flow/diffusion models. Generally speaking, two families of techniques have emerged for solving this problem for *gradient-based guidance*: namely, *posterior guidance* (*i.e.*, guidance by projecting the current sample to the target distribution via the target prediction model) and *end-to-end guidance* (*i.e.*, guidance by performing backpropagation throughout the entire ODE solve). In this work, we show that these two seemingly separate families can actually be *unified* by looking at the posterior guidance as a *greedy strategy* of *end-to-end guidance*. We explore the theoretical connections between these two families and provide an in-depth theoretical understanding of these two techniques relative to the *continuous ideal gradients*. Motivated by this analysis, we then show a method for *interpolating* between these two families enabling a trade-off between compute and accuracy of the guidance gradients. We then validate this work on several inverse image problems and property-guided molecular generation.

1 Introduction

Guided generation greatly extends the utility of state-of-the-art generative models by allowing the end user to exert greater control over the generative process, ultimately making the tool more useful in a wide variety of applications ranging from conditional generation, editing of samples, inverse problems &c. We focus particularly on a subset of neural differential equations that model *affine probability paths*, in other words, diffusion and flow models due to their widespread adoption in a large variety of practical tasks. *E.g.*, audio (H. Liu et al. 2023; Schneider et al. 2024), images (Rombach et al. 2022; Black Forest Labs 2024), biometrics (Blasingame and C. Liu 2024c), molecules (Hoogeboom et al. 2022; Ben-Hamu et al. 2024), proteins (Watson et al. 2023; Skreta et al. 2025), &c.

We can divide the guided generation techniques into two broad categories: conditional training and training-free methods. The former of these two requires the training of the underlying diffusion/flow model on additional conditional information, either as a part of the training or at a later time as additional fine-tuning (Ho and Salimans 2021; J. Song, Meng, and Ermon 2021; Hu et al. 2022). The latter category instead makes use of some known guidance function defined on the data distribution and incorporates this information back to the model to influence the generative process. These training-free techniques can be further broken down into two sub-categories, *i.e.*, posterior and end-to-end guidance. The former class of techniques uses a simple estimation of the posterior distribution that can be easily found in diffusion models (Chung, J. Kim, et al. 2023) and *some* flow models (*cf.* Lipman, Havasi, et al. 2024, Section 4.8). This simple posterior estimate can then be fed into a guidance function to construct a gradient w.r.t. to the current timestep. We refer to this category as *posterior guidance* as they use this posterior estimate to perform the guidance process. This can then be used to update the ODE solve as a form of classifier guidance (Chung, J. Kim, et al. 2023; Yu et al.

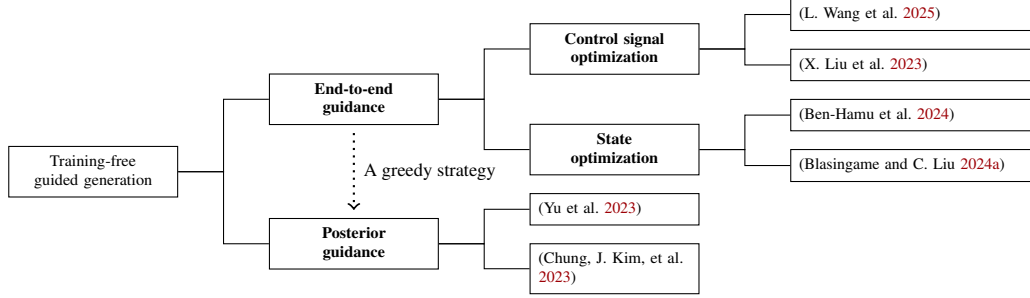


Figure 1: The greedy perspective as a unification of separate families in the taxonomy of training-free guided generation. We provide a more detailed version of this in Figure 5.

2023). The latter class of techniques, in contrast, performs backpropagation throughout the entire sampling process of the flow/diffusion model (Ben-Hamu et al. 2024; Blasingame and C. Liu 2024a). We refer to this category as *end-to-end guidance* as it performs backpropagation throughout the *entire* sampling trajectory.

The aim of this work is to bring these two seemingly disparate family of techniques together into a *single unified view*.

Our key insight is that we can *bridge* between techniques that use posterior sampling and techniques that use end-to-end optimization for guidance by viewing the former as a *greedy strategy* on the latter.

Contributions. In light of this insight, we compare several state-of-the-art techniques from this perspective, showing how this perspective yields a unified and flexible framework for viewing guided generation with flow/diffusion models. We perform a detailed analysis of this greedy strategy, showing that it is not only a unifying view, but that it actually makes *good* decisions under certain scenarios. We then show a perspective which allows one to move between these two classes of guided generation techniques, opening up an exciting and novel design space. Lastly, we conduct some numerical experiments on inverse image problems and molecule generation.

2 Preliminaries

Flow models (Lipman, R. T. Q. Chen, et al. 2023) are a highly popular class of generative models that model the generative process as a neural *ordinary differential equation* (ODE) (R. T. Chen et al. 2018). Consider two $\mathbb{R}^d$ -valued random variables: $\mathbf{X}_0 \sim p(\mathbf{x})$ and $\mathbf{X}_1 \sim q(\mathbf{x})$, denoting the *source* (noise) and *target* (data) distributions, respectively. Then consider a time-dependent vector field $\mathbf{u} \in \mathcal{C}^{1,r}([0, 1] \times \mathbb{R}^d; \mathbb{R}^d)$ ¹ with $r \geq 1$ which determines a time-dependent flow $\Phi_t \in \mathcal{C}^{1,r}([0, 1] \times \mathbb{R}^d; \mathbb{R}^d)$ which satisfies the ODE

$$\Phi_0(\mathbf{x}) = \mathbf{x}, \quad \frac{d}{dt}\Phi_t(\mathbf{x}) = \mathbf{u}(t, \Phi_t(\mathbf{x})). \quad (1)$$

This is known as a $\mathcal{C}^r$ -flow and this flow is diffeomorphism in its second argument for all $t \in [0, 1]$. For notational simplicity let $\mathbf{u}_t(\mathbf{x}) \mapsto \mathbf{u}(t, \mathbf{x})$. A special case of flow models are known as *affine probability paths* and are defined as $\mathbf{X}_t = \alpha_t \mathbf{X}_0 + \sigma_t \mathbf{X}_1$ with schedule (α_t, σ_t) . We provide more details on flow models in Appendix B.1.²

¹For notational simplicity, we let $\mathcal{C}^{k_1, k_2, \dots, k_n}(X_1 \times X_2 \times \dots \times X_n; Y)$ denote the set of continuous functions that are k_i -times differentiable in the i -th argument mapping from $(X_1 \times X_2 \times \dots \times X_n)$ to Y , if Y is omitted, then $Y = \mathbb{R}$.

²Without loss of generality we consider flow models which subsume the ODE formulation of diffusion models.

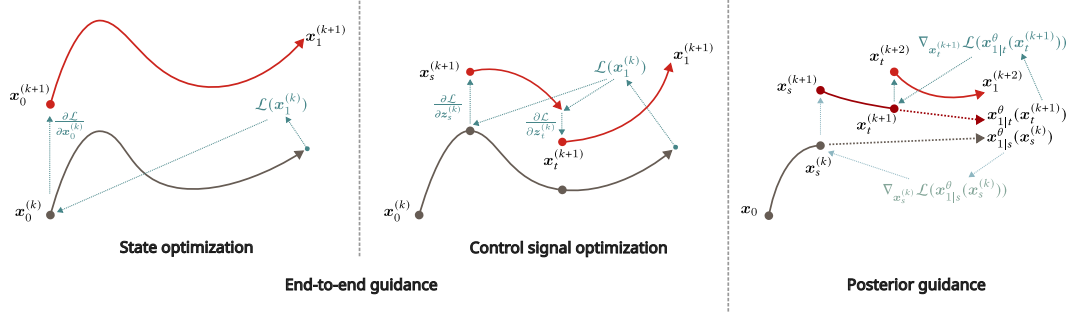


Figure 2: Visual comparison of different training-free guided generation techniques.

3 An overview of training-free guidance with gradients

We explore techniques for solving *training-free* guidance problems—this is in contrast with techniques like classifier (Dhariwal and Nichol 2021; Y. Song, Sohl-Dickstein, et al. 2021) and classifier-free (Ho and Salimans 2021) guidance—which use some off-the-shelf guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$ defined on the output of the flow model. *I.e.*, we wish to optimize the ODE solve such that the output x_1 minimizes $\mathcal{L}$. Suppose we have numerical scheme (Euler, RK4, DPM-Solver, &c.) denoted

$$\begin{aligned} \Phi : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^d \times \mathcal{C}(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d) &\rightarrow \mathbb{R}^d, \\ \Phi(t_n, t_{n+1}, x_n, u_t^\theta) &\mapsto x_{n+1}. \end{aligned} \quad (2)$$

For simplicity we will omit the explicit dependency of the numerical scheme on u_t^θ and assume it implicitly; likewise, let $\Phi_h(t_n, \cdot, \cdot) \mapsto \Phi(t_n, t_{n+1}, \cdot, \cdot)$ where $h = t_{n+1} - t_n$. We write this objective more formally below in Equation (3).

Problem statement. Given some $t_1 \in [0, 1)$ and step size regime $\{t_1 < t_2 < \dots < t_N = 1\}$ solve:

$$\begin{aligned} &\text{Find a sequence } \{x_n\}_{n=1}^N \text{ which minimizes } \mathcal{L}(x_N), \\ &\text{subject to } x_{n+1} = \Phi(t_{n+1}, t_n, x_n). \end{aligned} \quad (3)$$

Next, we will detail two popular families of techniques for solving the problem mentioned above. We illustrate the relationships between these different families in Figure 1, a taxonomy of training-free guidance methods. We note that these two seemingly separate branches can be unified back into a single branch, by the viewing posterior guidance techniques as a greedy strategy of the later. Likewise, we provide a visual overview of the guidance mechanisms in Figure 2.

3.1 Posterior guidance

A popular technique for *training-free* guidance is what we will term *posterior guidance* (Chung, J. Kim, et al. 2023; Yu et al. 2023). The key idea behind this strategy is to use the parameterized target prediction model $x_{1|t}^\theta(x)$, *i.e.*, the expected value of the posterior distribution given $X_t = x$, to provide a guidance gradient of the form $\nabla_x \mathcal{L}(x_{1|t}^\theta(x))$ for some guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$. For literature working with score-based generative models (Y. Song, Sohl-Dickstein, et al. 2021), this interpretation arose from the famous Tweedie’s formula (Stein 1981; Efron and N. R. Zhang 2011). Thus, for each x_n in the ODE solve, we add guidance to it in the form of posterior guidance gradient.

3.2 End-to-end optimization for guidance

Another popular class of techniques is what we will term *end-to-end guidance* (Ben-Hamu et al. 2024; Blasingame and C. Liu 2024a), *i.e.*, techniques which perform guidance by optimizing the initial condition x_0 w.r.t. the guidance function $\mathcal{L}$; such techniques require performing backpropagation through a neural ODE. Fittingly, we will import notations and terminology from the study of *neural differential equations* (Kidger 2022) to discuss these techniques. The first technique for performing this kind of guidance is known as *discretize-then-optimize* (DTO) where the numerical

scheme (cf. Equation (2)) is part of the computation graph of the model reverse-mode automatic differentiation (Linnainmaa 1976) is applied, *i.e.*, *vanilla backpropagation*. The memory cost of such techniques, however, is $\mathcal{O}(n)$, prompting researchers to explore the second method known as *optimize-then-discretize* (OTD) which instead solves *another* ODE in *reverse-time* which models the continuous-time dynamics of reverse-mode differentiation, this is called the *continuous adjoint method* (R. T. Chen et al. 2018; cf. Kidger 2022, Section 5.1.2).

Given a flow model $\mathbf{u}_\theta \in \mathcal{C}^{1,1}([0, 1] \times \mathbb{R}^d; \mathbb{R}^d)$ that is Lipschitz continuous in its second argument and the solution $\mathbf{x} : [0, 1] \rightarrow \mathbb{R}^d, \mathbf{x}_t \mapsto \mathbf{x}(t)$, let $\mathbf{a}_\mathbf{x} := \partial \mathcal{L} / \partial \mathbf{x}_t$ denote the *adjoint state*. Then $\mathbf{a}_\mathbf{x}(t)$ can be found by solving the continuous adjoint equation:

$$\mathbf{a}_\mathbf{x}(1) = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_1}, \quad \frac{d\mathbf{a}_\mathbf{x}}{dt}(t) = -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \mathbf{u}_t^\theta}{\partial \mathbf{x}}(\mathbf{x}_t). \quad (4)$$

N.B., this technique was first proposed by Pontryagin et al. (1963) and popularized for neural differential equations by R. T. Chen et al. (2018). This approach has a constant memory cost $\mathcal{O}(1)$; however, this comes with the cost of several drawbacks related to the numerical scheme. While these issues are not particularly relevant to our theoretical analyses, we note them in Appendix E for the ML practitioner.

4 A greedy perspective on guidance

Now returning back to our problem statement from Equation (3), the end-to-end guidance techniques amount to optimizing the initial condition $\mathbf{x}_0$ in light of the entire solution trajectory admitted by the numerical scheme. A natural question we consider for problems of this form is that rather than finding the full sequence $\{\mathbf{x}_n\}$, can we make use of local information instead? *I.e.*,

Key insight 1. Rather than solving the full ODE from $\mathbf{x}_t$, what if we greedily took a locally optimal step at each $\mathbf{x}_t$ instead?

Formally, we define a greedy strategy is the following augmentation to the numerical scheme from Equation (2) as

$$\mathbf{x}_n^\mathcal{G} = \mathcal{G}(t_n, \mathbf{x}_n, \mathbf{u}_{t_n}^\theta), \quad (5)$$

$$\mathbf{x}_{n+1} = \Phi(t_n, t_{n+1}, \mathbf{x}_n^\mathcal{G}), \quad (6)$$

where $\mathcal{G}$ is the *greedy action* which makes its decision from only information available at time t_n .

Now in particular we are interested in a specific greedy action, *i.e.*, posterior guidance. We define this greedy action as the solution to the following iterative process with initial value $\mathbf{x}_n^{(0)} = \mathbf{x}_n$ which solves

$$\mathbf{x}_n^{(k+1)} = \mathbf{x}_n^{(k)} - \eta \nabla \mathcal{L} \left(\mathbf{x}_{1|t_n}^\theta(\mathbf{x}_n^{(k)}) \right), \quad (7)$$

for some sufficient number $k > 0$ and learning rate $\eta > 0$.

By construction this greedy action is the popular strategy of posterior guidance. The rest of this section is then devoted to exploring the connections between this greedy action and end-to-end guidance schemes. More, succinctly we state our insight below:

Key insight 2. Posterior guidance can be viewed as Euler schemes within the DTO or OTD backpropagation schemes.

To make our analysis simpler, let us write the flow from s to t in terms of the target prediction model. The flow from time s to time t can then be expressed as the integral of the right-hand side of Equation (19) over time. Thus, the flow is now expressed as a semi-linear integral equation with linear term $a_t \mathbf{x}$ and non-linear term $b_t \mathbf{x}_{1|t}^\theta(\mathbf{x})$. Due to this semi-linear structure, we apply the same technique of *exponential integrators* (Hochbruck and Ostermann 2010) that has been successfully used to simplify numerical solvers for diffusion models (Lu et al. 2022a; Q. Zhang and Y. Chen 2023; Gonzalez et al. 2024). *N.B.*, the full derivations and proofs for this section can be found in Appendix B.

Let $\gamma_t := \alpha_t / \sigma_t$ denote the signal-to-noise ratio (SNR), then γ_t is a monotonically increasing sequence in t , due to the properties of (α_t, σ_t) (cf. Equation (17)) and thus has an inverse t_γ such that $t_\gamma(\gamma(t)) = t$. With abuse of notation, we let $\mathbf{x}_\gamma := \mathbf{x}_{t_\gamma(\gamma)}$ and $\mathbf{x}_{1|\gamma}^\theta(\cdot) = \mathbf{x}_{1|t_\gamma(\gamma)}^\theta(\cdot)$. As such, we can rewrite the solution to the flow model in terms of γ by making use of exponential integrators, which we show in Proposition 4.1 with the full proof provided in Appendix B.3.

Proposition 4.1 (Exact solution of affine probability paths). *Given an initial value of $\mathbf{x}_s$ at time $s \in [0, 1]$ the solution $\mathbf{x}_t$ at time $t \in [0, 1]$ of an ODE governed by the vector field in Equation (18) is:*

$$\mathbf{x}_t = \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \mathbf{x}_{1|\gamma}^\theta(\mathbf{x}_\gamma) d\gamma. \quad (8)$$

Remark 4.1. This result bears some similarity to Lu et al. (2022b, Proposition 5.1); however, they integrate w.r.t. the log-SNR; their result can be recovered, *mutatis mutandis*, with the identity $\lambda_t = \log \gamma_t$.

4.1 Greedy guidance as an Euler scheme

Now equipped with this simplified form, we begin to draw connections between end-to-end guidance and our greedy strategy. In Proposition 4.2 we show that the greedy action in Equation (7) can be interpreted as backpropagation via a DTO scheme with an Euler step of size $h = \gamma_1 - \gamma_t$.

Proposition 4.2 (Greedy as an explicit Euler scheme within DTO). *For some trajectory state $\mathbf{x}_t$ at time t , the greedy gradient given by $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{1|t}^\theta(\mathbf{x}))$ is the DTO scheme with an explicit Euler discretization with step size $h = \gamma_1 - \gamma_t$.*

Now we examine greedy action from the perspective of an OTD scheme. In Proposition 4.3 we show that a greedy strategy can be viewed as the first iteration of a fixed-point method of an implicit Euler discretization of the continuous adjoint equations.

Proposition 4.3 (Greedy as an implicit Euler scheme within OTD). *For some trajectory state $\mathbf{x}_t$ at time t , the greedy gradient given by $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{1|t}^\theta(\mathbf{x}_t))$ is an implicit Euler discretization of the continuous adjoint equations for the true gradients with step size $h = \gamma_1 - \gamma_t$.*

Proof sketch. First, we use the technique of exponential integrators to simplify the continuous adjoint equations. Then we perform a first-order Taylor expansion around γ_t , which is equivalent to an implicit Euler scheme, as we calculate the gradient flow from 1 to t . The full proof is provided in Appendix B.5. $\square$

5 Is greed good?

A natural question to ask in light of this discussion on taking this greedy action is why even bother backpropagating through the ODE solve at all for guidance? After all, we could simply run the optimization process directly in the data space (cf. Equation (7)). So why perform end-to-end guidance or this greedy action at all? *N.B.*, the full derivations and proofs for this section may be found in Appendix C.

We begin by examining the structure of the gradient $\nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x}))$. By the chain rule we observe the following:³

$$\nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x})) = \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x}_1)). \quad (9)$$

The question then is what is the behavior of $\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})$? We answer this in Theorem 5.1 below, providing an integral equation for $\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x})$.

³Let $\nabla_{\mathbf{x}_1}$ be shorthand for the gradient w.r.t. the output $\Phi_{t,1}^\theta(\mathbf{x})$.

Theorem 5.1 (Jacobian matrices of affine Gaussian probability paths). *For the standard affine Gaussian probability path with flow model $\Phi_{s,t}^\theta(\mathbf{x})$, the Jacobian matrix $\nabla_{\mathbf{x}}\Phi_{s,t}^\theta(\mathbf{x})$ as function of $\mathbf{x}$ is given as the solution to*

$$\nabla_{\mathbf{x}}\Phi_{s,t}^\theta(\mathbf{x}) = \frac{\sigma_t}{\sigma_s}\mathbf{I} + \sigma_t \int_s^t \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \text{Var}_{1|u}(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}}\Phi_{s,u}^\theta(\mathbf{x}) \, du, \quad (10)$$

where

$$\text{Var}_{1|t}(\mathbf{x}) = \mathbb{E}_{p_{1|t}(\mathbf{x}_1|\mathbf{x})} \left[(\mathbf{x}_1 - \mathbf{x}_{1|t}^\theta(\mathbf{x}))(\mathbf{x}_1 - \mathbf{x}_{1|t}^\theta(\mathbf{x}))^\top \right]. \quad (11)$$

Remark 5.1. From Theorem 5.1 we observe the Jacobian-vector product $\nabla_{\mathbf{x}}\Phi_{s,t}^\theta(\mathbf{x})^\top \mathbf{v}$ corresponds to an integral of covariance projections applied to $\mathbf{v}$.⁴

Thus, we see that the continuous-time backpropagation process through the flow model is a projection of the loss by a covariance matrix into the directions of highest variance, *i.e.*, the guidance encourages the state to evolve within states on the data manifold. We elaborate on this more in Appendix C.2. While this is a nice observation we cannot solve such an integral in practice. What about our greedy strategy, how does it impact the loss function?

5.1 Dynamics of gradient guidance

We now consider how the output of the flow model will change under greedy guidance. In particular, we are interested in how $\Phi_{t,1}^\theta(\mathbf{x})$ changes under the following gradient step

$$\mathbf{x}' = \mathbf{x} - \eta \nabla_{\mathbf{x}} \mathcal{L} \left(\mathbf{x}_{1|t}^\theta(\mathbf{x}) \right). \quad (12)$$

To do this, we make use of the Gateaux differential (Gateaux 1913) which allows us to define the differential that describes how the output of the flow model $\mathbf{x}_1$ evolves with changes to $\mathbf{x}$ at time t . We present the result to this question in Proposition 5.2 below.

Proposition 5.2 (Dynamics of greedy gradient guidance). *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\mathbf{x}$ at some time $t \in [0, 1]$ in the direction of the gradient $\nabla_{\mathbf{x}} \mathcal{L} \left(\mathbf{x}_{1|t}^\theta(\mathbf{x}) \right)$ is given by*

$$\delta_{\mathbf{x}}^{\mathcal{G}} \Phi_{t,1}^\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1). \quad (13)$$

Remark 5.2. Recall that from Theorem 5.1 and (Ben-Hamu et al. 2024, Proposition 4.1) we know that both $\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})$ and $\nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x})$ consist of covariance matrices, thus the dynamics of greedy gradient guidance are governed by this covariance projection of the loss.

Next, we ask what is the difference between the *idealized* gradient $\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})$ and the greedy gradient $\nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x})$? Intuitively, we find that it is bound by the local truncation error, *i.e.*, $\mathcal{O}(h^2)$ which we show below.

Theorem 5.3 (Dynamics of gradient vs greedy guidance). *The difference between the dynamics of gradient guidance in Proposition C.4 and greedy gradient guidance in Proposition 5.2 for a point $\mathbf{x}$ at time t with guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$ is bounded by $\mathcal{O}(h^2)$ where $h := \gamma_1 - \gamma_t$, *i.e.*,*

$$\left\| \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x}) \right\| = \mathcal{O}(h^2). \quad (14)$$

An important question is whether a greedy strategy makes *good* decisions at each timestep. *I.e.*, if we make a good decision at time t , does that ensure that an optimal solution was made in the sense of $\Phi_{1|t}^\theta(\mathbf{x}_t)$. A natural way to examine this question is to consider whether convergence in the local case implies convergence of the whole solution trajectory. We find that up to a bound dependent on the

⁴Readers familiar with the work of Ben-Hamu et al. (2024) may notice some similarities between our result Theorem 5.1 and Ben-Hamu et al. (2024, Theorem 4.2). We discuss this more in Remark C.3.

step size, convergence in the greedy solution implies convergence in the flow, which we state more formally in Theorem 5.4.

Theorem 5.4 (Greedy convergence). *For affine probability paths, if there exists a sequence of states $\mathbf{x}_t^{(n)}$ at time t such that it converges to the locally optimal solution $\mathbf{x}_{1|t}^\theta(\mathbf{x}_t^{(n)}) \rightarrow \mathbf{x}_1^*$. Then the solution, $\Phi_{1|t}^\theta(\mathbf{x}_t^{(n)})$, converges to a neighborhood of size $\mathcal{O}(h^2)$ centered at $\mathbf{x}_1^*$.*

6 Beyond Euler

Motivated by this connection between the powerful, but expensive, end-to-end guidance techniques and posterior guidance techniques, we ask is there a middle-ground between them? A natural extension would be to consider something beyond the Euler scheme from the previous section, e.g., applying the midpoint method or two Euler steps. To motivate this discussion more rigorously we present Theorem 6.1, which shows that for any explicit single-step Runge-Kutta solver, the error between the *ideal* gradient and this estimated gradient is on the order of the local truncation error of the underlying numerical solver.

Theorem 6.1 (Truncation error of single-step gradients). *Let Φ be an explicit Runge-Kutta solver of order $\alpha > 0$ of a flow model with flow $\Phi_{s,t}^\theta(\mathbf{x})$. Then for any $t \in [0, 1]$,*

$$\|\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \Phi_{t,1}(\mathbf{x})\| = \mathcal{O}(h^{\alpha+1}), \quad (15)$$

where $h = 1 - t$.

Key insight 3. We can use a higher-order solver to move between posterior and end-to-end guidance exchanging compute and gradient accuracy.

This theoretical tool enables us to move between posterior and full end-to-end guidance choosing whichever point between compute and accuracy happens to be most suitable, hopefully opening a larger design space for solving interesting problems. Additional discussions and the full derivations are found in Appendix D.

7 Experiments

Motivated by the theoretical connections from the previous sections we apply the greedy posterior strategy (Euler) to several problems using flow/diffusion models, as well as several methods lying in the in between space of end-to-end guidance and posterior guidance, namely, a single-step midpoint scheme and 2-step Euler scheme.

7.1 Inverse problems for images

A common application of posterior guidance has been in solving inverse problems (Y. Song, Sohl-Dickstein, et al. 2021; Chung, Sim, and Ye 2022) (cf. Appendix H). As such, we explore several inverse problems in the image domain. In particular, we explore a set of inverse image problems on a subset of 100 images from the FFHQ (Karras, Laine, and Aila 2019) 256×256 dataset. We make use of the pre-trained diffusion model from Chung, J. Kim, et al. (2023) trained on the FFHQ dataset.

Inverse problems and metrics. Following (B. Zhang et al. 2025) we conduct experiments on the following linear tasks: super resolution, Gaussian deblurring, motion deblurring, inpainting (with a box mask), and inpainting (with a 70% random mask); along with three non-linear problems: phase retrieval, high dynamic range (HDR) reconstruction, and non-linear deblurring. We use the standard evaluation metrics of *peak signal-to-noise-ratio* (PSNR), *structural similarity index measure* (SSIM), *Learned Perceptual Image Patch Similarity* (LPIPS) (R. Zhang et al. 2018), and *Fréchet Inception Distance* (FID) (Heusel et al. 2017). We solve the probability flow ODE with the midpoint scheme and 20 discretization steps; further configuration details are reported in Appendix I.1.



Figure 3: Qualitative visualization of using posterior guidance to solve an inverse problem on the task of inpainting with a 70% random mask. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.

Table 1: A snapshot of the quantitative results for solving inverse image problems on FFHQ. We report the mean performance (PSNR, SSIM, and LPIPS) across 100 validation images along with the FID. All tasks are using a noisy measurement with noise level $\beta_y = 0.05$. The full results are found in Table 6.

Task	Method	PSNR ($\uparrow$)	SSIM ($\uparrow$)	LPIPS ($\downarrow$)	FID ($\downarrow$)
Inpaint (random)	Greedy (Euler)	30.87	0.823	0.141	40.73
	Greedy (midpoint)	31.03	0.816	0.139	38.80
	Greedy (2-step Euler)	30.80	0.811	0.144	39.23
	DAPS	31.12	0.844	0.098	32.17
	DPS	25.46	0.823	0.203	69.20
Gaussian deblurring	Greedy (Euler)	28.01	0.766	0.182	57.04
	Greedy (midpoint)	28.36	0.776	0.185	58.55
	Greedy (2-step Euler)	28.18	0.774	0.181	57.18
	DAPS	29.19	0.817	0.165	53.33
	DPS	25.87	0.764	0.219	79.75

Results. We present some qualitative results on reconstructing images from a random mask in Section 7. Quantitatively, we present a snapshot of our full results (*cf.* Table 6) on the inpainting with random mask and Gaussian deblurring tasks. For reference we include the standard DPS (Chung, J. Kim, et al. 2023) and the recent state-of-the-art DAPS (B. Zhang et al. 2025). We observe that the posterior guidance strategy works well performing closer to DAPS than DPS. Interestingly, on these tasks the extra compute and smaller truncation error of the midpoint and 2-step Euler did not lead to any noticeable performance gains. We report further results in Appendix J.2 along with additional analysis and discussion.

Ablations on discretization steps. As we discussed in Section 6 we can improve performance by taking more step sizes. We perform a more involved ablation of this design axis on the *high-dynamic range* (HRD) reconstruction experiment detailed in Table 2. Additionally, we also report the results from the RED-diff (Mardani et al. 2024) algorithm. Following B. Zhang et al. (2025) for the *non-linear* inverse problem of HDR reconstruction we perform 4 runs per algorithm and report the mean and standard deviation. We notice that the Greedy (5-step Euler) performs very well beating the SOTA DAPS algorithm, even the 2-step and 3-step perform variants as well or better than DAPS on this problem. Increasing the number of discretization steps leads to better performance (*cf.* Theorem 6.1). Interestingly, the standard deviation decreases as well with the results becoming more consistent. We also compared to a full DTO run with vary step sizes, *i.e.*, end-to-end optimization with a vary number of steps used in calculating the gradient (the full 20 are used for the final sampling). We do observe a similar trend of increasing performance as we increase the number of steps, however, it far under-performs the greedy strategy for a similar compute budget.

Table 2: Further ablations on the number of discretization steps on the non-linear HDR inverse problem.

Method	PSNR ($\uparrow$)	SSIM ($\uparrow$)	LPIPS ($\downarrow$)	FID ($\downarrow$)
DAPS	27.12 $\pm$ 3.53	0.752 $\pm$ 0.041	0.162 $\pm$ 0.072	42.97
DPS	22.73 $\pm$ 6.07	0.591 $\pm$ 0.141	0.264 $\pm$ 0.156	112.82
RED-diff	22.16 $\pm$ 3.41	0.512 $\pm$ 0.083	0.258 $\pm$ 0.089	108.32
Greedy (Euler)	25.07 $\pm$ 4.25	0.776 $\pm$ 0.126	0.173 $\pm$ 0.070	43.25
Greedy (2-step Euler)	26.32 $\pm$ 4.34	0.802 $\pm$ 0.111	0.173 $\pm$ 0.065	38.64
Greedy (3-step Euler)	27.17 $\pm$ 4.21	0.820 $\pm$ 0.096	0.154 $\pm$ 0.062	36.07
Greedy (4-step Euler)	27.89 $\pm$ 4.10	0.828 $\pm$ 0.092	0.151 $\pm$ 0.061	36.94
Greedy (5-step Euler)	28.27 $\pm$ 4.01	0.831 $\pm$ 0.088	0.149 $\pm$ 0.059	35.35
DTO (1-step)	13.16 $\pm$ 1.15	0.372 $\pm$ 0.083	0.521 $\pm$ 0.059	108.39
DTO (2-step)	14.91 $\pm$ 1.23	0.372 $\pm$ 0.080	0.483 $\pm$ 0.061	98.93
DTO (4-step)	16.37 $\pm$ 1.38	0.455 $\pm$ 0.082	0.457 $\pm$ 0.066	93.52
DTO (8-step)	16.37 $\pm$ 1.38	0.455 $\pm$ 0.082	0.457 $\pm$ 0.066	93.52

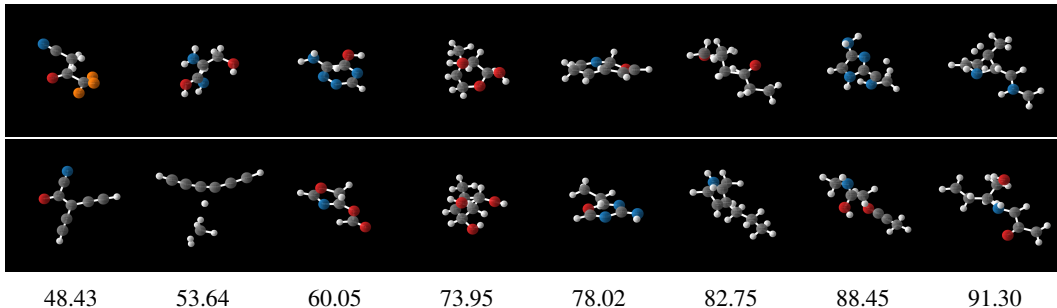


Figure 4: Qualitative visualization of controlled generated molecules for various polarizability (α) levels. Top row is generated using end-to-end guidance with a DTO scheme and the bottom row is generated using greedy guidance.

7.2 Molecule generation for QM9

We also illustrate the core ideas with some experiments in controllable molecule generation on the QM9 dataset (Ruddigkeit et al. 2012), a popular molecular dataset containing small molecules with up to 29 atoms. Following Hoogetboom et al. (2022) and Ben-Hamu et al. (2024), we perform the conditional generation of molecules with specified quantum chemical property values. In particular, we target the following properties: polarizability α , orbital energies ϵ_{HOMO} , ϵ_{LUMO} and their gap $\Delta\epsilon$, dipole moment μ , and heat capacity C_v . The property classifiers were trained following the methodology outlined in Hoogetboom et al. (2022). The underlying flow model is an unconditional equivariant flow matching model with *conditional optimal transport* path (Lipman, Havasi, et al. 2024, Section 4.7; cf. Tong, Malkin, et al. 2023; Tong, FATRAS, et al. 2024), i.e., the EquiFM (Y. Song, Gong, et al. 2023) model. We solve the ODE with Euler’s method and 50 discretization steps; further configuration details are reported in Appendix I.1. Further details are provided in Appendix I.2.

Metrics. To evaluate the guided generation we calculate the *mean absolute error* (MAE) between the predicted property value of the generated molecule by the property classifier and the target property value (Satorras et al. 2021). Additionally in Appendix J.1 we report the quality of the generated molecules by evaluating the atom stability (the percentage of atoms with correct valency) and molecule stability (the percentage of molecules where all atoms are stable).

Results. In Section 7.1 we present a visual comparison between molecules generated targeting different polarizability α values using a DTO end-to-end guidance scheme (essentially D-Flow) and the posterior guidance scheme. Notice that as α increases the compactness of the molecules

Table 3: Quantitative evaluation of conditional molecule generation. The MAE is reported for each molecule property (lower is better).

Property Unit	α Bohr ²	$\Delta\varepsilon$ meV	$\varepsilon_{\text{HOMO}}$ meV	$\varepsilon_{\text{LUMO}}$ meV	μ D	C_v $\frac{\text{cal}}{\text{K}\cdot\text{mol}}$
Greedy (Euler)	11.282	1265	725	1092	1.559	6.469
Greedy (midpoint)	5.313	1196	599	1057	1.417	2.967
Greedy (2-step Euler)	5.667	1205	695	1222	1.491	2.767
Greedy (3-step Euler)	5.098	1152	600	1152	1.384	3.229
Greedy (5-step Euler)	4.177	1083	571	939	1.328	2.332
DTO (1-step)	13.049	989×10^{12}	681	86.512	1.666	15.144
DTO (2-step)	6.113	1359	666	1199	1.533	3.757
DTO (4-step)	6.115	1294	668	1190	1.406	2.829
DTO (8-step)	4.549	1070	608	1078	1.247	2.594
DTO (16-step)	3.454	817	608	939	1.177	2.003
DTO (32-step)	2.912	750	410	666	0.721	1.566
DTO (40-step)	2.384	625	372	556	0.719	1.425
DTO (50-step)	1.404	401	176	373	0.372	0.866
EquiFM	9.525	1494	622	1523	1.628	6.689
Lower bound	0.10	64	39	46	0.043	0.040

generated by a DTO scheme decreases. This trend is less noticeable for the posterior guided samples. We report quantitative results in Table 3. We report the unguided EquiFM generated molecules as an upper bound and include the theoretical lower bounds from Ben-Hamu et al. (2024). It is here that we notice a sharp decrease in performance from using posterior guidance. In particular the greedy (Euler) strategy is highly unstable even performing worse than the unguided model on the α property. The introduction of an additional step in the form of either midpoint or 2-step Euler does seem to improve performance; although the significance varies property to property. We observe that the midpoint method seems to perform slightly better than the 2-step Euler. We performed experiments with Ralston’s third-order method and a fourth-order Runge-Kutta scheme but noticed significant instability in comparison to just taking more steps, we posit this is due to the large step size and that a hybrid scheme like that employed by Moufad et al. (2025) might be a reasonable solution to such problems, but ultimately we leave that question up to future work. Moreover, we observe that increasing the number of steps generally improves performance with greedy (5-step Euler) performing the best among all the greedy guidance strategies. The gradients of DTO (50-step) strategy are the *ideal* and perfect gradients w.r.t. the flow model as the numerical solver takes 50 discretization steps and thus form the *upper bound of performance*. We see that as the DTO strategy incorporates more discretization steps the results converge to the upper bound along this particular design axis; in particular, we notice that the greedy strategy does well in the regime of less discretization steps.

8 Conclusion

In this paper we present a unifying view of two different families of guided generation: end-to-end guidance and posterior guidance from the lens of a greedy algorithm. We present numerous theoretical connections tying these two families together. Our theoretical analysis shows that there might be some reason to believe that such a cheap approximation of the gradient can be reasonable for *certain* tasks. By exploiting the theoretical connections we created, we investigate guidance techniques which lie in between these two families giving rise to an exciting novel design space. We then conduct several experiments on inverse image problems and on controlled molecule generation to illustrate this new design space. We hope that our findings can help future researchers find the optimal spot between computational cost and accuracy of gradients for guidance problems.

References

- Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., and Goldstein, T. (2023). “Universal guidance for diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 843–852 (cit. on p. 26).
- Ben-Hamu, H., Puny, O., Gat, I., Karrer, B., Singer, U., and Lipman, Y. (2024). “D-Flow: Differentiating through Flows for Controlled Generation”. In: *Forty-first International Conference on Machine Learning*. URL: <https://openreview.net/forum?id=SE20BFqj6J> (cit. on pp. 1–3, 6, 9, 10, 26–28, 30, 31, 33, 45).
- Black Forest Labs (2024). *FLUX*. <https://github.com/black-forest-labs/flux> (cit. on p. 1).
- Blasingame and Liu, C. (2024a). “AdjointDEIS: Efficient Gradients for Diffusion Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang. Vol. 37. Curran Associates, Inc., pp. 2449–2483. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf (cit. on pp. 2, 3, 26, 27, 32, 40, 41).
- (2024b). “Greedy-DiM: Greedy Algorithms for Unreasonably Effective Face Morphs”. In: *2024 IEEE International Joint Conference on Biometrics (IJCBI)*, pp. 1–11. DOI: 10.1109/IJCBI62174.2024.10744517 (cit. on p. 41).
- (2024c). “Leveraging diffusion for strong and high quality face morphing attacks”. In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 6.1, pp. 118–131 (cit. on p. 1).
- (2025). “A Reversible Solver for Diffusion SDEs”. In: *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. URL: <https://openreview.net/forum?id=0gEFLVUL6n> (cit. on p. 40).
- Butcher, J. C. (2016). *Numerical methods for ordinary differential equations*. Third Edition. John Wiley & Sons (cit. on p. 40).
- Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (cit. on pp. 2, 4, 39, 41).
- Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). “Diffusion Posterior Sampling for General Noisy Inverse Problems”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023*. The International Conference on Learning Representations (cit. on pp. 1–3, 7, 8, 26, 27, 43–45, 47).
- Chung, H., Sim, B., and Ye, J. C. (2022). “Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12413–12422 (cit. on p. 7).
- Clark, K., Vicol, P., Swersky, K., and Fleet, D. J. (2024). “Directly Fine-Tuning Diffusion Models on Differentiable Rewards”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=1vmSEVL19f> (cit. on pp. 26, 27).
- Daras, G., Chung, H., Lai, C.-H., Mitsufuji, Y., Ye, J. C., Milanfar, P., Dimakis, A. G., and Delbracio, M. (2024). “A survey on diffusion models for inverse problems”. In: *arXiv preprint arXiv:2410.00083* (cit. on p. 26).
- Dhariwal, P. and Nichol, A. (2021). “Diffusion Models Beat GANs on Image Synthesis”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., pp. 8780–8794. URL: <https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf> (cit. on p. 3).
- Domingo-Enrich, C., Drozdal, M., Karrer, B., and Chen, R. T. Q. (2025). “Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control”. In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=xQBRrtQM8u> (cit. on p. 40).
- Dou, Z. and Song, Y. (2024). “Diffusion Posterior Sampling for Linear Inverse Problem Solving: A Filtering Perspective”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=tplXNcHZs1> (cit. on p. 47).
- Dyson, F. J. (1949). “The radiation theories of Tomonaga, Schwinger, and Feynman”. In: *Physical Review* 75.3, p. 486 (cit. on p. 33).
- Efron, B. and Zhang, N. R. (2011). “False discovery rates and copy number variation”. In: *Biometrika* 98.2, pp. 251–271 (cit. on p. 3).

- Frazer, R. A., Duncan, W. J., and Collar, A. R. (1938). *Elementary matrices and some applications to dynamics and differential equations*. Cambridge University Press (cit. on p. 33).
- Friz, P. K. and Victoir, N. B. (2010). *Multidimensional stochastic processes as rough paths: theory and applications*. Vol. 120. Cambridge University Press (cit. on p. 32).
- Gâteaux, R. (1913). “Sur les fonctionnelles continues et les fonctionnelles analytiques”. In: *CR Acad. Sci. Paris* 157.325-327, p. 65 (cit. on p. 6).
- Gonzalez, M., Fernandez Pinto, N., Tran, T., Hajri, H., Masmoudi, N., et al. (2024). “Seeds: Exponential sde solvers for fast high-quality sampling from diffusion models”. In: *Advances in Neural Information Processing Systems* 36 (cit. on pp. 4, 29).
- Griewank, A. (1992). “Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation”. In: *Optimization Methods and software* 1.1, pp. 35–54 (cit. on p. 40).
- Griewank, A. and Walther, A. (2000). “Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation”. In: *ACM Trans. Math. Softw.* 26.1, pp. 19–45. DOI: [10.1145/347837.347846](https://doi.org/10.1145/347837.347846) (cit. on p. 40).
- Grossman, M. and Katz, R. (1972). *Non-Newtonian Calculus: A Self-contained, Elementary Exposition of the Authors’ Investigations...* Non-Newtonian Calculus (cit. on p. 33).
- Harier, E. and Wanner, G. (Feb. 2002). *Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems*. 2nd ed. Springer Series in Computational Mathematics. Berlin, Germany: Springer (cit. on p. 40).
- He, Y., Murata, N., Lai, C.-H., Takida, Y., Uesaka, T., Kim, D., Liao, W.-H., Mitsufuji, Y., Kolter, J. Z., Salakhutdinov, R., and Ermon, S. (2024). “Manifold Preserving Guided Diffusion”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=o3Bx0L0xm1> (cit. on p. 26).
- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf (cit. on p. 7).
- Ho, J., Jain, A., and Abbeel, P. (2020). “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33, pp. 6840–6851 (cit. on p. 28).
- Ho, J. and Salimans, T. (2021). “Classifier-Free Diffusion Guidance”. In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications* (cit. on pp. 1, 3).
- Hochbruck, M. and Ostermann, A. (2010). “Exponential integrators”. In: *Acta Numerica* 19, pp. 209–286 (cit. on p. 4).
- Holderrieth, P., Havasi, M., Yim, J., Shaul, N., Gat, I., Jaakkola, T., Karrer, B., Chen, R. T. Q., and Lipman, Y. (2025). “Generator Matching: Generative modeling with arbitrary Markov processes”. In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=RuP17cJtZo> (cit. on p. 28).
- Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). “Equivariant diffusion for molecule generation in 3d”. In: *International conference on machine learning*. PMLR, pp. 8867–8887 (cit. on pp. 1, 9, 45).
- Hu, E. J., shen, yelong, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=nZeVKeeFYf9> (cit. on p. 1).
- Kadkhodaie, Z. and Simoncelli, E. (2021). “Stochastic Solutions for Linear Inverse Problems using the Prior Implicit in a Denoiser”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., pp. 13242–13254. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/6e28943943dbed3c7f82fc05f269947a-Paper.pdf (cit. on p. 26).
- Karras, T., Laine, S., and Aila, T. (2019). “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405. DOI: [10.1109/CVPR.2019.00453](https://doi.org/10.1109/CVPR.2019.00453) (cit. on p. 7).
- Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). “Elucidating the Design Space of Diffusion-Based Generative Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=k7FuTOWMOc7> (cit. on pp. 44, 45).

- Karunratanakul, K., Preechakul, K., Aksan, E., Beeler, T., Suwajanakorn, S., and Tang, S. (2024). “Optimizing diffusion noise can serve as universal motion priors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1334–1345 (cit. on pp. 26, 27).
- Kawar, B., Elad, M., Ermon, S., and Song, J. (2022). “Denoising Diffusion Restoration Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=kxXvopt9pWK> (cit. on p. 47).
- Kidger, P. (2022). “On Neural Differential Equations”. Available at <https://arxiv.org/abs/2202.02435>. Ph.D. thesis. Oxford University (cit. on pp. 3, 4, 32, 39, 40).
- Kidger, P., Foster, J., Li, X. C., and Lyons, T. (2021). “Efficient and accurate gradients for neural sdes”. In: *Advances in Neural Information Processing Systems* 34, pp. 18747–18761 (cit. on p. 40).
- Kim, S., Ji, W., Deng, S., Ma, Y., and Rackauckas, C. (2021). “Stiff neural ordinary differential equations”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.9 (cit. on p. 40).
- Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). “Variational diffusion models”. In: *Advances in neural information processing systems* 34, pp. 21696–21707 (cit. on p. 28).
- Kirk, D. E. (2004). *Optimal control theory: an introduction*. Courier Corporation (cit. on p. 41).
- Li, X., Kwon, S. M., Alkhouri, I. R., Ravishankar, S., and Qu, Q. (2024). “Decoupled data consistency with diffusion purification for image restoration”. In: *arXiv preprint arXiv:2403.06054* (cit. on p. 47).
- Linnainmaa, S. (June 1976). “Taylor expansion of the accumulated rounding error”. In: *BIT* 16.2, pp. 146–160. ISSN: 0006-3835. DOI: 10.1007/BF01931367. URL: <https://doi.org/10.1007/BF01931367> (cit. on p. 4).
- Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). “Flow Matching for Generative Modeling”. In: *The Eleventh International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=PqvMRDCJT9t> (cit. on pp. 2, 28).
- Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. (2024). “Flow Matching Guide and Code”. In: *arXiv preprint arXiv:2412.06264* (cit. on pp. 1, 9, 28, 44).
- Liu, D. C. and Nocedal, J. (1989). “On the limited memory BFGS method for large scale optimization”. In: *Mathematical programming* 45.1, pp. 503–528 (cit. on p. 45).
- Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D. (2023). “AudioLDM: Text-to-Audio Generation with Latent Diffusion Models”. In: *International Conference on Machine Learning*. PMLR, pp. 21450–21474 (cit. on p. 1).
- Liu, X., Wu, L., Zhang, S., Gong, C., Ping, W., and Liu, Q. (2023). “Flowgrad: Controlling the output of generative odes with gradients”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24335–24344 (cit. on pp. 2, 26, 27, 41).
- Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022a). “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: https://openreview.net/forum?id=2uAaGw1P_V (cit. on pp. 4, 29).
- (2022b). “Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models”. In: *arXiv preprint arXiv:2211.01095* (cit. on p. 5).
- Magnus, W. (1954). “On the exponential solution of differential equations for a linear operator”. In: *Communications on pure and applied mathematics* 7.4, pp. 649–673 (cit. on p. 33).
- Mardani, M., Song, J., Kautz, J., and Vahdat, A. (2024). “A Variational Perspective on Solving Inverse Problems with Diffusion Models”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=1Y04EE3SPB> (cit. on pp. 8, 47).
- Marion, P., Korba, A., Bartlett, P., Blondel, M., Bortoli, V. D., Doucet, A., Llinares-López, F., Paquette, C., and Berthet, Q. (2025). “Implicit Diffusion: Efficient optimization through stochastic sampling”. In: *The 28th International Conference on Artificial Intelligence and Statistics*. URL: <https://openreview.net/forum?id=r5F7Z8s0Qk> (cit. on pp. 26, 27).
- McCallum, S. and Foster, J. (2024). “Efficient, Accurate and Stable Gradients for Neural ODEs”. In: *arXiv preprint arXiv:2410.11648* (cit. on p. 40).
- Moufadi, B., Janati, Y., Bedin, L., Durmus, A. O., Douc, R., Moulines, E., and Olsson, J. (2025). “Variational Diffusion Posterior Sampling with Midpoint Guidance”. In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=6EUtjXAvmj> (cit. on pp. 10, 26, 27, 44, 45, 53).
- Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. (17–23 Jul 2022). “Diffusion Models for Adversarial Purification”. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S.

- Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 16805–16827. URL: <https://proceedings.mlr.press/v162/nie22a.html> (cit. on p. 27).
- Novack, Z., McAuley, J., Berg-Kirkpatrick, T., and Bryan, N. J. (2024). “DITTO: Diffusion Inference-Time T-Optimization for Music Generation”. In: *Forty-first International Conference on Machine Learning*. URL: <https://openreview.net/forum?id=z5Ux2u6t7U> (cit. on pp. 26, 27).
- Onken, D. and Ruthotto, L. (2020). “Discretize-optimize vs. optimize-discretize for time-series regression and continuous normalizing flows”. In: *arXiv preprint arXiv:2005.13420* (cit. on p. 38).
- Pan, J., Liew, J. H., Tan, V., Feng, J., and Yan, H. (2024). “AdjointDPM: Adjoint Sensitivity Method for Gradient Backpropagation of Diffusion Probabilistic Models”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=y331DRBgWI> (cit. on pp. 26, 27, 39).
- Pan, J., Yan, H., Liew, J. H., Feng, J., and Tan, V. Y. (2023). “Towards accurate guided diffusion sampling through symplectic adjoint method”. In: *arXiv preprint arXiv:2312.12030* (cit. on p. 27).
- Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., and Mishechenko, E. F. (1963). “The Mathematical Theory of Optimal Processes.” In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11, pp. 514–515. DOI: <https://doi.org/10.1002/zamm.19630431023>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19630431023>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023> (cit. on pp. 4, 41).
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695 (cit. on p. 1).
- Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). “Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17”. In: *Journal of chemical information and modeling* 52.11, pp. 2864–2875 (cit. on p. 9).
- Sakurai, J. J. and Napolitano, J. (2020). *Modern quantum mechanics*. Cambridge University Press (cit. on p. 33).
- Satorras, V. G., Hoogeboom, E., Fuchs, F. B., Posner, I., and Welling, M. (2021). “E(n) Equivariant Normalizing Flows”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. URL: https://openreview.net/forum?id=N5hQI_RowVA (cit. on p. 9).
- Schneider, F., Kamal, O., Jin, Z., and Schölkopf, B. (2024). “Moûsai: Efficient text-to-music diffusion models”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8050–8068 (cit. on p. 1).
- Skreta, M., Atanackovic, L., Bose, J., Tong, A., and Neklyudov, K. (2025). “The Superposition of Diffusion Models Using the Itô Density Estimator”. In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=2o58Mbqkd2> (cit. on p. 1).
- Song, B., Kwon, S. M., Zhang, Z., Hu, X., Qu, Q., and Shen, L. (2024). “Solving Inverse Problems with Latent Diffusion Models via Hard Data Consistency”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=j8hdRqOUhN> (cit. on p. 44).
- Song, J., Meng, C., and Ermon, S. (2021). “Denoising Diffusion Implicit Models”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=St1giarCHLP> (cit. on p. 1).
- Song, J., Vahdat, A., Mardani, M., and Kautz, J. (2023). “Pseudoinverse-Guided Diffusion Models for Inverse Problems”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=9_gsMA8MRKQ (cit. on p. 26).
- Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (23–29 Jul 2023). “Consistency Models”. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 32211–32252. URL: <https://proceedings.mlr.press/v202/song23a.html> (cit. on p. 45).
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=PxTIG12RRHS> (cit. on pp. 3, 7, 28, 44).
- Song, Y., Gong, J., Xu, M., Cao, Z., Lan, Y., Ermon, S., Zhou, H., and Ma, W.-Y. (2023). “Equivariant Flow Matching with Hybrid Probability Transport for 3D Molecule Generation”. In: *Thirty-seventh*

- Conference on Neural Information Processing Systems. URL: <https://openreview.net/forum?id=hHUZ5V9XFu> (cit. on pp. 9, 45).
- Stein, C. M. (1981). “Estimation of the mean of a multivariate normal distribution”. In: *The annals of Statistics*, pp. 1135–1151 (cit. on pp. 3, 26, 44).
- Stewart, D. E. (2022). *Numerical analysis: A graduate course*. Vol. 258. Springer (cit. on pp. 37, 42).
- Stumm, P. and Walther, A. (2010). “New Algorithms for Optimal Online Checkpointing”. In: *SIAM Journal on Scientific Computing* 32.2, pp. 836–854. DOI: [10.1137/080742439](https://doi.org/10.1137/080742439) (cit. on p. 40).
- Tong, A., FATRAS, K., Malkin, N., Huguët, G., Zhang, Y., Rector-Brooks, J., Wolf, G., and Bengio, Y. (2024). “Improving and generalizing flow-based generative models with minibatch optimal transport”. In: *Transactions on Machine Learning Research*. Expert Certification. ISSN: 2835-8856. URL: <https://openreview.net/forum?id=CD9Snc73AW> (cit. on p. 9).
- Tong, A., Malkin, N., Huguët, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. (2023). “Conditional flow matching: Simulation-free dynamic optimal transport”. In: *arXiv preprint arXiv:2302.00482* 2.3 (cit. on p. 9).
- Wallace, B., Gokul, A., Ermon, S., and Naik, N. (2023). “End-to-end diffusion latent optimization improves classifier guidance”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7280–7290 (cit. on p. 27).
- Wallace, B., Gokul, A., and Naik, N. (2023). “Edict: Exact diffusion inversion via coupled transformations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22532–22541 (cit. on p. 27).
- Wang, F., Yin, H., Dong, Y.-J., Zhu, H., Zhang, C., Zhao, H., Qian, H., and Li, C. (2024). “BELM: Bidirectional Explicit Linear Multi-step Sampler for Exact Inversion in Diffusion Models”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=ccQ4fmwLDb> (cit. on p. 27).
- Wang, L., Cheng, C., Liao, Y., Qu, Y., and Liu, G. (2025). “Training Free Guided Flow-Matching with Optimal Control”. In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=61ss5RA1MM> (cit. on pp. 2, 26, 27, 41, 45).
- Wang, Y., Yu, J., and Zhang, J. (2023). “Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model”. In: *The Eleventh International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=mRieQgMtNTQ> (cit. on pp. 26, 47).
- Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. (2023). “De novo design of protein structure and function with RFdiffusion”. In: *Nature* 620.7976, pp. 1089–1100 (cit. on p. 1).
- Weinberg, S. (1995). *The quantum theory of fields*. Vol. 2. Cambridge university press (cit. on p. 33).
- Yu, J., Wang, Y., Zhao, C., Ghanem, B., and Zhang, J. (2023). “Freedom: Training-free energy-guided conditional diffusion model”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23174–23184 (cit. on pp. 1–3, 26, 27, 45).
- Zhang, B., Chu, W., Berner, J., Meng, C., Anandkumar, A., and Song, Y. (2025). “Improving diffusion inverse problem solving with decoupled noise annealing”. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20895–20905 (cit. on pp. 7, 8, 26, 44, 45, 47).
- Zhang, Q. and Chen, Y. (2023). “Fast Sampling of Diffusion Models with Exponential Integrator”. In: *The Eleventh International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Loek7hfb46P> (cit. on pp. 4, 29).
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595 (cit. on p. 7).
- Zhu, Y., Zhang, K., Liang, J., Cao, J., Wen, B., Timofte, R., and Van Gool, L. (June 2023). “Denoising Diffusion Models for Plug-and-Play Image Restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1219–1229 (cit. on p. 47).

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The theoretical claims introduced in the abstract and introduction are verified in Sections 4 to 6 and in the appendices. Likewise, the experimental claims made are verified in Section 7 and appropriate appendices.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We address the limitations of this work in Appendix K.2.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: The theoretical results from the main paper are proved with full assumptions in Appendices [B](#) to [D](#).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Yes, we provide details on our experimental procedures in Appendix [I](#). Additionally, the random seeds we used are fixed in the configuration files in our code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The source code will be uploaded as a part of the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, the full details for the experiments are provided in Appendix I.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Due to the heavy computational demands in running these experiments we did not report statistical significance which is inline with other works in this space.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide the full details of the compute resources in Appendix I.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We conducted research conforming in every aspect with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Yes, we discuss such impacts in Appendix K.1.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: While the theoretical concepts could be distilled for potential malicious downstream tasks it is not immediately applicable.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The details for the datasets are described in Section 7 and Appendix I.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets are being released.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We did not perform crowdsourcing. We used FFHQ, which contains subjects with human faces, but this is a publicly available dataset.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our research only used publicly available datasets and as such IRB approvals were not required.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs were not used in any part of this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

Organization of the appendix

In Appendix A we discuss previous approaches by exploring posterior guidance and end-to-end guidance in greater detail to provide a more comprehensive overview of how this greedy perspective connects these various works. Appendix B is devoted to the proofs and derivations from Section 4 in the main paper. Likewise, Appendices C and D is devoted to proofs and derivations from Sections 5 and 6 respectively. In Appendix E we discuss some important practical issues when using OTD for guidance, which we believe several to be useful background for the reader. We provide some additional connections between posterior guidance and control signal optimization in Appendix F that we were unable to include in the main paper. Appendix H is devoted to providing a brief background on inverse problems. Likewise, Appendix I is devoted to discussing the implementation details of the numerical experiments in Section 7 and providing a background for the experiments. In Appendix J we include additional results that we could not fit into the main paper. Lastly, in Appendix K we discuss the limitations and broader impacts of this research.

Appendices

A	<i>Related works</i>	26
A.1	<i>Posterior guidance</i>	26
A.2	<i>End-to-end guidance</i>	27
B	<i>A greedy perspective</i>	27
B.1	<i>Additional details on flow models</i>	28
B.2	<i>Assumptions</i>	28
B.3	<i>Proof of Proposition 4.1</i>	28
B.4	<i>Proof of Proposition 4.2</i>	29
B.5	<i>Proof of Proposition 4.3</i>	30
C	<i>Dynamics of guidance</i>	31
C.1	<i>Proof of Theorem 5.1</i>	32
C.2	<i>Dynamics of gradient guidance</i>	33
C.3	<i>Proof of Proposition C.4</i>	34
C.4	<i>Proof of Proposition 5.2</i>	34
C.5	<i>Proof of Theorem 5.3</i>	35
C.6	<i>Proof of Theorem 5.4</i>	36
D	<i>Beyond Euler</i>	37
D.1	<i>Proof of Theorem 6.1</i>	37
D.2	<i>A useful reparameterization of the flow model</i>	38
E	<i>Notes on using OTD in practice</i>	39
F	<i>On control signal optimization</i>	41
F.1	<i>Continuous adjoint equations for control signals</i>	41
G	<i>Implementation details</i>	42
G.1	<i>The construction of the greedy guidance schemes</i>	42
G.2	<i>A PyTorch pseudocode illustration</i>	43
H	<i>A brief introduction to inverse problems</i>	43
H.1	<i>Inverse problems and diffusion models</i>	44
I	<i>Experimental details</i>	44
I.1	<i>Inverse image problems</i>	44
I.2	<i>Molecule generation for QM9</i>	45
I.3	<i>Numerical schemes</i>	46

	<i>I.4 Hardware and compute cost</i>	46
<i>J</i>	<i>Further experimental results</i>	46
	<i>J.1 Molecule generation for QM9</i>	47
	<i>J.2 Further results on inverse image problems</i>	47
	<i>J.3 Sampling trajectories for inverse problems</i>	49
	<i>J.4 More qualitative samples for inverse problems</i>	49
	<i>J.5 More qualitative samples for controlled molecule generation</i>	53
<i>K</i>	<i>Discussions</i>	53
	<i>K.1 Broader Impacts</i>	53
	<i>K.2 Limitations</i>	53

Overview of theoretical results

For convenience we provide a list of theorems to make navigating the theoretical results easier.

4.1	Proposition (Exact solution of affine probability paths)	5
4.2	Proposition (Greedy as an explicit Euler scheme within DTO)	5
4.3	Proposition (Greedy as an implicit Euler scheme within OTD)	5
5.1	Theorem (Jacobian matrices of affine Gaussian probability paths)	6
5.2	Proposition (Dynamics of greedy gradient guidance)	6
5.3	Theorem (Dynamics of gradient vs greedy guidance)	6
5.4	Theorem (Greedy convergence)	7
6.1	Theorem (Truncation error of single-step gradients)	7
4.1	Proposition (Exact solution of affine probability paths)	29
4.2	Proposition (Greedy as an explicit Euler scheme within DTO)	29
4.3	Proposition (Greedy as an implicit Euler scheme within OTD)	30
C.1	Lemma (Gradient of target prediction model)	31
C.2	Lemma (Dynamics of Jacobian matrices for flows)	31
5.1	Theorem (Jacobian matrices of affine Gaussian probability paths)	32
C.4	Proposition (Dynamics of gradient guidance)	34
C.4	Proposition (Dynamics of gradient guidance)	34
5.2	Proposition (Dynamics of greedy gradient guidance)	34
C.4.1	Corollary (Dynamics of gradient vs greedy guidance)	35
5.3	Theorem (Dynamics of gradient vs greedy guidance)	35
5.4	Theorem (Greedy convergence)	36
D.1	Theorem (Local truncation error of discretize-then-optimize gradients)	37
6.1	Theorem (Truncation error of single-step gradients)	38
D.1.1	Corollary (Convergence of a α -th order posterior gradient)	38
D.1.2	Corollary (Dynamics of α -th order posterior gradient)	38
D.2	Proposition (Reparameterized for the target prediction model of affine probability paths)	39

F.1	Theorem (Continuous adjoint equations for the control term)	41
-----	---	----

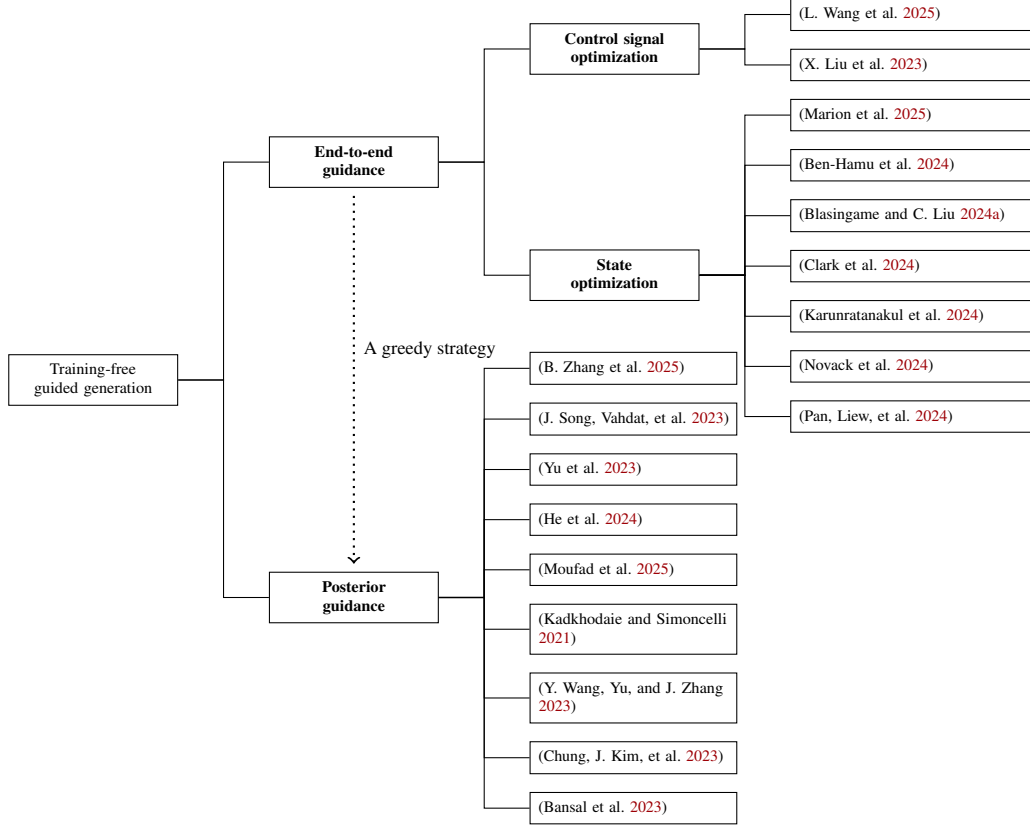


Figure 5: A more detailed taxonomy of *training-free guided generation* methods from Figure 1 from the main paper.

A Related works

We provide a brief summary of previous work exploring either posterior guidance or end-to-end guidance strategies. In Figure 5 we provide a more detailed taxonomy of training-free methods for gradient-based guided generation based on Figure 1 from the main paper.

A.1 Posterior guidance

Recent work in flow/diffusion models has explored the guidance using this strategy; we highlight a few notable examples. Diffusion Posterior Sampling (DPS) (Chung, J. Kim, et al. 2023) is a guidance method that uses Tweedie’s formula (Stein 1981) to estimate the gradient of some guidance function defined in the output state w.r.t. the noisy state, *i.e.*, $\mathbb{E}[\mathbf{X}_1 | \mathbf{X}_t = \mathbf{x}]$. Likewise, the work of Bansal et al. (2023), He et al. (2024), Y. Wang, Yu, and J. Zhang (2023), and Yu et al. (2023) explores similar concepts by employing Tweedie’s formula for diffusion models. Most of these works have explored using the SDE (or Markov chain) formulation of diffusion models rather than the ODE formulation, which is what we primarily focused on in our analysis.

Correcting the guidance trajectory. Several works have explored extensions to the DPS framework by using multiple steps of an SDE solver to correct *errors* made by the guidance steps. In particular, FreeDoM (Yu et al. 2023) explores the usage of a *time-reversal* strategy repeated for a set number of times in each sampling step to correct possible guidance errors. Likewise, recent work by B. Zhang et al. (2025) explored modeling Langevin dynamics on top of a diffusion ODE to correct measurement errors in inverse problems. A significant number of the proposed methods which use posterior guidance arise from solving inverse problems (*cf.* Daras et al. 2024).

Scheduled hyperparameters. Researchers realized that extra performance can be gained in such problems by scheduling hyperparameters like the learning rate (or guidance strength) at different timesteps in the numerical scheme (Moufada et al. 2025; Yu et al. 2023).

Beyond Euler. Recent work by Moufada et al. (2025) explores an extension to (Chung, J. Kim, et al. 2023) by using a two-step method to estimate the guidance gradient. This is mostly closely related to the *greedy (2-step Euler)* method from the main paper, although they use a stochastic sampling method, so it would be more akin to taking two Euler-Maruyama steps.

A.2 End-to-end guidance

Within the last year, many researchers have explored backpropagation through flow/diffusion models for controllable generation. As mentioned in the main paper, the two main strategies for solving such a problem is a DTO or OTD scheme (cf. Appendix E).

Discretize-then-optimize. FlowGrad proposed by X. Liu et al. (2023) uses a DTO scheme to optimize an additional control signal (more details on this later) to perform guidance with flow models. Although the analysis of Ben-Hamu et al. (2024) makes use of the continuous adjoint equations, in practice they use the *generally* preferred approach of DTO with gradient checkpointing.⁵ Likewise, Clark et al. (2024), Karunratanakul et al. (2024), and Novack et al. (2024) all use gradient checkpointing with DTO to perform backpropagation through the flow/diffusion model.

Optimize-then-discretize. Another stream of work has explored the use of continuous adjoint equations to perform the backpropagation. The advantage of such approaches is the $\mathcal{O}(1)$ memory cost, and we enumerate the drawbacks in Appendix E, but suffice to say there are several. To the best of our knowledge, the first work to explore this was Nie et al. (2022) which used OTD with SDEs for the adversarial purification task. More general work came later by Ben-Hamu et al. (2024), Blasingame and C. Liu (2024a), and Pan, Liew, et al. (2024). More specifically, Pan, Liew, et al. (2024) and Pan, Yan, et al. (2023) explore bespoke solvers for the continuous adjoint equations of diffusion ODEs. Blasingame and C. Liu (2024a) extends these works by developing bespoke solvers for diffusion ODEs and SDEs and performs more theoretical analysis of the problem in the SDE setting. Marion et al. (2025) explore using the continuous adjoint equations as a part of a larger bi-level optimization scheme for guided generation. The work of Ben-Hamu et al. (2024) extends the analysis of continuous adjoint equations for diffusion models to flow-based models and provides an alternative perspective to the analysis performed in the earlier works. Recent work by L. Wang et al. (2025) explores an extension of Ben-Hamu et al. (2024) to Riemannian manifolds which incorporates a control signal to the vector field and optimizes both the solution state and *co-state*, they call their approach OC-Flow.

Parallel to these works (conceptually) is the work of Wallace, Gokul, Ermon, et al. (2023) who uses EDICT (Wallace, Gokul, and Naik 2023), an invertible formulation of diffusion models, to perform backpropagation through the diffusion model. Although not presented or viewed this way in the original work, the later work by Blasingame and C. Liu (2024a) showed that this approach can be viewed as a specific discretization scheme of continuous adjoint equations. We note that the EDICT solver, while reversible, is a zeroth-order solver and has poor convergence properties (cf. F. Wang et al. 2024).

Control signal optimization. We discuss this in more detail in Appendix F, but there are several works that explore the optimization of an additional control signal $z(t)$ rather than the solution trajectory $x(t)$; namely, X. Liu et al. (2023) and L. Wang et al. (2025).

B A greedy perspective

We present the proofs and derivations associated with Section 4.

⁵See <https://docs.kidger.site/diffraction/api/adjoints/> for an excellent summary of such design considerations and why DTO is generally preferable over OTD.

B.1 Additional details on flow models

Applying this flow to the random variable $\mathbf{X}_0$ we define a *continuous-time Markov process* $\{\mathbf{X}_t\}_{t \in [0,1]}$ with mapping $\mathbf{X}_t = \Phi_t(\mathbf{X}_0)$. The *goal*, then, is to learn a flow Φ_t such that $\mathbf{X}_1 = \Phi_1(\mathbf{X}_0) \sim q(\mathbf{x})$. This procedure amounts to learning a neural network parameterized vector field $\mathbf{u}^\theta \in \mathcal{C}^{1,r}([0,1] \times \mathbb{R}^d; \mathbb{R}^d)$; this learning procedure can be performed efficiently through a *simulation-free* training process known as *flow matching* (Lipman, R. T. Q. Chen, et al. 2023) or more generally *generator matching* (Holderrieth et al. 2025).

Throughout the rest of this paper we will assume a standard flow model trained to zero loss and we denote the parameterized flow model via $\Phi_t^\theta(\mathbf{x})$. We let $\Phi_{s,t}(\mathbf{x}) = (\Phi_t \circ \Phi_s^{-1})(\mathbf{x})$ denote the flow from time s to time t , $s, t \in [0,1]$.

Affine probability paths. A special subset of flow models, are flows which model an *affine probability path*, i.e., given a schedule (α_t, σ_t) the random process $\{\mathbf{X}_t\}$ is described via the affine equation

$$\mathbf{X}_t = \alpha_t \mathbf{X}_1 + \sigma_t \mathbf{X}_0, \quad (16)$$

where $\alpha_t, \sigma_t \in \mathcal{C}^\infty([0,1]; [0,1])$ which satisfy

$$\alpha_0 = \sigma_1 = 0, \quad \alpha_1 = \sigma_0 = 1, \quad \forall t \in (0,1) \quad [\dot{\alpha}_t > 0, \dot{\sigma}_t < 0]. \quad (17)$$

The *marginal vector field* can then be expressed as the following conditional expectation:

$$\mathbf{u}_t(\mathbf{x}) = \mathbb{E}[\dot{\alpha}_t \mathbf{X}_1 + \dot{\sigma}_t \mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]. \quad (18)$$

This *nice* form of the marginal vector field enables use to rewrite the vector field in the forms of either source (Ho, Jain, and Abbeel 2020) or target (Kingma et al. 2021) prediction as

$$\mathbf{u}_t(\mathbf{x}) = \underbrace{\frac{\dot{\beta}_t}{\beta_t}}_{=a_t} \mathbf{x} + \underbrace{\frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t}}_{=b_t} \mathbf{f}_t(\mathbf{x}), \quad (19)$$

where $\beta_t = -\alpha_t$ for source prediction with $\mathbf{f}_t(\mathbf{x}) = \mathbf{x}_{0|t}(\mathbf{x}) = \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]$ and $\beta_t = \sigma_t$ for target prediction with $\mathbf{f}_t(\mathbf{x}) = \mathbf{x}_{1|t}(\mathbf{x}) = \mathbb{E}[\mathbf{X}_1 | \mathbf{X}_t = \mathbf{x}]$; and a_t, b_t are useful shorthands to be used later.

Remark B.1. The probability flow ODE formulation of *diffusion models* (Y. Song, Sohl-Dickstein, et al. 2021) is subsumed by flow models, and represents a model with an affine Gaussian probability paths (AGGP), i.e., $(\mathbf{X}_0, \mathbf{X}_1) \sim \pi_{0,1}(\mathbf{x}_0, \mathbf{x}_1) = p(\mathbf{x}_0)q(\mathbf{x}_1)$ with $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \mathbf{0}, \sigma^2 \mathbf{I})$ (Lipman, Havasi, et al. 2024). Thus without loss of generality we consider flow models of affine probability paths.⁶

B.2 Assumptions

Throughout the norm $\|\cdot\|$ corresponds to the Euclidean norm $\|\cdot\|_2$. Additionally, we make the following (mild) regularity assumptions:

Assumption B.1. The function $a_t := \frac{\dot{\alpha}_t}{\sigma_t}$ is integrable in $[0,1]$.

Assumption B.2. The total derivatives $\frac{d^n}{d\gamma^n} [\mathbf{x}_{1|\gamma}^\theta(\mathbf{x})]$ exist and are continuous for $0 \leq n \leq k-1$.

Assumption B.1 is necessary for the simplification that we perform with exponential integrators and Ben-Hamu et al. (2024) make the same assumption in their analysis of the continuous adjoint equations for affine probability paths. Assumption B.2 is to ensure that we can take a Taylor expansion of $\mathbf{x}_{1|\gamma}^\theta(\mathbf{x})$.

B.3 Proof of Proposition 4.1

We restate Proposition 4.1 below.

⁶Clearly, diffusion models which solve the reverse-time SDE are different and require a separate analysis.

Proposition 4.1 (Exact solution of affine probability paths). *Given an initial value of $\mathbf{x}_s$ at time $s \in [0, 1]$ the solution $\mathbf{x}_t$ at time $t \in [0, 1]$ of an ODE governed by the vector field in Equation (18) is:*

$$\mathbf{x}_t = \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \mathbf{x}_{1|\gamma}^\theta(\mathbf{x}_\gamma) d\gamma. \quad (8)$$

Proof. Recall that we uniquely define a flow model through the vector field $\mathbf{u} \in \mathcal{C}^{1,1}([0, 1] \times \mathbb{R}^d; \mathbb{R}^d)$. The vector field which models the affine conditional flow with schedule (α_t, σ_t) , is defined as

$$\mathbf{u}_t^\theta(\mathbf{x}) = \mathbb{E}[\dot{\alpha}_t \mathbf{X}_1 + \dot{\sigma}_t \mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]. \quad (20)$$

With some simple algebra, we can rewrite the vector field in terms of $\hat{\mathbf{x}}_{1|t}$,

$$\begin{aligned} \mathbf{u}_t^\theta(\mathbf{x}) &= a_t \mathbf{x} + b_t \hat{\mathbf{x}}_{1|t}^\theta(\mathbf{x}), \\ a_t &= \frac{\dot{\sigma}_t}{\sigma_t} \quad b_t = \dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}. \end{aligned} \quad (21)$$

Now using this definition we can rewrite the solution for $\mathbf{x}_t$ from $\mathbf{x}_s$ in terms of $\hat{\mathbf{x}}_{1|t}$,

$$\mathbf{x}_t = \mathbf{x}_s + \int_s^t \mathbf{u}_\tau^\theta(\mathbf{x}_\tau) d\tau, \quad (22)$$

$$\mathbf{x}_t = \mathbf{x}_s + \int_s^t a_\tau \mathbf{x}_\tau + b_\tau \hat{\mathbf{x}}_{1|\tau}^\theta(\mathbf{x}_\tau) d\tau. \quad (23)$$

Note the semi-linear form of the integral equation. We can exploit this structure using the technique of *exponential integrators*, (see Gonzalez et al. 2024; Lu et al. 2022a; Q. Zhang and Y. Chen 2023), to simplify Equation (23), under Assumption B.1, to

$$\mathbf{x}_t = e^{\int_s^t a_u du} \mathbf{x}_s + \int_s^t e^{\int_\tau^t a_u du} b_\tau \hat{\mathbf{x}}_{1|\tau}^\theta(\mathbf{x}_\tau) d\tau. \quad (24)$$

Now, the integrating factor simplifies quite nicely to

$$e^{\int_s^t a_u du} = e^{\int_s^t \frac{\dot{\sigma}_u}{\sigma_u} du} = e^{\int_s^t \frac{1}{\sigma} d\sigma} = \frac{\sigma_t}{\sigma_s}, \quad (25)$$

such that Equation (24) becomes

$$\mathbf{x}_t = \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + \sigma_t \int_s^t \frac{b_\tau}{\sigma_\tau} \hat{\mathbf{x}}_{1|\tau}^\theta(\mathbf{x}_\tau) d\tau. \quad (26)$$

We can simplify b_t/σ_t to find:

$$\frac{b_t}{\sigma_t} = \frac{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}{\sigma_t^2} = \frac{d}{dt} \left(\frac{\alpha_t}{\sigma_t} \right) = \frac{d}{dt} \gamma_t, \quad (27)$$

where $\gamma_t := \alpha_t/\sigma_t$, i.e., the signal-to-noise ratio. As such, we can rewrite Equation (26) with a change of variables $\mathbf{x}_\gamma = \mathbf{x}_{\gamma_t^{-1}(\gamma)} = \mathbf{x}_t$,

$$\mathbf{x}_t = \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \mathbf{x}_{1|\gamma}^\theta(\mathbf{x}_\gamma) d\gamma, \quad (28)$$

concluding the proof. $\square$

B.4 Proof of Proposition 4.2

Proposition 4.2 (Greedy as an explicit Euler scheme within DTO). *For some trajectory state $\mathbf{x}_t$ at time t , the greedy gradient given by $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{1|t}^\theta(\mathbf{x}))$ is the DTO scheme with an explicit Euler discretization with step size $h = \gamma_1 - \gamma_t$.*

Proof. From Proposition 4.1 we see that using the target prediction model to estimate $\mathbf{x}_1$ is akin to taking a first-order approximation of the flow. More specifically, under Assumption B.2 we can construct a $(k-1)$ -th Taylor expansion of Equation (8) with:

$$\mathbf{x}_t = \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + \sigma_t \sum_{n=0}^{k-1} \frac{d^n}{d\gamma^n} \left[\mathbf{x}_{1|\gamma}^\theta(\mathbf{x}_\gamma) \right]_{\gamma=\gamma_s} \int_{\gamma_s}^{\gamma_t} \frac{(\gamma - \gamma_s)^n}{n!} d\gamma + \mathcal{O}(h^{k+1}), \quad (29)$$

$$= \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + \sigma_t \sum_{n=0}^{k-1} \frac{d^n}{d\gamma^n} \left[\mathbf{x}_{1|\gamma}^\theta(\mathbf{x}_\gamma) \right]_{\gamma=\gamma_s} \frac{h^{n+1}}{(n+1)!} + \mathcal{O}(h^{k+1}), \quad (30)$$

where $h := \gamma_t - \gamma_s$ is the step size. Then it follows that for $k=1$ the first-order discretization of the flow, omitting high-order error terms becomes,

$$\mathbf{x}_t \approx \tilde{\mathbf{x}}_t = \frac{\sigma_t}{\sigma_s} \mathbf{x}_s + (\alpha_t + \frac{\sigma_t \alpha_s}{\sigma_s}) \mathbf{x}_{1|s}^\theta(\mathbf{x}_s). \quad (31)$$

In the limit as $t \rightarrow 1$ we have $\tilde{\mathbf{x}}_t = \mathbf{x}_{1|s}^\theta(\mathbf{x}_s)$.⁷ Thus, the greedy gradient is a DTO scheme with an explicit Euler discretization with step size $h = \gamma_1 - \gamma_t$. $\square$

B.5 Proof of Proposition 4.3

We restate Proposition 4.3 below.

Proposition 4.3 (Greedy as an implicit Euler scheme within OTD). *For some trajectory state $\mathbf{x}_t$ at time t , the greedy gradient given by $\nabla_{\mathbf{x}_t} \mathcal{L}(\mathbf{x}_{1|t}^\theta(\mathbf{x}_t))$ is an implicit Euler discretization of the continuous adjoint equations for the true gradients with step size $h = \gamma_1 - \gamma_t$.*

For clarity we restate the definition of the continuous adjoint equations. Let $\mathbf{u}_\theta \in \mathcal{C}^{1,1}([0, 1] \times \mathbb{R}^d; \mathbb{R}^d)$ be a model that models the vector field of some ODE and be Lipschitz continuous in its second argument. Let $\mathbf{x} : [0, 1] \rightarrow \mathbb{R}^d$ be the solution to the ODE with the initial condition $\mathbf{x}_0 \in \mathbb{R}^d$, $\dot{\mathbf{x}}_t = \mathbf{u}_\theta(t, \mathbf{x}_t)$. For some scalar-valued loss function $\mathcal{L} \in \mathcal{C}^2(\mathbb{R}^d)$ in $\mathbf{x}_1$, let $\mathbf{a}_\mathbf{x} := \partial \mathcal{L} / \partial \mathbf{x}_t$ denote the gradient. Then $\mathbf{a}_\mathbf{x}$ and related quantity $\mathbf{a}_\theta := \partial \mathcal{L} / \partial \theta$ can be found by solving an augmented ODE of the form,

$$\begin{aligned} \mathbf{a}_\mathbf{x}(1) &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_1}, & \frac{d\mathbf{a}_\mathbf{x}}{dt}(t) &= -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \mathbf{u}_\theta}{\partial \mathbf{x}}(t, \mathbf{x}_t), \\ \mathbf{a}_\theta(1) &= \mathbf{0}, & \frac{d\mathbf{a}_\theta}{dt}(t) &= -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \mathbf{u}_\theta}{\partial \theta}(t, \mathbf{x}_t). \end{aligned} \quad (32)$$

Now we present the proof.

Proof. The adjoint state can be simplified by rewriting the vector field in terms of the target prediction model to find

$$\frac{d\mathbf{a}_\mathbf{x}}{dt}(t) = -a_t \mathbf{a}_\mathbf{x}(t) - b_t \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \mathbf{x}_{1|t}^\theta(\mathbf{x}_t)}{\partial \mathbf{x}_t}. \quad (33)$$

We can express this *backwards-in-time* ODE as an integral equation in the form of

$$\begin{aligned} \mathbf{a}_\mathbf{x}(s) &= \mathbf{a}_\mathbf{x}(t) - \int_t^s a_\tau \mathbf{a}_\mathbf{x}(\tau) + b_\tau \mathbf{a}_\mathbf{x}(\tau)^\top \frac{\partial \mathbf{x}_{1|\tau}^\theta(\mathbf{x}_\tau)}{\partial \mathbf{x}_\tau} d\tau, \\ &= \mathbf{a}_\mathbf{x}(t) + \int_s^t a_\tau \mathbf{a}_\mathbf{x}(\tau) + b_\tau \mathbf{a}_\mathbf{x}(\tau)^\top \frac{\partial \mathbf{x}_{1|\tau}^\theta(\mathbf{x}_\tau)}{\partial \mathbf{x}_\tau} d\tau. \end{aligned} \quad (\text{time-reversal}) \quad (34)$$

⁷Note that despite $\sigma_t \rightarrow 0$ the asymptotic behavior is well-defined (see Ben-Hamu et al. 2024).

Using the technique of exponential integrators we rewrite the integral as

$$\begin{aligned}
\mathbf{a}_x(s) &= e^{\int_s^t a_u \, du} \mathbf{a}_x(t) + \int_s^t e^{\int_s^u a_v \, dv} b_\tau \mathbf{a}_x(\tau)^\top \frac{\partial \mathbf{x}_{1|\tau}^\theta(\mathbf{x}_\tau)}{\partial \mathbf{x}_\tau} \, d\tau, \\
&= \frac{\sigma_t}{\sigma_s} \mathbf{a}_x(t) + \sigma_t \int_s^t \frac{b_\tau}{\sigma_\tau} \mathbf{a}_x(\tau)^\top \frac{\partial \mathbf{x}_{1|\tau}^\theta(\mathbf{x}_\tau)}{\partial \mathbf{x}_\tau} \, d\tau, \\
&= \frac{\sigma_t}{\sigma_s} \mathbf{a}_x(t) + \sigma_t \int_{\gamma_s}^{\gamma_t} \mathbf{a}_x(\gamma)^\top \frac{\partial \mathbf{x}_{1|\gamma}^\theta(\mathbf{x}_\gamma)}{\partial \mathbf{x}_\gamma} \, d\gamma.
\end{aligned} \tag{35}$$

By Assumption B.2 it follows that the vector-Jacobian product has $(k-1)$ -th total derivatives, allowing us to define a first-order Taylor expansion around γ_s :

$$\mathbf{a}_x(s) = \frac{\sigma_t}{\sigma_s} \mathbf{a}_x(t) + (\alpha_t - \frac{\sigma_t}{\sigma_s} \alpha_s) \mathbf{a}_x(s)^\top \frac{\partial \hat{\mathbf{x}}_{1|s}(\mathbf{x}_s)}{\partial \mathbf{x}_s} + \mathcal{O}(h^2). \tag{36}$$

Thus, the first-order approximation of the adjoint state at time t with a step size of $h = \gamma_1 - \gamma_t$ is the implicit equation

$$\mathbf{a}_x(t) = \mathbf{a}_x(t)^\top \frac{\partial \hat{\mathbf{x}}_{1|t}(\mathbf{x}_t)}{\partial \mathbf{x}_t}. \tag{37}$$

Now to solve the implicit equation we can use the fixed-point iteration method. Let $\mathbf{a}_x(t)^{(0)} = \mathbf{a}_x(1)$, then the first iteration has

$$\mathbf{a}_x(t)^{(1)} = \mathbf{a}_x(1)^\top \frac{\partial \hat{\mathbf{x}}_{1|t}(\mathbf{x}_t)}{\partial \mathbf{x}_t} = \nabla_{\mathbf{x}_t} \mathcal{L}(\hat{\mathbf{x}}_{1|t}(\mathbf{x}_t)). \tag{38}$$

Thus, we have shown that the greedy gradients are equivalent to the first iteration of an implicit Euler discretization of the continuous adjoint equations. $\square$

C Dynamics of guidance

In this section we detail some of the formalisms omitted in the main paper concerning the dynamics of the gradient flow and greedy gradients.

We begin by re-establishing some useful prior results. Ben-Hamu et al. (2024, Proposition 4.1) showed that the gradient of the target prediction model is proportional to the variance of the random variable defined by $p_{1|t}(\mathbf{x}_1|\mathbf{x})$, we restate their result below.

Lemma C.1 (Gradient of target prediction model). *For affine Gaussian probability paths, the gradient of the target prediction model $\mathbf{x}_{1|t}^\theta(\mathbf{x})$ w.r.t. $\mathbf{x}$ is proportional to the variance of $p_{1|t}(\mathbf{x}_1|\mathbf{x})$, i.e.,*

$$\nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x}) = \frac{\alpha_t}{\sigma_t^2} \text{Var}_{1|t}(\mathbf{x}), \tag{39}$$

where

$$\text{Var}_{1|t}(\mathbf{x}) = \mathbb{E}_{p_{1|t}(\mathbf{x}_1|\mathbf{x})} \left[(\mathbf{x}_1 - \mathbf{x}_{1|t}^\theta(\mathbf{x})) (\mathbf{x}_1 - \mathbf{x}_{1|t}^\theta(\mathbf{x}))^\top \right]. \tag{40}$$

Remark C.1. This can be written more generally in terms of the (pushforward) differential $D_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x})$ where the underlying spaces are smooth manifolds and $\mathbf{x}_{1|t}^\theta$ is a smooth map between them (Ben-Hamu et al. 2024). In this section, we only consider flow models defined in Euclidean spaces, and so we opt not to elaborate on this generalization.

We restate a well-known result below in Lemma C.2 regarding the continuous-time analogue to forward-mode autodifferentiation, or in other words, forward sensitivity.

Lemma C.2 (Dynamics of Jacobian matrices for flows). *Let $\mathbf{x}_0 \in \mathbb{R}^d$ and let $\mathbf{f} \in \mathcal{C}^{1,1}([0, T] \times \mathbb{R}^d; \mathbb{R}^d)$ be uniformly Lipschitz in $\mathbf{x}$. Let $\mathbf{x} : [0, T] \rightarrow \mathbb{R}^d$ be the unique solution to*

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \frac{d\mathbf{x}}{dt}(t) = \mathbf{f}(t, \mathbf{x}(t)). \tag{41}$$

Let $\Phi_{s,t}(\mathbf{x})$, $s, t \in [0, T]$ denote the flow associated with Equation (41). Then let $\mathbf{J}_s(t) := \nabla_{\mathbf{x}} \Phi_{s,t}(\mathbf{x})$ denote the Jacobian matrices, where $\mathbf{J}_s : [s, T] \rightarrow \mathbb{R}^{d \times d}$ solve the differential equation

$$\mathbf{J}_s(s) = \mathbf{I}, \quad \frac{d\mathbf{J}_s}{dt}(t) = \nabla_{\mathbf{x}} \mathbf{f}(t, \Phi_{s,t}(\mathbf{x}(s))) \mathbf{J}_s(t), \quad (42)$$

where $\nabla_{\mathbf{x}} \mathbf{f}(t, \cdot)$ refers to the gradient w.r.t. the second argument.

Remark C.2. This result is well known and has been extended to *controlled differential equations* (Friz and Victoir 2010, Theorem 4.4) and *rough differential equations* (Friz and Victoir 2010, Theorem 11.3). Kidger (2022, Theorem 5.8) discusses this result for neural ODEs.

C.1 Proof of Theorem 5.1

We restate Theorem 5.1 below.

Theorem 5.1 (Jacobian matrices of affine Gaussian probability paths). *For the standard affine Gaussian probability path with flow model $\Phi_{s,t}^\theta(\mathbf{x})$, the Jacobian matrix $\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x})$ as function of $\mathbf{x}$ is given as the solution to*

$$\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) = \frac{\sigma_t}{\sigma_s} \mathbf{I} + \sigma_t \int_s^t \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \text{Var}_{1|u}(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du, \quad (10)$$

where

$$\text{Var}_{1|t}(\mathbf{x}) = \mathbb{E}_{p_{1|t}(\mathbf{x}_1|\mathbf{x})} \left[(\mathbf{x}_1 - \mathbf{x}_{1|t}^\theta(\mathbf{x})) (\mathbf{x}_1 - \mathbf{x}_{1|t}^\theta(\mathbf{x}))^\top \right]. \quad (11)$$

This proof follows a similar technique to that used by Blasingame and C. Liu (2024a) to simplify adjoint equations for diffusion models using exponential integrators.

Proof. Now recall Lemma C.2 which discusses the dynamics of Jacobian matrices for flows, rewriting this as an integral equation yields:

$$\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) = \mathbf{I} + \int_s^t \nabla_{\mathbf{x}_u} \mathbf{u}_u^\theta(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du. \quad (43)$$

Now recall the definition of the marginal vector field in terms of the target prediction model (cf. Equation (19)) which we use to rewrite Equation (43) as

$$\begin{aligned} \nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) &= \mathbf{I} + \int_s^t \nabla_{\mathbf{x}_u} a_u \Phi_{s,u}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) + \nabla_{\mathbf{x}_u} b_u \mathbf{x}_{1|u}^\theta(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du, \\ &\stackrel{(i)}{=} \mathbf{I} + \int_s^t a_u \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) + b_u \nabla_{\mathbf{x}_u} \mathbf{x}_{1|u}^\theta(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du, \end{aligned} \quad (44)$$

where (i) holds by $\nabla_{\mathbf{x}_u} \Phi_{s,u}^\theta(\mathbf{x}) = \mathbf{I}$. Next we can make use of the popular technique of *exponential integrators* to simplify Equation (43) in combination with Equation (19). Thus, the integral equation in Equation (44) becomes

$$\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) = \Lambda_a(s, t) \mathbf{I} + \int_s^t \Lambda_a(u, t) b_u \nabla_{\mathbf{x}_u} \mathbf{x}_{1|u}^\theta(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du, \quad (45)$$

where $\Lambda_a(s, t) := \exp \int_s^t a_u du$ is the integrating factor. This simplifies to $\Lambda_a(s, t) = \sigma_t / \sigma_s$. Using this, Equation (45) can be simplified to

$$\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) = \frac{\sigma_t}{\sigma_s} \mathbf{I} + \sigma_t \int_s^t \frac{b_u}{\sigma_u} \nabla_{\mathbf{x}_u} \mathbf{x}_{1|u}^\theta(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du. \quad (46)$$

Now we can apply Lemma C.1 to further simplify Equation (46) to find

$$\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) = \frac{\sigma_t}{\sigma_s} \mathbf{I} + \sigma_t \int_s^t \frac{\alpha_u}{\sigma_u^3} b_u \text{Var}_{1|u}(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^\theta(\mathbf{x}) du. \quad (47)$$

Next we simplify the coefficient $\alpha_u b_u / \sigma_u^3$ in the integral term. Let $\gamma_t := \alpha_t / \sigma_t$ equal the signal-to-noise-ratio. Then we observe

$$\begin{aligned}
b_t \frac{\alpha_t}{\sigma_t^3} &= \left(\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t} \right) \frac{\alpha_t}{\sigma_t^3}, \\
&= \frac{\dot{\alpha}_t \sigma_t - \dot{\sigma}_t \alpha_t}{\sigma_t^3} \frac{\alpha_t}{\sigma_t^2}, \\
&\stackrel{(i)}{=} \frac{d}{dt} \left[\frac{\alpha_t}{\sigma_t} \right] \frac{\alpha_t}{\sigma_t} \frac{1}{\sigma_t}, \\
&\stackrel{(ii)}{=} \dot{\gamma}_t \frac{\gamma_t}{\sigma_t},
\end{aligned} \tag{48}$$

where (i) holds by the quotient rule and (ii) holds by definition of γ_t . Using this simplification we can perform a change-of-variables to simplify the gradient resulting in

$$\nabla_{\mathbf{x}} \Phi_{s,t}^{\theta}(\mathbf{x}) = \frac{\sigma_t}{\sigma_s} \mathbf{I} + \sigma_t \int_s^t \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \text{Var}_{1|u}(\Phi_{s,u}^{\theta}(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{s,u}^{\theta}(\mathbf{x}) \, du. \tag{49}$$

□

Remark C.3. Readers familiar with the work of Ben-Hamu et al. (2024) may notice some similarities between our result Theorem 5.1 and Ben-Hamu et al. (2024, Theorem 4.2). The difference between the two is that the former is a simplified integral equation; whereas, the latter is the exact solution and no longer requires solving an ODE. However, this later solution does require solving a time-ordered exponential which requires a formal truncated series expansion, *e.g.*, Magnus expansion.

Theorem 5.1 is closely related to Ben-Hamu et al. (2024, Theorem 4.2) which we restate below within the context of our notational conventions.⁸

Theorem C.3. *For the standard affine Gaussian probability path, the differential of $\Phi_{0,1}^{\theta}(\mathbf{x})$ as of function of $\mathbf{x}$ is*

$$\nabla_{\mathbf{x}} \Phi_{0,1}^{\theta}(\mathbf{x}) = \sigma_1 \mathcal{T} \exp \left[\int_0^1 \frac{1}{2} \dot{\gamma}_t^2 \text{Var}_{1|t}(\mathbf{x}) \, dt \right], \tag{50}$$

where $\mathcal{T} \exp$ denotes the time-ordered exponential.

The time-ordered exponential⁹ (Grossman and Katz 1972) is defined as

$$\begin{aligned}
\mathcal{T} \exp \left[\int_t^1 \mathbf{A}(s) \, ds \right] &= \sum_{n=0}^{\infty} \frac{1}{n!} \int_t^1 ds_1 \cdots \int_t^1 ds_n \, \mathcal{T} \{ \mathbf{A}(s_1) \cdots \mathbf{A}(s_n) \}, \\
&= \sum_{n=0}^{\infty} \int_t^1 ds_1 \int_t^{s_1} ds_2 \cdots \int_t^{s_{n-1}} ds_n \, \mathbf{A}(s_1) \mathbf{A}(s_2) \cdots \mathbf{A}(s_n),
\end{aligned} \tag{51}$$

and the solution can be found the Dyson series (Sakurai and Napolitano 2020) or Magnus expansion (Magnus 1954), which are truncated in practice. The meta-operator $\mathcal{T}$ denotes the time-ordering (Dyson 1949), *e.g.*, consider the time-ordering of two operators $\mathbf{A}, \mathbf{B}$:

$$\mathcal{T} \{ \mathbf{A}(s_1) \mathbf{B}(s_2) \} := \begin{cases} \mathbf{A}(s_1) \mathbf{B}(s_2) & \text{if } s_1 > s_2, \\ \pm \mathbf{B}(s_2) \mathbf{A}(s_1) & \text{otherwise.} \end{cases} \tag{52}$$

For more details we refer the reader to Weinberg (1995).

C.2 Dynamics of gradient guidance

We state this more formally below in Proposition C.4.

⁸With abuse of notation let $\dot{\gamma}_t^2$ denote the time derivative of γ_t^2 .

⁹This is closely related to the *Peano-Baker series* (see Frazer, Duncan, and Collar 1938, Section 7.5).

Proposition C.4 (Dynamics of gradient guidance). *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\mathbf{x}$ at some time $t \in [0, 1]$ in the direction of the gradient $\nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x}))$ is given by*

$$\delta_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1). \quad (53)$$

Thus the behavior of $\mathbf{x}_1$ when guided by $\mathcal{L}$ is determined by the operator $\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})$ which iteratively projects the gradient of the loss function by the covariance matrix $\text{Var}_{1|t}(\mathbf{x})$. Put another way:

Performing gradient guidance with $\mathcal{L}$ at time $t < 1$ amounts to guidance which follows the target distribution $p(\mathbf{X}_1)$ by projecting $\nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1)$ onto to the target distribution via the local covariance matrix.

It is for this reason that it is undesirable to simply perform guidance in the data space as we are likely to deviate from this target distribution. From Equation (53) we know that applying the gradient at earlier timesteps causes the initial gradient $\nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1)$ to be projected into high-variance directions of the target distribution causing the guided sample to stay closer to the true target distribution.

The next question is: how does $\mathbf{x}_1$ change when $\mathbf{x}$ is updated with our greedy guidance strategy?

C.3 Proof of Proposition C.4

We restate Proposition C.4 below.

Proposition C.4 (Dynamics of gradient guidance). *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\mathbf{x}$ at some time $t \in [0, 1]$ in the direction of the gradient $\nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x}))$ is given by*

$$\delta_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1). \quad (53)$$

Proof. This can be shown from a straightforward derivation:

$$\begin{aligned} \delta_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) &\stackrel{(i)}{=} \left. \frac{d}{d\eta} \right|_{\eta=0} \Phi_{t,1}^\theta(\mathbf{x} - \eta \nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x}))), \\ &\stackrel{(ii)}{=} -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x} - \eta \nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x}))) \nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x})) \Big|_{\eta=0}, \\ &= -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}^\theta(\mathbf{x})), \\ &\stackrel{(iii)}{=} -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1), \end{aligned} \quad (54)$$

where (i) holds by the definition of the Gateaux differential, (ii) holds by the chain rule, and (iii) holds by a substitution of Equation (9) with the simplification of $\mathbf{x}_1 = \Phi_{t,1}^\theta(\mathbf{x})$. $\square$

C.4 Proof of Proposition 5.2

We restate Proposition 5.2 below.

Proposition 5.2 (Dynamics of greedy gradient guidance). *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\mathbf{x}$ at some time $t \in [0, 1]$ in the direction of the gradient $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{1|t}^\theta(\mathbf{x}))$ is given by*

$$\delta_{\mathbf{x}}^G \Phi_{t,1}^\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1). \quad (13)$$

Proof. This can be shown from a straightforward derivation:

$$\begin{aligned}
\delta_{\mathbf{x}}^G \Phi_{t,1}^\theta(\mathbf{x}) &\stackrel{(i)}{=} \frac{d}{d\eta} \bigg|_{\eta=0} \Phi_{t,1}^\theta \left(\mathbf{x} - \eta \nabla_{\mathbf{x}} \mathcal{L} \left(\mathbf{x}_{1|t}^\theta(\mathbf{x}) \right) \right), \\
&\stackrel{(ii)}{=} -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta \left(\mathbf{x} - \eta \nabla_{\mathbf{x}} \mathcal{L} \left(\Phi_{t,1}^\theta(\mathbf{x}) \right) \right) \nabla_{\mathbf{x}} \mathcal{L} \left(\mathbf{x}_{1|t}^\theta(\mathbf{x}) \right) \bigg|_{\eta=0}, \\
&= -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \mathcal{L} \left(\mathbf{x}_{1|t}^\theta(\mathbf{x}) \right), \\
&\stackrel{(iii)}{=} -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1),
\end{aligned} \tag{55}$$

where (i) holds by the definition of the Gateaux differential, (ii) holds by the chain rule, and (iii) holds by the chain rule. $\square$

We note an interesting corollary below.

Corollary C.4.1 (Dynamics of gradient vs greedy guidance). *The difference between the dynamics of gradient guidance in Proposition C.4 and greedy gradient guidance in Proposition 5.2 for a point $\mathbf{x}$ at time t with guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$ is*

$$\left\| \delta_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) - \delta_{\mathbf{x}}^G \Phi_{t,1}^\theta(\mathbf{x}) \right\| = \left\| \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \left(\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x}) \right)^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1) \right\|. \tag{56}$$

C.5 Proof of Theorem 5.3

We restate Theorem 5.3 below.

Theorem 5.3 (Dynamics of gradient vs greedy guidance). *The difference between the dynamics of gradient guidance in Proposition C.4 and greedy gradient guidance in Proposition 5.2 for a point $\mathbf{x}$ at time t with guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$ is bounded by $\mathcal{O}(h^2)$ where $h := \gamma_1 - \gamma_t$, i.e.,*

$$\left\| \nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^\theta(\mathbf{x}) \right\| = \mathcal{O}(h^2). \tag{14}$$

Proof. From Corollary C.4.1 it is clear that the difference between $\delta_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x})$ and $\delta_{\mathbf{x}}^G \Phi_{t,1}^\theta(\mathbf{x})$ amounts to the difference between the true gradient and gradient of the target prediction model. Recall Theorem 5.1 which enables to write the gradient as the solution to an integral equation:

$$\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) = \frac{\sigma_1}{\sigma_t} \mathbf{I} + \sigma_1 \int_t^1 \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \text{Var}_{1|u}(\Phi_{s,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{t,u}^\theta(\mathbf{x}) \, du. \tag{57}$$

Now as $\sigma_t \rightarrow 0$ as $t \rightarrow 1$, we can simplify the integral equation

$$\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) = \sigma_1 \int_t^1 \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \text{Var}_{1|u}(\Phi_{t,u}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{t,u}^\theta(\mathbf{x}) \, du, \tag{58}$$

and then by rewriting the integral in terms of $d\gamma = \dot{\gamma}_u du$ we find

$$\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) = \sigma_1 \int_{\gamma_t}^{\gamma_1} \frac{\gamma}{\sigma_\gamma} \text{Var}_{1|\gamma}(\Phi_{\gamma_t,\gamma}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{\gamma_t,\gamma}^\theta(\mathbf{x}) \, d\gamma. \tag{59}$$

Next we take a first-order Taylor expansion of $\frac{1}{\sigma_\gamma} \text{Var}_{1|\gamma}(\Phi_{\gamma_t,\gamma}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{\gamma_t,\gamma}^\theta(\mathbf{x})$ centered at γ_t which yields:

$$\frac{\gamma}{\sigma_\gamma} \text{Var}_{1|\gamma}(\Phi_{\gamma_t,\gamma}^\theta(\mathbf{x})) \nabla_{\mathbf{x}} \Phi_{\gamma_t,\gamma}^\theta(\mathbf{x}) = \frac{\gamma_t}{\sigma_t} \text{Var}_{1|t}(\mathbf{x}) + \mathcal{O}(\gamma - \gamma_t). \tag{60}$$

For this analysis, it is actually more convenient to include the γ term as part of the Taylor expansion rather than computing it in closed form in the integral. Now plugging Equation (60) into Equation (59)

yields

$$\begin{aligned}
\nabla_{\mathbf{x}} \Phi_{t,1}^{\theta}(\mathbf{x}) &= \sigma_1 \int_{\gamma_t}^{\gamma_1} \gamma \frac{1}{\sigma_t} \text{Var}_{1|t}(\mathbf{x}) + \mathcal{O}(\gamma - \gamma_t) \, d\gamma, \\
&\stackrel{(i)}{=} \sigma_1 \frac{\gamma_t}{\sigma_t} \text{Var}_{1|t}(\mathbf{x}) \int_{\gamma_t}^{\gamma_1} d\gamma + \mathcal{O}(h^2), \\
&= \sigma_1 \frac{\gamma_t}{\sigma_t} \text{Var}_{1|t}(\mathbf{x}) (\gamma_1 - \gamma_t) + \mathcal{O}(h^2),
\end{aligned} \tag{61}$$

where (i) holds with $h := \gamma_1 - \gamma_t$. Then, with a little algebra we have

$$\begin{aligned}
\nabla_{\mathbf{x}} \Phi_{t,1}^{\theta}(\mathbf{x}) &= \sigma_1 \frac{\alpha_t}{\sigma_t^2} (\gamma_1 - \gamma_t) \text{Var}_{1|t}(\mathbf{x}) + \mathcal{O}(h^2), \\
&= \sigma_1 \frac{\alpha_t}{\sigma_t^2} \left(\frac{\alpha_1}{\sigma_1} - \frac{\alpha_t}{\sigma_t} \right) \text{Var}_{1|t}(\mathbf{x}) + \mathcal{O}(h^2), \\
&= \frac{\alpha_t}{\sigma_t^2} \left(\alpha_1 - \sigma_1 \frac{\alpha_t}{\sigma_t} \right) \text{Var}_{1|t}(\mathbf{x}) + \mathcal{O}(h^2), \\
&\stackrel{(i)}{=} \frac{\alpha_t}{\sigma_t^2} \text{Var}_{1|t}(\mathbf{x}) + \mathcal{O}(h^2),
\end{aligned} \tag{62}$$

where (i) holds by the boundary conditions of the schedule (cf. Equation (17)). Now recall Lemma C.1 which states:

$$\nabla_{\mathbf{x}} \mathbf{x}_{1|t}^{\theta}(\mathbf{x}) = \frac{\alpha_t}{\sigma_t^2} \text{Var}_{1|t}(\mathbf{x}). \tag{63}$$

Thus from Equation (62) and Equation (63) it is easy to see that

$$\left\| \nabla_{\mathbf{x}} \Phi_{t,1}^{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \mathbf{x}_{1|t}^{\theta}(\mathbf{x}) \right\| = \mathcal{O}(h^2), \tag{64}$$

holds and thus

$$\left\| \delta_{\mathbf{x}} \Phi_{t,1}^{\theta}(\mathbf{x}) - \delta_{\mathbf{x}}^{\mathcal{G}} \Phi_{t,1}^{\theta}(\mathbf{x}) \right\| = \mathcal{O}(h^2). \tag{65}$$

□

C.6 Proof of Theorem 5.4

We restate Theorem 5.4 below.

Theorem 5.4 (Greedy convergence). *For affine probability paths, if there exists a sequence of states $\mathbf{x}_t^{(n)}$ at time t such that it converges to the locally optimal solution $\mathbf{x}_{1|t}^{\theta}(\mathbf{x}_t^{(n)}) \rightarrow \mathbf{x}_1^*$. Then the solution, $\Phi_{1|t}^{\theta}(\mathbf{x}_t^{(n)})$, converges to a neighborhood of size $\mathcal{O}(h^2)$ centered at $\mathbf{x}_1^*$.*

Proof. By Assumption B.2, we can take a $(k-1)$ -th order Taylor expansion around γ_t of the flow in Equation (8) to obtain

$$\begin{aligned}
\Phi_{1|t}^{\theta}(\mathbf{x}_t) &= \frac{\sigma_1}{\sigma_t} \mathbf{x}_t + \sigma_1 \int_{\gamma_t}^{\gamma_1} \sum_{n=0}^{k-1} \frac{d^n}{d\gamma^n} \left[\mathbf{x}_{1|\gamma}^{\theta}(\mathbf{x}_{\gamma}) \right]_{\gamma=\gamma_t} \frac{(\gamma - \gamma_t)^n}{n!} d\gamma + \mathcal{O}(h^{k+1}), \\
&= \frac{\sigma_1}{\sigma_t} \mathbf{x}_t + \sigma_1 \sum_{n=0}^{k-1} \frac{d^n}{d\gamma^n} \left[\mathbf{x}_{1|\gamma}^{\theta}(\mathbf{x}_{\gamma}) \right]_{\gamma=\gamma_t} \int_{\gamma_t}^{\gamma_1} \frac{(\gamma - \gamma_t)^n}{n!} d\gamma + \mathcal{O}(h^{k+1}), \\
&= \frac{\sigma_1}{\sigma_t} \mathbf{x}_t + \sigma_1 \sum_{n=0}^{k-1} \frac{d^n}{d\gamma^n} \left[\mathbf{x}_{1|\gamma}^{\theta}(\mathbf{x}_{\gamma}) \right]_{\gamma=\gamma_t} \frac{h^{n+1}}{(n+1)!} + \mathcal{O}(h^{k+1}),
\end{aligned} \tag{66}$$

where $h := \gamma_1 - \gamma_t$ is the stepsize. Let $k = 1$, then we have:

$$\Phi_{1|t}^{\theta}(\mathbf{x}_t) = \frac{\sigma_1}{\sigma_t} \mathbf{x}_t + \sigma_1 \hat{\mathbf{x}}_{1|t}(\mathbf{x}_t) h + \mathcal{O}(h^2), \tag{67}$$

$$= \frac{\sigma_1}{\sigma_t} \mathbf{x}_t + \left(\alpha_1 - \frac{\sigma_1 \alpha_t}{\sigma_t} \right) \hat{\mathbf{x}}_{1|t}(\mathbf{x}_t) + \mathcal{O}(h^2). \tag{68}$$

By definition $\sigma_1 = 0$ and $\alpha_1 = 1$, then

$$\Phi_{1|t}^\theta(\mathbf{x}_t) = \hat{\mathbf{x}}_{1|t}(\mathbf{x}_t) + \mathcal{O}(h^2), \quad (69)$$

which is equivalent to

$$\left\| \Phi_{1|t}^\theta(\mathbf{x}_t) - \hat{\mathbf{x}}_{1|t}(\mathbf{x}_t) \right\| \leq C_1 h^2, \quad (70)$$

for some constant $C_1 > 0$. Since $\mathbf{x}_{1|t}^\theta(\mathbf{x}_t^{(n)}) \rightarrow \mathbf{x}_1^*$ we know that for any $\epsilon > 0$ there exists some $n \geq N$ such that $\|\mathbf{x}_1^* - \mathbf{x}_{1|t}^\theta(\mathbf{x}_t^{(n)})\| < \epsilon$. Thus,

$$\left\| \Phi_{1|t}^\theta(\mathbf{x}_t^{(n)}) - \mathbf{x}_1^* \right\| \leq \left\| \Phi_{1|t}^\theta(\mathbf{x}_t^{(n)}) - \mathbf{x}_{1|t}^\theta(\mathbf{x}_t^{(n)}) \right\| + \underbrace{\left\| \mathbf{x}_1^* - \mathbf{x}_{1|t}^\theta(\mathbf{x}_t^{(n)}) \right\|}_{:= C_2} < \epsilon + C_1 h^2. \quad (71)$$

Therefore, $\Phi_{1|t}(\mathbf{x}_t^{(n)})$ converges to a point inside a neighborhood centered at $\mathbf{x}_1^*$ with radius $\mathcal{O}(h^2)$. $\square$

D Beyond Euler

In this section we provide the full proofs and derivations for Section 6 in the main paper.

D.1 Proof of Theorem 6.1

Before showing Theorem 6.1 we show a more general version below.

Theorem D.1 (Local truncation error of discretize-then-optimize gradients). *Let Φ be an explicit Runge-Kutta solver of order $\alpha > 0$ to the ODE*

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \frac{d\mathbf{x}}{dt}(t) = \mathbf{u}_\theta(t, \mathbf{x}(t)), \quad (72)$$

on $[0, T]$ which satisfies the regularity conditions for the Picard-Lindelöf theorem. Let $\Phi_{s,t}^\theta(\mathbf{x})$ denote the flow from s to t , for any $s, t \in [0, T]$ admitted by the ODE. Then,

$$\left\| \nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \Phi_{s,t}(\mathbf{x}) \right\| = \mathcal{O}(h^{\alpha+1}). \quad (73)$$

Proof. Consider an explicit k -stage Runge-Kutta method given by

$$\mathbf{u}_{n,j} = \mathbf{u}_\theta \left(t_n + c_j h, \mathbf{x}_n + h \sum_{i=1}^j a_{j,i} \mathbf{u}_{n,i} \right), \quad j = 1, 2, \dots, k \quad (74)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h \sum_{j=1}^k b_j \mathbf{u}_{n,j}, \quad (75)$$

where $a_{j,i}, b_j, c_j$ are all given via the *Butcher Tableau* (Stewart 2022, Section 6.1.4). Now, we consider a single step from time s to time t with initial value $\mathbf{x}$ and step size $h := t - s$. Then, the gradient is

$$\begin{aligned} \nabla_{\mathbf{x}} \Phi_{s,t}(\mathbf{x}) &= \nabla_{\mathbf{x}} \mathbf{x} + h \sum_{j=1}^k b_j \nabla_{\mathbf{x}} \mathbf{u}_\theta \left(s + c_j h, \mathbf{x} + h \sum_{i=1}^j a_{j,i} \mathbf{u}_i \right), \\ &= \mathbf{I} + h \sum_{j=1}^k b_j \left[\nabla_{\hat{\mathbf{x}}_j} \mathbf{u}_\theta(s + c_j h, \hat{\mathbf{x}}_j) \left(\mathbf{I} + h \sum_{i=1}^j a_{j,i} \nabla_{\mathbf{x}} \mathbf{u}_i \right) \right], \end{aligned} \quad (76)$$

where we let

$$\hat{\mathbf{x}}_j = \mathbf{x} + h \sum_{i=1}^j a_{j,i} \mathbf{u}_i. \quad (77)$$

Next, recall Lemma C.2 which gives the following ODE

$$\mathbf{J}_s(s) = \mathbf{I}, \quad \frac{d\mathbf{J}_s}{dt}(t) = \nabla_{\mathbf{x}} \mathbf{u}_\theta(t, \Phi_{s,t}(\mathbf{x})) \mathbf{J}_s(t). \quad (78)$$

Next, we augmented the ODE above with the underlying ODE for the solution state, $\dot{\mathbf{x}}(t) = \mathbf{u}_\theta(t, \mathbf{x}(t))$. We now apply the same Runge-Kutta solver to this augmented ODE for the Jacobian matrices which yields

$$\mathbf{U}_j = \mathbf{I} + h \sum_{j=1}^k b_j \left[\nabla_{\hat{\mathbf{x}}_j} \mathbf{u}_\theta(s + c_j h, \hat{\mathbf{x}}_j) \left(\mathbf{I} + h \sum_{i=1}^j a_{j,i} \nabla_{\mathbf{x}} \mathbf{u}_i \right) \right]. \quad (79)$$

Clearly, Equation (79) and Equation (105) are equivalent. Now as the underlying numerical solver has local truncation error $\mathcal{O}(h^{\alpha+1})$ we find that

$$\|\nabla_{\mathbf{x}} \Phi_{s,t}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \Phi_{s,t}(\mathbf{x})\| = \mathcal{O}(h^{\alpha+1}). \quad (80)$$

□

Remark D.1. This result is intuitive as differentiation is a linear operator. However simple, we believe the insight is useful on the discussion of using DTO/OTD/posterior methods for guidance and thus include it here.

Remark D.2. Theorem D.1 shows that DTO and OTD are really just two sides of the same coin and that one of the main differences is the choice of end points when discretizing.

Remark D.3. Onken and Ruthotto (2020, Appendix A) made similar observations; however, it is for only of the case of Euler.

Theorem 6.1 (Truncation error of single-step gradients). *Let Φ be an explicit Runge-Kutta solver of order $\alpha > 0$ of a flow model with flow $\Phi_{s,t}^\theta(\mathbf{x})$. Then for any $t \in [0, 1]$,*

$$\|\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \Phi_{t,1}(\mathbf{x})\| = \mathcal{O}(h^{\alpha+1}), \quad (15)$$

where $h = 1 - t$.

Proof. This follows as a corollary of Theorem D.1. □

Corollary D.1.1 (Convergence of a α -th order posterior gradient). *For affine probability paths, if there exists a sequence of states $\mathbf{x}_t^{(n)}$ at time t such that it converges to the locally optimal solution $\Phi_{t,1}^\theta(\mathbf{x}_t^{(n)}) \rightarrow \mathbf{x}_1^*$. Then solution, $\Phi_{1|t}^\theta(\mathbf{x}_t^{(n)})$, converges to a neighborhood of size $\mathcal{O}(h^{\alpha+1})$ centered at $\mathbf{x}_1^*$.*

Proof. This follows as a straightforward derivation from Theorem D.1. □

Corollary D.1.2 (Dynamics of α -th order posterior gradient). *Consider the standard affine Gaussian probability paths model trained to zero loss. Let Φ be an explicit Runge-Kutta solver of order $\alpha > 0$ of a flow model with flow $\Phi_{s,t}^\theta(\mathbf{x})$. The Gateaux differential of $\mathbf{x}$ at some time $t \in [0, 1]$ in the direction of the gradient $\nabla_{\mathbf{x}} \mathcal{L}(\Phi_{t,1}(\mathbf{x}))$ is given by*

$$\delta_{\mathbf{x}}^\Phi(\mathbf{x}) = -\nabla_{\mathbf{x}} \Phi_{t,1}^\theta(\mathbf{x}) \nabla_{\mathbf{x}} \Phi_{t,1}(\mathbf{x})^\top \nabla_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1). \quad (81)$$

Proof. This follows straightforwardly from Proposition 5.2 and Theorem 6.1. □

D.2 A useful reparameterization of the flow model

We present a useful reparameterization of the flow model, which is a parallel result to Proposition 4.1.

Proposition D.2 (Reparameterized for the target prediction model of affine probability paths). *The ODE governed by the vector field in Equation (18) can be reparameterized as*

$$\frac{d\mathbf{y}_\gamma}{d\gamma} = \sigma_0 \mathbf{x}_{1|\zeta}^\theta \left(\frac{\sigma_\gamma}{\sigma_0} \mathbf{y}_\zeta \right), \quad (82)$$

where $\mathbf{y}_t = \frac{\sigma_0}{\sigma_t} \mathbf{x}_t$.

Proof. The ODE governed by the vector field in Equation (18) can be written as

$$\frac{d\mathbf{x}_t}{dt} = a_t \mathbf{x}_t + b_t \mathbf{x}_{1|t}^\theta(\mathbf{x}_t). \quad (83)$$

Now we can use the technique of exponential integrators to rewrite the ODE as

$$\frac{d}{dt} \left[e^{\int_0^t -a_u du} \mathbf{x}_t \right] = e^{\int_0^t -a_u du} b_t \mathbf{x}_{1|t}^\theta(\mathbf{x}_t). \quad (84)$$

The exponential term can be simplified to

$$e^{\int_0^t -a_u du} = \frac{\sigma_0}{\sigma_t}. \quad (85)$$

We introduce a *change-of-variables*, $\mathbf{y}_t = \frac{\sigma_0}{\sigma_t} \mathbf{x}_t$. Thus, the ODE becomes

$$\frac{d\mathbf{y}_t}{dt} = \frac{\sigma_0}{\sigma_t} b_t \mathbf{x}_{1|t}^\theta \left(\frac{\sigma_t}{\sigma_0} \mathbf{y}_t \right). \quad (86)$$

Next, recall that $b_t/\sigma_t = \dot{\gamma}_t$ (cf. Equation (27)) which enables a change of integration variable:

$$\frac{d\mathbf{y}_\gamma}{d\gamma} = \sigma_0 \mathbf{x}_{1|\gamma}^\theta \left(\frac{\sigma_\gamma}{\sigma_0} \mathbf{y}_\gamma \right). \quad (87)$$

□

Remark D.4. Recall that, often, for affine probability paths we let $\sigma_0 = 1$, further simplifying Proposition D.2 to

$$\frac{d\mathbf{y}_\gamma}{d\gamma} = \mathbf{x}_{1|\gamma}^\theta(\sigma_\gamma \mathbf{y}_\gamma). \quad (88)$$

Remark D.5. Proposition D.2 is a tangential result to the prior result of Pan, Liew, et al. (2024, Equation (11)) which was for diffusion models and was developed w.r.t. the source prediction model rather than the target prediction model and was solved in reverse-time.¹⁰

This parameterization in Proposition D.2 can be combined with Theorem D.1 to construct a DTO approximation of the gradient with truncation error $(\gamma_t - \gamma_s)^{\alpha+1}$.

E Notes on using OTD in practice

While the OTD approach has become quite popular after the work of R. T. Chen et al. (2018), several later works have noticed several key issues that we wish to note for ML practitioners.

Recall our prototypical neural ODE (or flow model) of the form

$$\frac{d\mathbf{x}}{dt}(t) = \mathbf{u}_\theta(t, \mathbf{x}(t)), \quad (89)$$

and assume it is defined on the interval $[0, T]$ and the flow model satisfies the usual regularity conditions. Then, the continuous adjoint equations (Kidger 2022, Theorem 5.2) are:

$$\begin{aligned} \mathbf{a}_\mathbf{x}(T) &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_T}, & \frac{d\mathbf{a}_\mathbf{x}}{dt}(t) &= -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \mathbf{u}_\theta}{\partial \mathbf{x}}(t, \mathbf{x}(t)), \\ \mathbf{a}_\theta(T) &= \mathbf{0}, & \frac{d\mathbf{a}_\theta}{dt}(t) &= -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \mathbf{u}_\theta}{\partial \theta}(t, \mathbf{x}(t)), \end{aligned} \quad (90)$$

where $\mathbf{a}_\mathbf{x}(t) := \partial \mathcal{L} / \partial \mathbf{x}(t)$ and $\mathbf{a}_\theta(0) := \partial \mathcal{L} / \partial \theta$.

¹⁰Technically forward-time due to the conventions of diffusion models.

Table 4: Comparison of different strategies for performing backpropagation through flow models. For the complexity analysis n denotes the number of discretization steps and d the dimensionality of the state. Note, for accuracy we mean there are no truncation errors. Note that whilst in general the stability of reversible solvers is quite poor, there are *some* solvers which have a non-trivial region of stability.

Method	Time	Memory	Accurate gradients	Stability
DTO	$\mathcal{O}(n)$	$\mathcal{O}(nd^2)$	✓	-
DTO + recursive checkpointing	$\mathcal{O}(n \log n)$	$\mathcal{O}(d^2 \log n)$	✓	-
OTD + stored trajectory	$\mathcal{O}(n)$	$\mathcal{O}(nd + d^2)$	✓	-
OTD + reversible solver	$\mathcal{O}(n)$	$\mathcal{O}(d^2)$	✓	?
OTD	$\mathcal{O}(n)$	$\mathcal{O}(d^2)$	✗	✗

Truncation errors. One area of concern is the potential mismatch between the forward trajectory $\{\mathbf{x}_{t_i}\}_{i=1}^N$ and the backward trajectory $\{\tilde{\mathbf{x}}_{t_i}\}_{i=1}^N$ when performing the backwards solve. *E.g.*, consider an explicit Euler scheme

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + (t_{i+1} - t_i)\mathbf{u}_\theta(t_i, \mathbf{x}_{t_i}). \quad (91)$$

The same scheme when applied to solving the backward trajectory would yield,

$$\tilde{\mathbf{x}}_{t_i} = \tilde{\mathbf{x}}_{t_{i+1}} + (t_i - t_{i+1})\mathbf{u}_\theta(t_{i+1}, \tilde{\mathbf{x}}_{t_{i+1}}). \quad (92)$$

Clearly, there is no guarantee that these two trajectories match during the forward and backward solve introducing a source of error. One potential solution is to use an *algebraically reversible solver* (see Blasingame and C. Liu 2025; Kidger et al. 2021; McCallum and Foster 2024) which guarantees that the forward and backward trajectory match *perfectly*. Another option is to store the forward trajectory $\{\mathbf{x}_{t_i}\}_{i=1}^N$ in memory and use *interpolated adjoints* if the backward timesteps do not perfectly align with the forward timesteps (see S. Kim et al. 2021).

Stability concerns. Consider the simple ODE, $\dot{y}(t) = \lambda y(t)$ defined on $t \in [0, T]$ with $y(0) = y_0$ and $\lambda < 0$. Clearly, most ODE solvers with a non-trivial region of stability (see Harier and Wanner 2002, Definition 2.1) will solve this ODE without an issue, as the errors will decrease exponentially with $\lambda < 0$. However, in the backwards in time solve from $y(T)$ the errors will *grow exponentially*. It can be shown that the adjoint state suffers from similar stability issues. The local behavior of a differential equation is described through the eigenvalues of the Jacobian of the vector field (see Butcher 2016). For $\mathbf{x}_t$ this is given by $\frac{\partial \mathbf{u}_\theta}{\partial \mathbf{x}}$ and for $\mathbf{a}_x$ this is given by

$$\frac{\partial}{\partial \mathbf{a}_x} \left(-\mathbf{a}_x(t)^\top \frac{\partial \mathbf{u}_\theta}{\partial \mathbf{x}}(t, \mathbf{x}(t)) \right) = -\frac{\partial \mathbf{u}_\theta}{\partial \mathbf{x}}(t, \mathbf{x}(t)). \quad (93)$$

Clearly, the Jacobians for $\mathbf{a}_x$ and $\mathbf{x}_t$ solved in reverse-time are identical, meaning the stability of the backward solve is pushed onto the solve for the adjoint state (see Kidger 2022, Section 5.1.2.4) for more details. Reversible solvers eliminate truncation errors, but tend to suffer from poor stability, *e.g.*, the region of stability for reversible Heun applied to neural ODEs is the complex interval $[-i, i]$ (Kidger et al. 2021). Recent work by McCallum and Foster (2024), however, has shown a strategy for constructing reversible solvers with a non-trivial region of stability.

Recommendations. In light of these concerns we propose we consider to be best practices for deciding what scheme to use.

Generally, the best choice is DTO when memory allows as it is the most *accurate* in terms of the forward discretization. If memory is an issue then using a clever checkpointing scheme (Griewank 1992; Griewank and Walther 2000; Stumm and Walther 2010) can help alleviate such issues in exchange for additional compute time. The recursive checkpointing strategy in combination with DTO is actually the default (and recommended) implementation in the `DiffraX` library. Alternatively, one could store the forward trajectory in memory and then apply the OTD scheme on these stored states (not activations). This strategy of caching the forward trajectory is quite popular and was used by Blasingame and C. Liu (2024a) and Domingo-Enrich et al. (2025) in practice when solving the continuous adjoint equations. Another option is to use an algebraically reversible solver in

conjunction with OTD. Lastly, one could use vanilla OTD, which we should mention can actually work reasonably well depending on the application despite the concerns listed above.

In Table 4 we summarize the discussion of this section and hope it is helpful to the reader.

F On control signal optimization

Rather than optimizing the trajectory of the solution or the initial condition, several works (X. Liu et al. 2023; L. Wang et al. 2025) have explored the guidance from the perspective of optimal control (Kirk 2004). In essence this technique first injects an additional control signal, $\mathbf{z} \in \mathcal{C}^1(\mathbb{R}; \mathbb{R}^d)$, to the vector field, $\mathbf{u}_t^\theta$, such that

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{u}_t^\theta(\mathbf{x}_t) + \mathbf{z}(t). \quad (94)$$

Thus, instead of optimizing $\{\mathbf{x}_t\}_{t \in [0, T]}$ directly, this control signal can instead be optimized, serving as one of the key insights in (X. Liu et al. 2023; L. Wang et al. 2025). *I.e.*, suppose we have a neural ODE with vector field $\mathbf{u}_t^\theta(\mathbf{x})$, then we can write the optimization problem as

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathcal{L}(\mathbf{x}_T) + \lambda \int_0^T \|\mathbf{z}(t)\| dt, \\ \text{s.t.} \quad & \mathbf{x}_T = \mathbf{x}_0 + \int_0^T \mathbf{u}_t^\theta(\mathbf{x}_t) + \mathbf{z}(t) dt. \end{aligned} \quad (95)$$

The next natural question then is to ask about the behavior of a greedy strategy applied to $\mathbf{z}(t)$. To simplify the analysis, we now consider a control signal applied to the posterior model $\mathbf{x}_{1|t}^\theta$ such that it is replaced by $\mathbf{x}_{1|t}^\theta(\mathbf{x}_t) + \mathbf{z}(t)$ which amounts to simply rescaling $\mathbf{z}(t)$ from Equation (94) with b_t . From this construction, it should be clear that the greedy gradient for the control signal is merely $\nabla_{\tilde{\mathbf{x}}_1} \mathcal{L}(\tilde{\mathbf{x}}_1)$. If using the original formulation where the control signal is applied to the vector field, rather than the denoiser, the gradient is simply scaled by a weighting function dependent on time. Note that this approach is similar to the greedy approach taken by Blasingame and C. Liu (2024b); however, they inject the control signal to the source prediction model rather than the target prediction model.

F.1 Continuous adjoint equations for control signals

We can model the gradient for this signal by augmenting the continuous adjoint equations with the adjoint state $\mathbf{a}_z(t) := \partial \mathcal{L} / \partial \mathbf{z}(t)$. In Theorem F.1 we show that this gradient is simply an integral of the adjoint state $\mathbf{a}_x(t)$.

Theorem F.1 (Continuous adjoint equations for the control term). *Let $\mathbf{u}_t^\theta \in \mathcal{C}^{1,1}([0, T] \times \mathbb{R}^{d_x}; \mathbb{R}^{d_x})$ be a parameterization of some time-dependent vector field of a neural ODE that is Lipschitz continuous in its second argument, and let $\mathbf{z} \in \mathcal{C}^1([0, 1]; \mathbb{R}^d)$ be an additional control signal such that the new dynamics are given by Equation (94). Let $\mathbf{a}_z(t) := \partial \mathcal{L} / \partial \mathbf{z}(t)$, then*

$$\mathbf{a}_z(t) = - \int_T^t \mathbf{a}_x(s) ds. \quad (96)$$

Our proof follows the structure of the modern proof of Pontryagin’s original result (Pontryagin et al. 1963) presented by (R. T. Chen et al. 2018); and is similar to the form used by Blasingame and C. Liu (2024a, Theorem 2.2).

Proof. For notational clarity, we use the notation $\mathbf{x}(t) = \mathbf{x}_t$. We define the augmented state on $[0, T]$ as

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} (t) = \mathbf{f}_{\text{aug}} = \begin{bmatrix} \mathbf{u}_\theta(t, \mathbf{x}(t)) + \mathbf{z}(t) \\ \frac{d\mathbf{z}}{dt}(t) \end{bmatrix}, \quad (97)$$

and the augmented adjoint state as

$$\mathbf{a}_{\text{aug}}(t) := \begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_z \end{bmatrix} (t). \quad (98)$$

The Jacobian of $\mathbf{f}_{\text{aug}}$ has form

$$\frac{\partial \mathbf{f}_{\text{aug}}}{\partial [\mathbf{x}, \mathbf{z}]} = \begin{bmatrix} \frac{\partial \mathbf{u}_\theta(t, \mathbf{x}(t))}{\partial \mathbf{x}} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (99)$$

The evolution of the adjoint state is given by

$$\frac{d\mathbf{a}_{\text{aug}}}{dt}(t) = -[\mathbf{a}_x \quad \mathbf{a}_z](t) \frac{\partial \mathbf{f}_{\text{aug}}}{\partial [\mathbf{x}, \mathbf{z}]}(t). \quad (100)$$

Therefore, $\mathbf{a}_u(t)$ evolves with

$$\mathbf{a}_u(T) = \mathbf{0}, \quad \frac{d\mathbf{a}_u}{dt}(t) = -\mathbf{a}_x(t), \quad (101)$$

thereby finishing the proof. $\square$

G Implementation details

We discuss how to implement the greedy strategy.

G.1 The construction of the greedy guidance schemes

Recall that the general Butcher tableau for a k -stage explicit RK scheme (Stewart 2022, Section 6.1.4) is written as

$$\begin{array}{c|cccccc} c_1 & & & & & & \\ c_2 & a_{2,1} & & & & & \\ c_3 & a_{3,1} & a_{3,2} & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ c_k & a_{k,1} & a_{k,2} & \cdots & a_{(k-1),k} & & \\ \hline & b_1 & b_2 & \cdots & b_{k-1} & b_k & \end{array} = \frac{c}{b}. \quad (102)$$

Thus a single-step is given by

$$\mathbf{u}_{n,j} = \mathbf{u}_\theta \left(t_n + c_j h, \mathbf{x}_n + h \sum_{i=1}^j a_{j,i} \mathbf{u}_{n,i} \right), \quad j = 1, 2, \dots, k \quad (103)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h \sum_{j=1}^k b_j \mathbf{u}_{n,j}, \quad (104)$$

where $a_{j,i}, b_j, c_j$ are all given via the *Butcher Tableau* (Stewart 2022, Section 6.1.4). Now, we consider a single step from time s to time t with initial value $\mathbf{x}$ and step size $h := t - s$. Then, the gradient is

$$\begin{aligned} \nabla_{\mathbf{x}} \Phi_{s,t}(\mathbf{x}) &= \nabla_{\mathbf{x}} \mathbf{x} + h \sum_{j=1}^k b_j \nabla_{\mathbf{x}} \mathbf{u}_\theta \left(s + c_j h, \mathbf{x} + h \sum_{i=1}^j a_{j,i} \mathbf{u}_i \right), \\ &= \mathbf{I} + h \sum_{j=1}^k b_j \left[\nabla_{\hat{\mathbf{x}}_j} \mathbf{u}_\theta(s + c_j h, \hat{\mathbf{x}}_j) \left(\mathbf{I} + h \sum_{i=1}^j a_{j,i} \nabla_{\mathbf{x}} \mathbf{u}_i \right) \right], \end{aligned} \quad (105)$$

where we let

$$\hat{\mathbf{x}}_j = \mathbf{x} + h \sum_{i=1}^j a_{j,i} \mathbf{u}_i. \quad (106)$$

Which can easily be found through standard reverse-mode autodifferentiation frameworks; likewise, the gradients for multiple Euler steps can be found.

G.2 A PyTorch pseudocode illustration

Below in Code G.1 we provide an example PyTorch implementation of the greedy guidance strategy with an Euler scheme for sampling the main ODE.

Code G.1: Example implementation of greedy guidance

```
# posterior step
def posterior_step(t, x, n_steps=1)
    tlist = [0, 1]
    dt = tlist[1] - t

    if method == 'euler':
        # Euler step
        x = x + model(t, x) * dt
    elif method == 'midpoint':
        # Midpoint step
        x = x + dt * model(t + 0.5 * dt, x + 0.5 * dt * model(t, x))
    elif method == 'ralston3':
        # Ralston's third order
        k1 = model(t, x)
        k2 = model(t + 0.5 * dt, x + 0.5 * dt * k1)
        k3 = model(t + 0.75 * dt, x + 0.75 * dt * k2)

        x = x + dt * (2/9 * k1 + 1/3 * k2 + 4/9 * k3)
    elif method == 'heun3':
        # Heun's third order
        k1 = model(t, x)
        k2 = model(t + 1/3 * dt, x + 1/3 * dt * k1)
        k3 = model(t + 2/3 * dt, x + 2/3 * dt * k2)

        x = x + dt * (1/4 * k1 + 3/4 * k3)
    elif method == 'rk4':
        # RK-4 step
        k1 = model(t, x)
        k2 = model(t + 0.5*dt, x + 0.5*dt*k1)
        k3 = model(t + 0.5*dt, x + 0.5*dt*k2)
        k4 = model(t + dt, x + dt*k3)

        x = x + dt * (1/6 * k1 + 1/3 * k2 + 1/3 * k3 + 1/6 * k4)
    elif method == 'multiple_euler':
        # k Euler steps
        Dt = tlist[1] - t
        dt = Dt / n_steps
        ts = torch.linspace(t, tlist[1], n_steps + 1)

        for t in ts:
            x = x + model(t, x) * dt

    return x

# posterior guidance pseudocode
# assumed dt and loss are defined

xt = x0

for t in timesteps:
    # quick fix
    xt_opt = xt.detach().clone().requires_grad_(True)

    if optim == 'sgd':
        optimizer = torch.optim.SGD([xt_opt], lr=lr)
    else:
        optimizer = torch.optim.LBFGS([xt_opt], max_iter=max_iter, lr=lr, line_search_fn='strong_wolfe')

    for step in range(opt_steps):
        optimizer.zero_grad()

        xihat = posterior_step(t, xt_opt)

        loss(xihat).mean().backward()
        optimizer.step()

    with torch.no_grad():
        xt = xt_opt + model(t, xt_opt) * dt

return xt
```

H A brief introduction to inverse problems

Inverse problems cover a large class of scientific problems (Chung, J. Kim, et al. 2023) that encompass scenarios where a partial measurement $\mathbf{y}$ is made of $\mathbf{x}$. When the mapping $\mathbf{x} \mapsto \mathbf{y}$ is not an injection,

recovering $\mathbf{x}$ from $\mathbf{y}$ becomes an ill-posed inverse problem. Generally, the relationship between the underlying sample $\mathbf{x}$ and the measurement $\mathbf{y}$ is given by

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \boldsymbol{\eta}, \quad \mathbf{y}, \boldsymbol{\eta} \in \mathbb{R}^{d_y}, \mathbf{x} \in \mathbb{R}^{d_x}, \quad (107)$$

where $\mathcal{A} : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$ is the forward measurement operator and $\boldsymbol{\eta} \sim (0, \beta_{\mathbf{y}}^2 \mathbf{I})$ is the measurement noise.

The inverse problem then is to find $p(\mathbf{x}|\mathbf{y})$.

More details on these types of problems can be found in Chung, J. Kim, et al. (2023), Moufad et al. (2025), and B. Zhang et al. (2025).

H.1 Inverse problems and diffusion models

Recall that the ODE formulation of diffusion models is just a particular type of affine Gaussian probability path (Lipman, Havasi, et al. 2024). Following the conventions of the EDM model (Karras, Aittala, et al. 2022) we write this ODE formulation, known in the literature as the *probability flow ODE*, below in

$$d\mathbf{x}_t = -\dot{\sigma}_t \sigma_t \nabla_{\mathbf{x}_t} \log p(\sigma_t, \mathbf{x}_t) dt, \quad (108)$$

where $p(\sigma_t, \mathbf{x}_t)$ is the joint distribution of $\mathbf{x}_t$ at noise level σ_t .¹¹ *N.B.*, for diffusion models dt is a *negative* timestep and we integrate in reverse-time from T to 0. These models are also called *score-based generative models* due to learning the score function $\nabla_{\mathbf{x}_t} \log p(\sigma_t, \mathbf{x}_t)$.

One of the insights of Chung, J. Kim, et al. (2023) and Y. Song, Sohl-Dickstein, et al. (2021) is to apply Bayes' theorem for inverse problems to score-based generative models, *i.e.*,

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \quad (109)$$

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}). \quad (110)$$

Adapting this for diffusion models, assuming $\mathcal{A}$ is defined on $\mathbf{x}_0$ (the output), we have

$$\nabla_{\mathbf{x}_t} \log p(\sigma_t, \mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\sigma_t, \mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t, \sigma_t). \quad (111)$$

The unconditional score term is the regular score function learned by diffusion models and thus is *appropriately* learned; however, the other term is much more difficult to work with. The approach of Chung, J. Kim, et al. (2023) is to use an approximation of

$$p(\mathbf{y}|\mathbf{x}_t, \sigma_t) = \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[p(\mathbf{y}|\mathbf{x}_0, \sigma_0)], \quad (112)$$

via Tweedie's formula (Stein 1981) to write

$$p(\mathbf{y}|\mathbf{x}_t, \sigma_t) \approx p(\mathbf{y}|\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t], \sigma_0). \quad (113)$$

The approximation error can be quantified by the Jensen gap (Chung, J. Kim, et al. 2023, Theorem 1).

I Experimental details

We provide additional details of the experiments performed in Section 7. *N.B.*, for all experiments we used fixed random seeds between the different software components to ensure a fair comparison.

I.1 Inverse image problems

Inverse problems. The inverse problems are implemented in the same way as in B. Zhang et al. (2025). We reiterate some of the important settings below. For Gaussian and motion deblurring we made use of kernels of size 61×61 with standard deviations of 3.0 and 0.5 respectively. The box inpainting task makes use of a random box of size 128×128 to mask the original images, while the random inpainting task randomly masks each pixel with a probability of 70% following (B. Song et al. 2024). The measure for the high dynamic range reconstruction problem is defined as

$$\mathbf{y} \sim \mathcal{N}(\text{clip}(\alpha \mathbf{x}_0, -1, 1), \beta_{\mathbf{y}}^2 \mathbf{I}), \quad (114)$$

with $\alpha = 2$.

¹¹This σ_t is not the same as the σ_t from the scheduler (α_t, σ_t) used in the main paper.

Diffusion model. We make use of the pre-trained diffusion model from Chung, J. Kim, et al. (2023), trained on the FFHQ 256×256 dataset. We focus on the probability flow ODE formulation popularized by Karras, Aittala, et al. (2022) known as EDM described as

$$d\mathbf{x}_t = -\dot{\sigma}_t \sigma_t \nabla_{\mathbf{x}_t} \log p(\sigma_t, \mathbf{x}_t) dt. \quad (115)$$

Following Ben-Hamu et al. (2024), we employ a midpoint scheme to solve this ODE in *reverse-time* with $N = 20$ steps. We use the noise schedule $\sigma_t = t$ which means $\dot{\sigma}_t = 1$. The discretized noise schedule $\{\sigma_n\}_{n=1}^N$ is given by the following polynomial interpolation

$$\sigma_n = \left(\sigma_{\max}^\rho + \frac{n}{N-1} \left(\sigma_{\min}^\rho - \sigma_{\max}^\rho \right) \right)^\rho. \quad (116)$$

We use $\rho = 7$, $T = \sigma_{\max} = 100$, and $\epsilon = \sigma_{\min} = 0.01$ for all experiments and integrate over $[\epsilon, T]$. *N.B.*, truncating the integration domain at ϵ rather than 0 is quite common in diffusion models (Y. Song, Dhariwal, et al. 2023).

Hyperparameters. Unlike previous works (B. Zhang et al. 2025) we did not adjust the hyperparameters per task and left them the same throughout. The learning rate was set at $\eta = 1$ for all experiments, and we performed $n_{\text{opt}} = 50$ optimization steps with the stock implementation of `torch.optim.SGD` method for each step of the ODE solve. We set $\beta_y = 0.05$ for all tasks.

Ablation study. For the ablation study in Table 2 we used the L-BGFS optimizer over the standard SGD optimizer used in the main experiments (for the greedy guidance runs). For full DTO we used SGD as it provided better performance over L-BGFS in that scenario. *N.B.*, due to compute limitations we couldn’t run DTO for step sizes larger than 8. Importantly, we fix the maximum number of optimization steps between the greedy and DTO strategies; for greedy we take 5 optimization step per step in the ODE solver, so a 100 in total. Likewise, for DTO we take a 100 optimization steps in total.

I.2 Molecule generation for QM9

We follow the experimental methodology taken in previous work (Ben-Hamu et al. 2024; L. Wang et al. 2025) and follow the conditional generation pipeline used by Hooeboom et al. (2022). An equivariant *graph neural network* (GNN) was trained for each property on half of the QM9 dataset, serving as a classifier—this model was then used as a guidance function during the experiments. The EquiFM (Y. Song, Gong, et al. 2023) model was trained on the whole QM9 training set and was used as the underlying flow model for the experiments. Following L. Wang et al. (2025), the test time properties were sampled from the whole training set; in contrast to Ben-Hamu et al. (2024).

Following Ben-Hamu et al. (2024) we used the L-BFGS algorithm (D. C. Liu and Nocedal 1989) with 5 optimizer steps and 5 inner steps with a linear search, in particular we used the stock PyTorch implementation `torch.optim.LBFGS`. For the DTO experiment we used a learning rate of $\eta = 1$. We tried this for the posterior guidance experiments but encountered severe instability. We found that a learning rate of $\eta = 0.001$ seemed to work better.

Recall that Proposition 5.2 states that the greedy gradient is scaled by the covariance projection. This effect is lessened as $t \rightarrow 1$, thus in later timesteps the greedy gradient is more likely to push samples off the data manifold. We observed this, with exploding losses even at small learning rates. To remedy this, we took inspiration from other works (Chung, J. Kim, et al. 2023; Moufad et al. 2025; Yu et al. 2023) and annealed the learning rate. We chose the following simple scheduler:

$$\eta_t = \begin{cases} \eta(1-t) & t > 0.5 \\ 0 & t \leq 0.5 \end{cases}, \quad (117)$$

where $\eta = 0.001$ is the base learning rate.

Runge-Kutta 4. Additionally, we ran some experiments using RK4 but ran into insurmountable stability issues. Recall that RK4 is given by

$$\mathbf{k}_1 = \mathbf{u}_\theta(t_n, \mathbf{x}_n), \quad (118)$$

$$\mathbf{k}_2 = \mathbf{u}_\theta\left(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}\mathbf{k}_1\right), \quad (119)$$

$$\mathbf{k}_3 = \mathbf{u}_\theta\left(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}\mathbf{k}_2\right), \quad (120)$$

$$\mathbf{k}_4 = \mathbf{u}_\theta(t_n + h, \mathbf{x}_n + h\mathbf{k}_3), \quad (121)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \quad (122)$$

Using the step size $h = 1 - t$ we encountered large stability issues with the $\mathbf{k}_4$ term due to being evaluated at the endpoint of the flow model trajectory. We tried a mixed-solver scheme where we would start with Euler and then switch to RK4, but that did not help. We also tried the common diffusion trick of truncating the time interval to $[0, 1 - \epsilon]$ for some small $\epsilon > 0$, but this did not solve the stability issues either. Ultimately, we abandoned it for this work and left such explorations for future work. It seems reasonable to suppose that schemes which don’t evaluate on the endpoint, *e.g.*, Ralston’s method, Heun’s third-order method, or Ralston’s third-order method may fair better.

I.3 Numerical schemes

We detail the numerical schemes used for posterior guidance beyond Euler.

Midpoint. The midpoint scheme used in both experiments is implemented as

$$\mathbf{x}_1 = \mathbf{x}_t + h\mathbf{u}_\theta\left(t + \frac{h}{2}, \mathbf{x}_t + \frac{h}{2}\mathbf{u}_\theta(t, \mathbf{x}_t)\right) \quad (123)$$

with step size $h = 1 - t$.¹²

2-step Euler. This scheme used in both experiments is implemented as

$$\mathbf{x}_{t+\frac{h}{2}} = \mathbf{x}_t + \frac{h}{2}\mathbf{u}_\theta(t, \mathbf{x}_t), \quad (124)$$

$$\mathbf{x}_1 = \mathbf{x}_{t+\frac{h}{2}} + \frac{h}{2}\mathbf{u}_\theta\left(t + \frac{h}{2}, \mathbf{x}_{t+\frac{h}{2}}\right), \quad (125)$$

with step sizes $h = 1 - t$.

I.4 Hardware and compute cost

Inverse image problems. The inverse image problem experiments were run on a single NVIDIA H100 80GB GPU. It took roughly 4 minutes and 78 GB of VRAM to generate 10 images for each inverse problem. As such each experiment took about an 40–50 minutes. Experiments which used the midpoint method, unsurprisingly ran about 90% slower.

Molecule generation. The molecule generation experiments were run on a single NVIDIA V100 16GB GPU. It took about 3 minutes and 1.5 GB of VRAM to generate 1 molecule leading to the experiments taking on the order of 300 minutes to complete. Experiments which used the midpoint method, unsurprisingly ran about 90% slower.

J Further experimental results

We present additional experimental results that we could not include in the main paper for the sake of space.

¹²This is appropriately adjusted for diffusion models with a terminal time of 0.

J.1 Molecule generation for QM9

In Table 5 we present the *atom stability percentage* (ASP) and *molecule stability percentage* (MSP) per property for each guided generation model. Interestingly, despite their poor quantitative performance in MAE (cf. Table 3) the greedy (midpoint) and (2-step Euler) strategies have slightly better stability than DTO.

Table 5: Stability reported in ASP/MSP per property.

Property	α	$\Delta\varepsilon$	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	μ	C_v
DTO	94.90/65.00	96.20/74.00	95.90/67.00	96.00/65.00	94.60/61.00	95.00/67.00
Greedy (Euler)	94.70/68.80	96.40/76.00	97.40/79.00	98.40/84.80	97.60/84.00	85.55/21.20
Greedy (midpoint)	97.46/80.00	97.51/83.00	97.91/81.00	97.77/83.00	97.70/81.00	97.09/80.00
Greedy (2-step Euler)	97.67/82.00	96.95/74.00	98.18/84.00	96.29/72.00	97.40/93.00	97.75/84.00
EquiFM	98.88/89.00					

J.2 Further results on inverse image problems

To put the results from Section 7.1 into context we present some detailed comparisons to other works from the domain of inverse problems with diffusion models, namely:

1. DAPS (B. Zhang et al. 2025),
2. DPS (Chung, J. Kim, et al. 2023),
3. DDRM (Kawar et al. 2022),
4. DDNM (Y. Wang, Yu, and J. Zhang 2023),
5. DCDP (Li et al. 2024),
6. FPS-SMC (Dou and Y. Song 2024),
7. DiffPIR (Zhu et al. 2023), and
8. RED-diff (Mardani et al. 2024).

We present the full comparison in Table 6.

Table 6: Additional results for inverse image problems on FFHQ 256×256 .

Task	Method	PSNR ($\uparrow$)	SSIM ($\uparrow$)	LPIPS ($\downarrow$)	FID ($\downarrow$)
Super resolution $4\times$	Greedy (Euler)	27.94	0.728	0.217	66.64
	Greedy (midpoint)	27.98	0.727	0.224	70.96
	Greedy (2-step Euler)	27.95	0.728	0.220	68.93
	DAPS	29.07	0.818	0.177	51.44
	DPS	25.86	0.753	0.269	81.07
	DDRM	26.58	0.782	0.282	79.25
	DDNM	28.03	0.795	0.197	64.62
	DCDP	28.66	0.807	0.178	53.81
	FPS-SMC	28.42	0.813	0.204	49.25
	DiffPIR	26.64	-	0.260	65.77
Inpaint (box)	Greedy (Euler)	23.74	0.732	0.187	46.87
	Greedy (midpoint)	24.08	0.724	0.186	44.55
	Greedy (2-step Euler)	23.88	0.720	0.188	44.09
	DAPS	24.07	0.814	0.133	43.10
	DPS	22.51	0.792	0.209	61.27
	DDRM	22.26	0.801	0.207	78.62
	DDNM	24.47	0.837	0.235	46.59
	DCDP	23.89	0.760	0.163	45.23
	FPS-SMC	24.86	0.823	0.146	48.34
Inpaint (random)	Greedy (Euler)	30.87	0.823	0.141	40.73
	Greedy (midpoint)	31.03	0.816	0.139	38.80
	Greedy (2-step Euler)	30.80	0.811	0.144	39.23
	DAPS	31.12	0.844	0.098	32.17
	DPS	25.46	0.823	0.203	69.20
	DDNM	29.91	0.817	0.121	44.37
	DCDP	30.69	0.842	0.142	52.51
	FPS-SMC	28.21	0.823	0.261	61.23
	DiffPIR	27.36	-	0.236	59.65
Gaussian deblurring	Greedy (Euler)	28.01	0.766	0.182	57.04
	Greedy (midpoint)	28.36	0.776	0.185	58.55
	Greedy (2-step Euler)	28.18	0.774	0.181	57.18
	DAPS	29.19	0.817	0.165	53.33
	DPS	25.87	0.764	0.219	79.75
	DDRM	24.93	0.732	0.239	92.43
	DDNM	28.20	0.804	0.216	57.83
	DCDP	27.50	0.699	0.304	86.43
	FPS-SMC	26.54	0.773	0.253	67.45
	DiffPIR	27.36	-	0.236	59.65
Motion deblurring	Greedy (Euler)	29.35	0.748	0.207	63.05
	Greedy (midpoint)	29.73	0.762	0.207	66.21
	Greedy (2-step Euler)	29.64	0.764	0.203	63.99
	DAPS	29.66	0.847	0.157	39.49
	DPS	24.52	0.801	0.246	65.23
	DCDP	25.08	0.512	0.364	125.13
	FPS-SMC	27.39	0.826	0.227	48.32
	DiffPIR	26.57	-	0.255	65.78
	DiffPIR	26.57	-	0.255	65.78
Phase retrieval	Greedy (Euler)	15.10	0.282	0.598	298.06
	Greedy (midpoint)	15.10	0.286	0.595	299.45
	Greedy (2-step Euler)	15.07	0.284	0.598	304.60
	DAPS	30.63 ± 3.13	0.851 ± 0.072	6.139 ± 0.060	42.71
	DPS	17.64 ± 2.97	0.441 ± 0.129	0.410 ± 0.090	104.52
	RED-diff	15.60 ± 4.48	0.398 ± 0.195	0.596 ± 0.092	167.43
	DCDP	28.65 ± 8.09	0.781 ± 0.217	0.203 ± 0.196	68.13
Nonlinear deblur	Greedy (Euler)	24.767	0.551	0.327	79.06
	Greedy (midpoint)	25.09	0.558	0.332	76.73
	Greedy (2-step Euler)	24.81	0.547	0.330	76.26
	DAPS	28.29 ± 1.77	0.783 ± 0.036	0.155 ± 0.032	49.38
	DPS	23.39 ± 2.01	0.623 ± 0.082	0.278 ± 0.060	91.31
	RED-diff	30.86 ± 0.51	0.795 ± 0.028	0.160 ± 0.034	43.84
	DCDP	27.92 ± 2.64	0.779 ± 0.067	0.183 ± 0.051	51.96
High dynamic range	Greedy (Euler)	24.16	0.767	0.181	43.59
	Greedy (midpoint)	26.62	0.809	0.160	37.86
	Greedy (2-step Euler)	25.70	0.797	0.165	37.97
	DAPS	27.12 ± 3.53	0.752 ± 0.041	0.162 ± 0.072	42.97
	DPS	22.73 ± 6.07	0.591 ± 0.141	0.264 ± 0.156	112.82
	RED-diff	22.16 ± 3.41	0.512 ± 0.083	0.258 ± 0.089	108.32

J.3 Sampling trajectories for inverse problems

We present the solution trajectories the different guidance algorithms below for solving the HDR inverse problem. Note that the midpoint and 2-step Euler, unsurprisingly, have better approximations of x_1 .



Figure 6: Sampling trajectory for greedy (Euler) solving the HDR inverse problem. Top row is $x_{1|t}^\theta(x_t)$ and the bottom row is x_t .



Figure 7: Sampling trajectory for greedy (midpoint) solving the HDR inverse problem. Top row is midpoint estimate and the bottom row is x_t .



Figure 8: Sampling trajectory for greedy (2-step Euler) solving the HDR inverse problem. Top row is 2-step Euler estimate and the bottom row is x_t .

J.4 More qualitative samples for inverse problems

We showcase some examples generated by the greedy gradient strategy (Euler) on the different inverse problems.



Figure 9: Qualitative visualization of using greedy guidance to solve the super resolution $4\times$ inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.



Figure 10: Qualitative visualization of using greedy guidance to solve the super resolution $4\times$ inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.



Figure 11: Qualitative visualization of using greedy guidance to solve the Gaussian deblurring inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.



Figure 12: Qualitative visualization of using greedy guidance to solve the motion deblurring inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.



Figure 13: Qualitative visualization of using greedy guidance to solve the Phase retrieval inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.



Figure 14: Qualitative visualization of using greedy guidance to solve the nonlinear deblurring inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.



Figure 15: Qualitative visualization of using greedy guidance to solve the HDR inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.

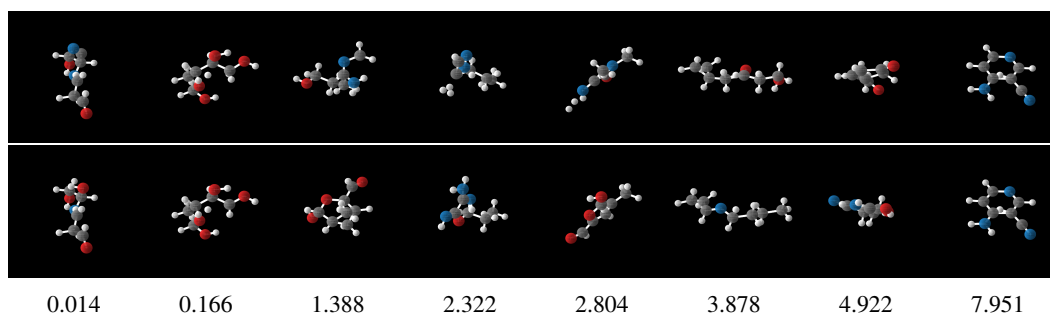


Figure 16: Qualitative visualization of controlled generated molecules for various dipole moments (μ). Top row is generated using a end-to-end guidance with a DTO scheme and the bottom row is generated using posterior guidance.

J.5 More qualitative samples for controlled molecule generation

In Appendix J.4 we present some qualitative results for property-guided molecule generation. In particular, we target different dipole moments.

K Discussions

K.1 Broader Impacts

Controllable generation can be used for many tasks both benign and malicious. The insights from this paper could be used to develop more effective adversarial attacks, generation of harmful content, or other malicious applications.

K.2 Limitations

As this work is mostly theoretical, our experimental illustrations are limited, serving more to illustrate the key concepts rather than advancing the state-of-the-art within the particular problem. We believe that future work can use these insights to make informed design choices when developing solutions to guided generation problems.

In our controllable molecule generation experiments, we take a naïve strategy for annealing the learning rate leaving performance on the table. Moreover, we don’t consider mixed accuracy schemes, *i.e.*, using Euler for certain steps closer to the target and midpoint for steps further away (*cf.* Moufad et al. 2025).