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 [47, 66], images [62, 3], biometrics [6], molecules [31, 2], proteins [82, 68], *&c*.

We focus on training-free guidance methods—in contrast to training-based methods which require training an additional component—due to their flexibility in downstream tasks. 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 [10] and *some* flow models [*cf*. 45, 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. The latter class of techniques, in contrast, performs backpropagation throughout the entire sampling process of the flow/diffusion model [2, 4]. 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.

First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025).

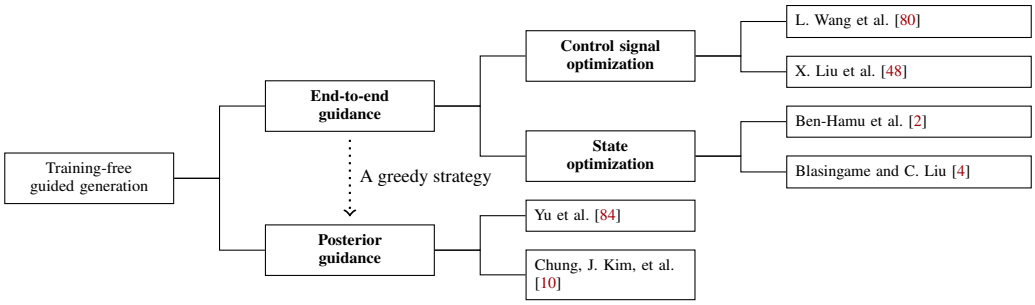

Figure 1: The greedy perspective as a unification of separate families in the taxonomy of training-free guided generation. We provide a more detailed version of this in Figure 5.

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.

Preliminaries. Flow models [44] are a highly popular class of generative models that model the generative process as a neural *ordinary differential equation* (ODE) [9]. Consider two $\mathbb{R}^d$-valued random variables: $\boldsymbol{X}_0 \sim p(\boldsymbol{x})$ and $\boldsymbol{X}_1 \sim q(\boldsymbol{x})$, denoting the *source* (noise) and *target* (data) distributions, respectively. Then consider a time-dependent vector field $\boldsymbol{u} \in \mathcal{C}^{1,r}([0,1] \times \mathbb{R}^d; \mathbb{R}^d)$[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

$$\Phi_0(\boldsymbol{x}) = \boldsymbol{x}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{x}) = \boldsymbol{u}(t, \Phi_t(\boldsymbol{x})). \tag{1}$$

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 $\boldsymbol{u}_t(\boldsymbol{x}) \mapsto \boldsymbol{u}(t, \boldsymbol{x})$. A special case of flow models are known as *affine probability paths* and are defined as $\boldsymbol{X}_t = \alpha_t \boldsymbol{X}_0 + \sigma_t \boldsymbol{X}_1$ with schedule (α_t, σ_t). We provide more details on flow models in Section B.1.[2]

2 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 [13, 72] and classifier-free [28] 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 $\boldsymbol{x}_1$ minimizes $\mathcal{L}$. Suppose we have numerical scheme (Euler, RK4, DPM-Solver, &*c.*) denoted

$$\boldsymbol{\Phi} : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^d \times \mathcal{C}(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d) \to \mathbb{R}^d, \quad \boldsymbol{\Phi}(t_n, t_{n+1}, \boldsymbol{x}_n, \boldsymbol{u}_t^\theta) \mapsto \boldsymbol{x}_{n+1}. \tag{2}$$

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

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

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

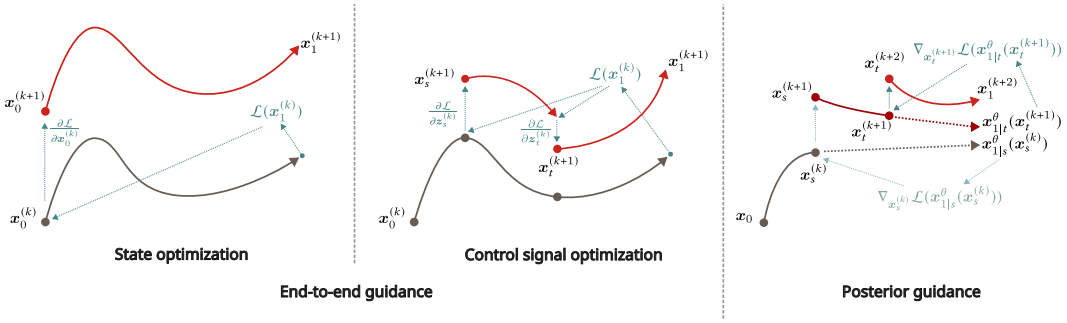

Figure 2: Visual comparison of different training-free guided generation techniques.

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

$$\begin{aligned} &\text{Find a sequence} && \{\boldsymbol{x}_n\}_{n=1}^N && \text{which minimizes } \mathcal{L}(\boldsymbol{x}_N), \\ &\text{subject to} && \boldsymbol{x}_{n+1} = \boldsymbol{\Phi}(t_{n+1}, t_n, \boldsymbol{x}_n). \end{aligned} \tag{3}$$

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.

Posterior guidance. A popular technique for *training-free* guidance is what we will term *posterior guidance* [10, 84]. The key idea behind this strategy is to use the parameterized target prediction model $\boldsymbol{x}_{1|t}^\theta(\boldsymbol{x})$, *i.e.*, the expected value of the posterior distribution given $\boldsymbol{X}_t = \boldsymbol{x}$, to provide a guidance gradient of the form $\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}))$ for some guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$. For literature working with score-based generative models [72], this interpretation arose from the famous Tweedie's formula [74, 17]. Thus, for each $\boldsymbol{x}_n$ in the ODE solve, we add guidance to it in the form of posterior guidance gradient.

End-to-end optimization for guidance. Another popular class of techniques is what we will term *end-to-end guidance* [2, 4], *i.e.*, techniques which perform guidance by optimizing the initial condition $\boldsymbol{x}_0$ w.r.t. the guidance function $\mathcal{L}$; such techniques require performing backpropagation through a neural ODE. 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 [43] 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* [9, *cf*. 37, Section 5.1.2]. 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.

3 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 $\boldsymbol{x}_0$ 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 $\{\boldsymbol{x}_n\}$, can we make use of local information instead? *I.e.*, *rather than solving the full ODE from $\boldsymbol{x}_t$, what if we greedily took a locally optimal step at each $\boldsymbol{x}_t$ instead?*

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

$$\boldsymbol{x}_n^{\mathcal{G}} = \mathcal{G}(t_n, \boldsymbol{x}_n, \boldsymbol{u}_{t_n}^{\theta}), \qquad \boldsymbol{x}_{n+1} = \boldsymbol{\Phi}(t_n, t_{n+1}, \boldsymbol{x}_n^{\mathcal{G}}), \tag{4}$$

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 $\boldsymbol{x}_n^{(0)} = \boldsymbol{x}_n$ which solves

$$\boldsymbol{x}_n^{(k+1)} = \boldsymbol{x}_n^{(k)} - \eta \nabla \mathcal{L}\left(\boldsymbol{x}_{1|t_n}^{\theta}(\boldsymbol{x}_n^{(k)})\right), \tag{5}$$

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 as: *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 (13) over time. Thus, the flow is now expressed as a semi-linear integral equation with linear term $a_t \boldsymbol{x}$ and non-linear term $b_t \boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})$. Due to this semi-linear structure, we apply the same technique of *exponential integrators* [29] that has been successfully used to simplify numerical solvers for diffusion models [49, 86, 21]. *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 (11)) and thus has an inverse t_γ such that $t_\gamma(\gamma(t)) = t$. With abuse of notation, we let $\boldsymbol{x}_\gamma := \boldsymbol{x}_{t_\gamma(\gamma)}$ and $\boldsymbol{x}_{1|\gamma}^{\theta}(\cdot) = \boldsymbol{x}_{1|t_\gamma(\gamma)}^{\theta}(\cdot)$. 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 3.1 with the full proof provided in Section B.3.

Proposition 3.1 (Exact solution of affine probability paths)**.** *Given an initial value of $\boldsymbol{x}_s$ at time $s \in [0, 1]$ the solution $\boldsymbol{x}_t$ at time $t \in [0, 1]$ of an ODE governed by the vector field in Equation (12) is:*

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s} \boldsymbol{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{x}_{1|\gamma}^{\theta}(\boldsymbol{x}_\gamma) \, \mathrm{d}\gamma. \tag{6}$$

Greedy guidance as an Euler scheme. Now equipped with this simplified form, we show in Theorem 3.2 we draw connections between the greedy strategy both DTO and OTD schemes for backpropagation, as an Euler discretization of these schemes with step size $h = \gamma_1 - \gamma_t$.

Theorem 3.2 (Greedy as an Euler scheme)**.** *For some trajectory state $\boldsymbol{x}_t$ at time t, the greedy gradient given by $\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x}))$ is:*

1. *a DTO scheme with an explicit Euler discretization with step size $h = \gamma_1 - \gamma_t$, and*
2. *an OTD scheme with implicit Euler discretization with step size $h = \gamma_1 - \gamma_t$.*

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

$$\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})\right). \tag{7}$$

To do this, we make use of the Gateaux differential [20] which allows us to define the differential that describes how the output of the flow model $\boldsymbol{x}_1$ evolves with changes to $\boldsymbol{x}$ at time t. We present the result to this question in Proposition 3.3 below.

Proposition 3.3 (Dynamics of greedy gradient guidance)**.** *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\boldsymbol{x}$ at some time $t \in [0,1]$ in the direction of the gradient $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})\right)$ is given by*

$$\delta_{\boldsymbol{x}}^{\mathcal{G}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})^{\top} \nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1). \tag{8}$$

Remark 3.1. From [2, Proposition 4.1] we know that both $\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x})$ and $\nabla_{\boldsymbol{x}} \boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})$ consist of covariance matrices, thus the dynamics of greedy gradient guidance are governed by this covariance projection of the loss.

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 $\Phi_{t,1}^{\theta}(\boldsymbol{x}_t)$. 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 step size, convergence in the greedy solution implies convergence in the flow, which we state more formally in Theorem 3.4.

Theorem 3.4 (Greedy convergence)**.** *For affine probability paths, if there exists a sequence of states $\boldsymbol{x}_t^{(n)}$ at time t such that it converges to the locally optimal solution $\boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x}_t^{(n)}) \to \boldsymbol{x}_1^*$. Then the solution, $\Phi_{t,1}^{\theta}(\boldsymbol{x}_t^{(n)})$, converges to a neighborhood of size $\mathcal{O}(h^2)$ centered at $\boldsymbol{x}_1^*$.*

3.2 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 3.5, 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 3.5 (Truncation error of single-step gradients)**.** *Let $\boldsymbol{\Phi}$ be an explict Runge-Kutta solver of order $\alpha > 0$ of a flow model with flow $\Phi_{s,t}^{\theta}(\boldsymbol{x})$. Then for any $t \in [0,1]$,*

$$\left\| \nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \boldsymbol{\Phi}_{t,1}(\boldsymbol{x}) \right\| = \mathcal{O}(h^{\alpha+1}), \tag{9}$$

where $h = 1 - t$.

Insight. *We can use a higher-order solver to move between posterior and end-to-end guidance exchanging compute for 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.

4 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.

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. All tasks are using a noisy measurement with noise level $\beta_{\boldsymbol{y}} = 0.05$. The full results are found in Table 5.

Task	**Method**	**PSNR** (↑)	**SSIM** (↑)	**LPIPS** (↓)	**FID** (↓)
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

4.1 Inverse problems for images

A common application of posterior guidance has been in solving inverse problems [72, 11] (*cf*. Appendix G). 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 [33] 256×256 dataset. We make use of the pre-trained diffusion model from Chung, J. Kim, et al. [10] trained on the FFHQ dataset. Following [85] we conduct experiments on the following linear tasks: super resolution, Gaussian deblurring, motion deblurring, inpaintining (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) [87], and *Fréchet Inception Distance* (FID) [26]. Further configuration details are reported in Section H.1.

Results. We present some qualitative results on reconstructing images from a random mask in Section 4. Quantitatively we present a snapshot of our full results (*cf*. Table 5) on the inpainting with random mask and Gaussian deblurring tasks. For reference we include the standard DPS [10] and the recent state-of-the-art DAPS [85]. 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 Section I.2 along with additional analysis and discussion.

4.2 Molecule generation for QM9

We also illustrate the core ideas with some experiments in controllable molecule generation on the QM9 dataset [63], a popular molecular dataset containing small molecules with up to 29 atoms. Following Hoogeboom et al. [31] and Ben-Hamu et al. [2], we perform the conditional generation of molecules with specified quantum chemical property values. In particular, we target the following properties: polarizability α, orbital energies $\varepsilon_{\text{HOMO}}, \varepsilon_{\text{LUMO}}$ and their gap $\Delta\varepsilon$, dipole moment μ, and heat capacity C_v. The property classifiers were trained following the methodology outlined in Hoogeboom et al. [31]. The underlying flow model is an unconditional equivariant flow matching model with *conditional optimal transport* path [45, Section 4.7], *i.e*., the EquiFM [73] model. Further details are provided in Section H.2. To evaluate the guided generation we calculate the *mean*

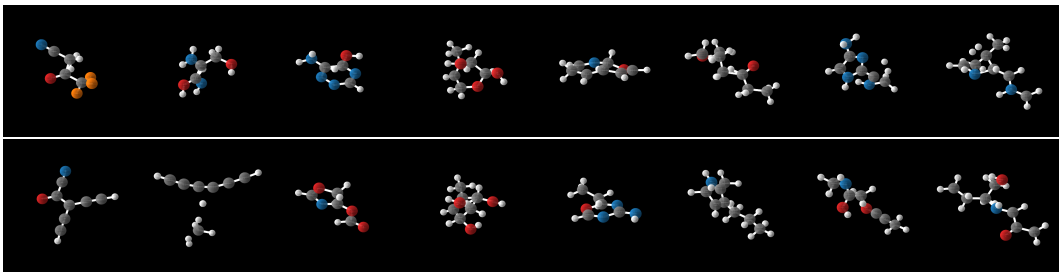

Figure 4: Qualitative visualization of controlled generated molecules for various polarizability (α) levels. Top row is generated using a end-to-end guidance with a DTO scheme and the bottom row is generated using posterior guidance.

absolute error (MAE) between the predicted property value of the generated molecule by the property classifier and the target property value [65]. Additionally in Section I.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).

Table 2: 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)	5.377	1275	560	1204	1.563	2.975
DTO	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

Results. In Section 4.1 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 generated by a DTO scheme decreases. This trend is less noticeable for the posterior guided samples. We report quantitative results in Table 2. We report the unguided EquiFM generated molecules as an upper bound and include the theoretical lower bounds from Ben-Hamu et al. [2]. It is here that we notice a sharp decrease in performance from using posterior guidance. In particular the greedy (Euler) strategy is 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.

5 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

[1] A. Bansal, H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein. "Universal guidance for diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 843–852 (cit. on p. 16).

[2] H. Ben-Hamu, O. Puny, I. Gat, B. Karrer, U. Singer, and Y. Lipman. "D-Flow: Differentiating through Flows for Controlled Generation". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=SE20BFqj6J (cit. on pp. 1–3, 5–7, 16–18, 20–23, 33).

[3] Black Forest Labs. *FLUX*. https://github.com/black-forest-labs/flux. 2024 (cit. on p. 1).

[4] Blasingame and C. Liu. "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., 2024, pp. 2449–2483. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf (cit. on pp. 1–3, 16, 17, 22, 30, 31).

[5] Blasingame and C. Liu. "Greedy-DiM: Greedy Algorithms for Unreasonably Effective Face Morphs". In: *2024 IEEE International Joint Conference on Biometrics (IJCB)*. 2024, pp. 1–11. DOI: 10.1109/IJCB62174.2024.10744517 (cit. on p. 31).

[6] Blasingame and C. Liu. "Leveraging diffusion for strong and high quality face morphing attacks". In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 6.1 (2024), pp. 118–131 (cit. on p. 1).

[7] Blasingame and C. Liu. "A Reversible Solver for Diffusion SDEs". In: *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. 2025. URL: https://openreview.net/forum?id=0gEFLVUL6n (cit. on p. 30).

[8] J. C. Butcher. *Numerical methods for ordinary differential equations*. Third Edition. John Wiley & Sons, 2016 (cit. on p. 30).

[9] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 2, 3, 29, 31).

[10] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. "Diffusion Posterior Sampling for General Noisy Inverse Problems". In: *The Eleventh International Conference on Learning Representations, ICLR 2023*. The International Conference on Learning Representations. 2023 (cit. on pp. 1–3, 6, 16, 32, 33, 35).

[11] H. Chung, B. Sim, and J. C. Ye. "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*. 2022, pp. 12413–12422 (cit. on p. 6).

[12] K. Clark, P. Vicol, K. Swersky, and D. J. Fleet. "Directly Fine-Tuning Diffusion Models on Differentiable Rewards". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=1vmSEVL19f (cit. on pp. 16, 17).

[13] P. Dhariwal and A. Nichol. "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., 2021, pp. 8780–8794. URL: https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf (cit. on p. 2).

[14] C. Domingo-Enrich, M. Drozdzal, B. Karrer, and R. T. Q. Chen. "Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=xQBRrtQM8u (cit. on p. 30).

[15] Z. Dou and Y. Song. "Diffusion Posterior Sampling for Linear Inverse Problem Solving: A Filtering Perspective". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=tplXNcHZs1 (cit. on p. 35).

[16] F. J. Dyson. "The radiation theories of Tomonaga, Schwinger, and Feynman". In: *Physical Review* 75.3 (1949), p. 486 (cit. on p. 23).

[17] B. Efron and N. R. Zhang. "False discovery rates and copy number variation". In: *Biometrika* 98.2 (2011), pp. 251–271 (cit. on p. 3).

[18] R. A. Frazer, W. J. Duncan, and A. R. Collar. *Elementary matrices and some applications to dynamics and differential equations*. Cambridge University Press, 1938 (cit. on p. 23).

[19] P. K. Friz and N. B. Victoir. *Multidimensional stochastic processes as rough paths: theory and applications*. Vol. 120. Cambridge University Press, 2010 (cit. on p. 22).

[20] R. Gâteaux. "Sur les fonctionnelles continues et les fonctionnelles analytiques". In: *CR Acad. Sci. Paris* 157.325-327 (1913), p. 65 (cit. on p. 5).

[21] M. Gonzalez, N. Fernandez Pinto, T. Tran, H. Hajri, N. Masmoudi, et al. "Seeds: Exponential sde solvers for fast high-quality sampling from diffusion models". In: *Advances in Neural Information Processing Systems* 36 (2024) (cit. on pp. 4, 19).

[22] A. Griewank. "Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation". In: *Optimization Methods and software* 1.1 (1992), pp. 35–54 (cit. on p. 30).

[23] A. Griewank and A. Walther. "Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation". In: *ACM Trans. Math. Softw.* 26.1 (2000), pp. 19–45. DOI: 10.1145/347837.347846 (cit. on p. 30).

[24] M. Grossman and R. Katz. *Non-Newtonian Calculus: A Self-contained, Elementary Exposition of the Authors' Investigations...* Non-Newtonian Calculus, 1972 (cit. on p. 23).

[25] E. Harier and G. Wanner. *Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems.* 2nd ed. Springer Series in Computational Mathematics. Berlin, Germany: Springer, Feb. 2002 (cit. on p. 30).

[26] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "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., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf (cit. on p. 6).

[27] J. Ho, A. Jain, and P. Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851 (cit. on p. 18).

[28] J. Ho and T. Salimans. "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021 (cit. on p. 2).

[29] M. Hochbruck and A. Ostermann. "Exponential integrators". In: *Acta Numerica* 19 (2010), pp. 209–286 (cit. on p. 4).

[30] P. Holderrieth, M. Havasi, J. Yim, N. Shaul, I. Gat, T. Jaakkola, B. Karrer, R. T. Q. Chen, and Y. Lipman. "Generator Matching: Generative modeling with arbitrary Markov processes". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=RuP17cJtZo (cit. on p. 17).

[31] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling. "Equivariant diffusion for molecule generation in 3d". In: *International conference on machine learning*. PMLR. 2022, pp. 8867–8887 (cit. on pp. 1, 6, 33).

[32] Z. Kadkhodaie and E. Simoncelli. "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., 2021, pp. 13242–13254. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/6e28943943dbed3c7f82fc05f269947a-Paper.pdf (cit. on p. 16).

[33] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4396–4405. DOI: 10.1109/CVPR.2019.00453 (cit. on p. 6).

[34] T. Karras, M. Aittala, T. Aila, and S. Laine. "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. 2022. URL: https://openreview.net/forum?id=k7FuTOWMOc7 (cit. on pp. 32, 33).

[35] K. Karunratanakul, K. Preechakul, E. Aksan, T. Beeler, S. Suwajanakorn, and S. Tang. "Optimizing diffusion noise can serve as universal motion priors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 1334–1345 (cit. on pp. 16, 17).

[36] B. Kawar, M. Elad, S. Ermon, and J. Song. "Denoising Diffusion Restoration Models". In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=kxXvopt9pWK (cit. on p. 35).

[37] P. Kidger. "On Neural Differential Equations". Available at https://arxiv.org/abs/2202.02435. Ph.D. thesis. Oxford University, 2022 (cit. on pp. 3, 22, 29, 30).

[38] P. Kidger, J. Foster, X. C. Li, and T. Lyons. "Efficient and accurate gradients for neural sdes". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18747–18761 (cit. on p. 30).

[39] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas. "Stiff neural ordinary differential equations". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.9 (2021) (cit. on p. 30).

[40] D. Kingma, T. Salimans, B. Poole, and J. Ho. "Variational diffusion models". In: *Advances in neural information processing systems* 34 (2021), pp. 21696–21707 (cit. on p. 18).

[41] D. E. Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004 (cit. on p. 31).

[42] X. Li, S. M. Kwon, I. R. Alkhouri, S. Ravishankar, and Q. Qu. "Decoupled data consistency with diffusion purification for image restoration". In: *arXiv preprint arXiv:2403.06054* (2024) (cit. on p. 35).

[43] S. Linnainmaa. "Taylor expansion of the accumulated rounding error". In: *BIT* 16.2 (June 1976), pp. 146–160. ISSN: 0006-3835. DOI: 10.1007/BF01931367. URL: https://doi.org/10.1007/BF01931367 (cit. on p. 3).

[44] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. "Flow Matching for Generative Modeling". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=PqvMRDCJT9t (cit. on pp. 2, 17).

[45] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat. "Flow Matching Guide and Code". In: *arXiv preprint arXiv:2412.06264* (2024) (cit. on pp. 1, 6, 18, 32).

[46] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528 (cit. on p. 33).

[47] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley. "AudioLDM: Text-to-Audio Generation with Latent Diffusion Models". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 21450–21474 (cit. on p. 1).

[48] X. Liu, L. Wu, S. Zhang, C. Gong, W. Ping, and Q. Liu. "Flowgrad: Controlling the output of generative odes with gradients". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 24335–24344 (cit. on pp. 2, 16, 17, 31).

[49] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. "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. 2022. URL: https://openreview.net/forum?id=2uAaGwlP_V (cit. on pp. 4, 19).

[50] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. "Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models". In: *arXiv preprint arXiv:2211.01095* (2022) (cit. on p. 19).

[51] W. Magnus. "On the exponential solution of differential equations for a linear operator". In: *Communications on pure and applied mathematics* 7.4 (1954), pp. 649–673 (cit. on p. 23).

[52] M. Mardani, J. Song, J. Kautz, and A. Vahdat. "A Variational Perspective on Solving Inverse Problems with Diffusion Models". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=1YO4EE3SPB (cit. on p. 35).

[53] P. Marion, A. Korba, P. Bartlett, M. Blondel, V. D. Bortoli, A. Doucet, F. Llinares-López, C. Paquette, and Q. Berthet. "Implicit Diffusion: Efficient optimization through stochastic sampling". In: *The 28th International Conference on Artificial Intelligence and Statistics*. 2025. URL: https://openreview.net/forum?id=r5F7Z8s0Qk (cit. on pp. 16, 17).

[54] S. McCallum and J. Foster. "Efficient, Accurate and Stable Gradients for Neural ODEs". In: *arXiv preprint arXiv:2410.11648* (2024) (cit. on p. 30).

[55] B. Moufad, Y. Janati, L. Bedin, A. O. Durmus, R. Douc, E. Moulines, and J. Olsson. "Variational Diffusion Posterior Sampling with Midpoint Guidance". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=6EUtjXAvmj (cit. on pp. 16, 32, 33, 41).

[56] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar. "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, 17–23 Jul 2022, pp. 16805–16827. URL: https://proceedings.mlr.press/v162/nie22a.html (cit. on p. 17).

[57] Z. Novack, J. McAuley, T. Berg-Kirkpatrick, and N. J. Bryan. "DITTO: Diffusion Inference-Time T-Optimization for Music Generation". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=z5Ux2u6t7U (cit. on pp. 16, 17).

[58] D. Onken and L. Ruthotto. "Discretize-optimize vs. optimize-discretize for time-series regression and continuous normalizing flows". In: *arXiv preprint arXiv:2005.13420* (2020) (cit. on p. 28).

[59] J. Pan, J. H. Liew, V. Tan, J. Feng, and H. Yan. "AdjointDPM: Adjoint Sensitivity Method for Gradient Backpropagation of Diffusion Probabilistic Models". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=y33lDRBgWI (cit. on pp. 16, 17, 29).

[60] J. Pan, H. Yan, J. H. Liew, J. Feng, and V. Y. Tan. "Towards accurate guided diffusion sampling through symplectic adjoint method". In: *arXiv preprint arXiv:2312.12030* (2023) (cit. on p. 17).

[61] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishechenko. "The Mathematical Theory of Optimal Processes." In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11 (1963), 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 p. 31).

[62] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695 (cit. on p. 1).

[63] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond. "Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17". In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2864–2875 (cit. on p. 6).

[64] J. J. Sakurai and J. Napolitano. *Modern quantum mechanics*. Cambridge University Press, 2020 (cit. on p. 23).

[65] V. G. Satorras, E. Hoogeboom, F. B. Fuchs, I. Posner, and M. Welling. "E(n) Equivariant Normalizing Flows". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021. URL: https://openreview.net/forum?id=N5hQI_RowVA (cit. on p. 7).

[66] F. Schneider, O. Kamal, Z. Jin, and B. Schölkopf. "Moûsai: Efficient text-to-music diffusion models". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2024, pp. 8050–8068 (cit. on p. 1).

[67] B. Shi, S. S. Du, M. I. Jordan, and W. J. Su. "Understanding the acceleration phenomenon via high-resolution differential equations". en. In: *Math. Program.* 195.1-2 (Sept. 2022), pp. 79–148 (cit. on p. 28).

[68] M. Skreta, L. Atanackovic, J. Bose, A. Tong, and K. Neklyudov. "The Superposition of Diffusion Models Using the Itô Density Estimator". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=2o58Mbqkd2 (cit. on p. 1).

[69] B. Song, S. M. Kwon, Z. Zhang, X. Hu, Q. Qu, and L. Shen. "Solving Inverse Problems with Latent Diffusion Models via Hard Data Consistency". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=j8hdRqOUhN (cit. on p. 33).

[70] J. Song, A. Vahdat, M. Mardani, and J. Kautz. "Pseudoinverse-Guided Diffusion Models for Inverse Problems". In: *International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=9_gsMA8MRKQ (cit. on p. 16).

[71] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. "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, 23–29 Jul 2023, pp. 32211–32252. URL: https://proceedings.mlr.press/v202/song23a.html (cit. on p. 33).

[72] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=PxTIG12RRHS (cit. on pp. 2, 3, 6, 18, 32).

[73] Y. Song, J. Gong, M. Xu, Z. Cao, Y. Lan, S. Ermon, H. Zhou, and W.-Y. Ma. "Equivariant Flow Matching with Hybrid Probability Transport for 3D Molecule Generation". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=hHUZ5V9XFu (cit. on pp. 6, 33).

[74] C. M. Stein. "Estimation of the mean of a multivariate normal distribution". In: *The annals of Statistics* (1981), pp. 1135–1151 (cit. on pp. 3, 16, 32).

[75] D. E. Stewart. *Numerical analysis: A graduate course*. Vol. 258. Springer, 2022 (cit. on p. 27).

[76] P. Stumm and A. Walther. "New Algorithms for Optimal Online Checkpointing". In: *SIAM Journal on Scientific Computing* 32.2 (2010), pp. 836–854. DOI: 10.1137/080742439 (cit. on p. 30).

[77] B. Wallace, A. Gokul, S. Ermon, and N. Naik. "End-to-end diffusion latent optimization improves classifier guidance". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 7280–7290 (cit. on p. 17).

[78] B. Wallace, A. Gokul, and N. Naik. "Edict: Exact diffusion inversion via coupled transformations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22532–22541 (cit. on p. 17).

[79] F. Wang, H. Yin, Y.-J. Dong, H. Zhu, C. Zhang, H. Zhao, H. Qian, and C. Li. "BELM: Bidirectional Explicit Linear Multi-step Sampler for Exact Inversion in Diffusion Models". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=ccQ4fmwLDb (cit. on p. 17).

[80] L. Wang, C. Cheng, Y. Liao, Y. Qu, and G. Liu. "Training Free Guided Flow-Matching with Optimal Control". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=61ss5RA1MM (cit. on pp. 2, 16, 17, 31, 33).

[81] Y. Wang, J. Yu, and J. Zhang. "Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=mRieQgMtNTQ (cit. on pp. 16, 35).

[82] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, et al. "De novo design of protein structure and function with RFdiffusion". In: *Nature* 620.7976 (2023), pp. 1089–1100 (cit. on p. 1).

[83] S. Weinberg. *The quantum theory of fields*. Vol. 2. Cambridge university press, 1995 (cit. on p. 23).

[84] J. Yu, Y. Wang, C. Zhao, B. Ghanem, and J. Zhang. "Freedom: Training-free energy-guided conditional diffusion model". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 23174–23184 (cit. on pp. 2, 3, 16, 33).

[85] B. Zhang, W. Chu, J. Berner, C. Meng, A. Anandkumar, and Y. Song. *Improving Diffusion Inverse Problem Solving with Decoupled Noise Annealing*. 2024. arXiv: 2407.01521 [cs.LG]. URL: https://arxiv.org/abs/2407.01521 (cit. on pp. 6, 16, 32, 33, 35).

[86] Q. Zhang and Y. Chen. "Fast Sampling of Diffusion Models with Exponential Integrator". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=Loek7hfb46P (cit. on pp. 4, 19).

[87] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595 (cit. on p. 6).

[88] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool. “Denoising Diffusion Models for Plug-and-Play Image Restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 1219–1229 (cit. on p. 35).

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 3 in the main paper. Likewise, Appendices C and D is devoted to proofs and derivations from Sections 3.1 and 3.2 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 G is devoted to providing a brief background on inverse problems. Likewise, Appendix H is devoted to discussing the implementation details of the numerical experiments in Section 4 and providing a background for the experiments. In Appendix I we include additional results that we could not fit into the main paper. Lastly, in Appendix J we discuss the limitations and broader impacts of this research.

Appendices

Overview of theoretical results

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

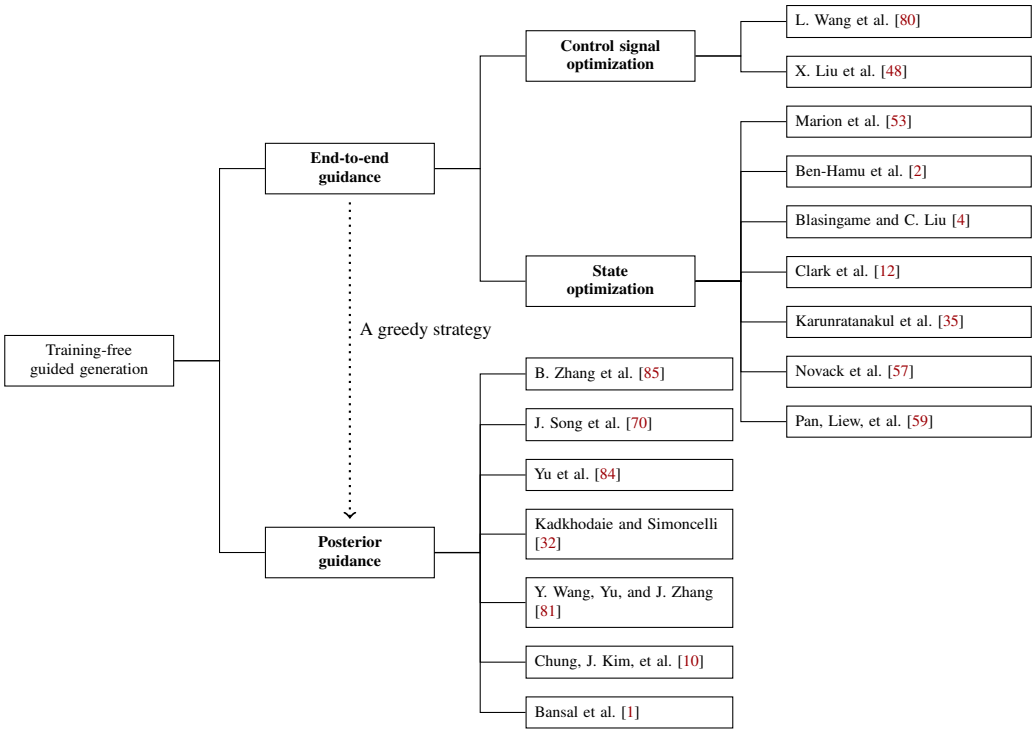

Figure 5: A more detailed taxonomy of *training-free guided generation* methods from Figure 1 from the main paper.

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) [10] is a guidance method that uses Tweedie's formula [74] to estimate the gradient of some guidance function defined in the output state w.r.t. the noisy state, *i.e.*, $\mathbb{E}[\boldsymbol{X}_1|\boldsymbol{X}_t = \boldsymbol{x}]$. Likewise, the work of Bansal et al. [1], Y. Wang, Yu, and J. Zhang [81], and Yu et al. [84] 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 [84] 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. [85] explored modeling Langevin dynamics on top of a diffusion ODE to correct measurement errors in inverse problems.

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 [55, 84].

Beyond Euler. Recent work by Moufad et al. [55] explores an extension to [10] 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. [48] 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. [2] makes use of the continuous adjoint equations, in practice they use the *generally* preferred approach of DTO with gradient checkpointing.[3] Likewise, Clark et al. [12], Karunratanakul et al. [35], and Novack et al. [57] 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. [56] which used OTD with SDEs for the adversarial purification task. More general work came later by Ben-Hamu et al. [2], Blasingame and C. Liu [4], and Pan, Liew, et al. [59]. More specifically, Pan, Liew, et al. [59] and Pan, Yan, et al. [60] explore bespoke solvers for the continuous adjoint equations of diffusion ODEs. Blasingame and C. Liu [4] 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. [53] 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. [2] 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. [80] explores an extension of Ben-Hamu et al. [2] 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. [77] who uses EDICT [78], 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 [4] 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*. 79].

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 $\boldsymbol{z}(t)$ rather than the solution trajectory $\boldsymbol{x}(t)$; namely, X. Liu et al. [48] and L. Wang et al. [80].

B A greedy perspective

We present the proofs and derivations associated with Section 3.

B.1 Additional details on flow models

Applying this flow to the random variable $\boldsymbol{X}_0$ we define a *continuous-time Markov process* $\{\boldsymbol{X}_t\}_{t\in[0,1]}$ with mapping $\boldsymbol{X}_t = \Phi_t(\boldsymbol{X}_0)$. The *goal*, then, is to learn a flow Φ_t such that $\boldsymbol{X}_1 = \Phi_1(\boldsymbol{X}_0) \sim q(\boldsymbol{x})$. This procedure amounts to learning a neural network parameterized vector field $\boldsymbol{u}^\theta \in \mathcal{C}^{1,r}([0,1]\times\mathbb{R}^d;\mathbb{R}^d)$; this learning procedure can be performed efficiently through a *simulation-free* training process known as *flow matching* [44] or more generally *generator matching* [30].

[3] See https://docs.kidger.site/diffrax/api/adjoints/ for an excellent summary of such design considerations and why DTO is generally preferable over OTD.

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 $\Phi_t^\theta(\boldsymbol{x})$. We let $\Phi_{s,t}(\boldsymbol{x}) = (\Phi_t \circ \Phi_s^{-1})(\boldsymbol{x})$ 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 $\{\boldsymbol{X}_t\}$ is described via the affine equation

$$\boldsymbol{X}_t = \alpha_t \boldsymbol{X}_1 + \sigma_t \boldsymbol{X}_0, \tag{10}$$

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

$$\alpha_0 = \sigma_1 = 0, \quad \alpha_1 = \sigma_0 = 1, \quad \forall t \in (0,1)\ [\dot{\alpha}_t > 0,\ \dot{\sigma}_t < 0]. \tag{11}$$

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

$$\boldsymbol{u}_t(\boldsymbol{x}) = \mathbb{E}[\dot{\alpha}_t \boldsymbol{X}_1 + \dot{\sigma}_t \boldsymbol{X}_0 | \boldsymbol{X}_t = \boldsymbol{x}]. \tag{12}$$

This *nice* form of the marginal vector field enables use to rewrite the vector field in the forms of either source [27] or target [40] prediction as

$$\boldsymbol{u}_t(\boldsymbol{x}) = \underbrace{\frac{\dot{\beta}_t}{\beta_t}}_{=a_t} \boldsymbol{x} + \underbrace{\frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t}}_{=b_t} \boldsymbol{f}_t(\boldsymbol{x}), \tag{13}$$

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

Remark B.1. The probability flow ODE formulation of *diffusion models* [72] is subsumed by flow models, and represents a model with an affine Gaussian probability paths (AGGP), *i.e.*, $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim \pi_{0,1}(\boldsymbol{x}_0, \boldsymbol{x}_1) = p(\boldsymbol{x}_0)q(\boldsymbol{x}_1)$ with $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \sigma^2 \mathbf{I})$ [45]. Thus without loss of generality we consider flow models of affine probability paths.[4]

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* $a_t := \frac{\dot{\sigma}_t}{\sigma_t}$ *is integrable in* $[0,1]$.

Assumption B.2. *The total derivatives* $\frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}_{1|\gamma}^\theta(\boldsymbol{x})\right]$ *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. [2] 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 $\boldsymbol{x}_{1|\gamma}^\theta(\boldsymbol{x})$.

B.3 Proof of Proposition 3.1

We restate Proposition 3.1 below.

Proposition 3.1 (Exact solution of affine probability paths)**.** *Given an initial value of* $\boldsymbol{x}_s$ *at time* $s \in [0,1]$ *the solution* $\boldsymbol{x}_t$ *at time* $t \in [0,1]$ *of an ODE governed by the vector field in Equation* (12) *is:*

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s} \boldsymbol{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{x}_{1|\gamma}^\theta(\boldsymbol{x}_\gamma)\, \mathrm{d}\gamma. \tag{6}$$

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

$$\boldsymbol{u}_t^\theta(\boldsymbol{x}) = \mathbb{E}[\dot{\alpha}_t \boldsymbol{X}_1 + \dot{\sigma}_t \boldsymbol{X}_0 | \boldsymbol{X}_t = \boldsymbol{x}]. \tag{14}$$

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

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

$$\begin{aligned} \boldsymbol{u}_t^\theta(\boldsymbol{x}) &= a_t \boldsymbol{x} + b_t \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}), \\ a_t = \frac{\dot{\sigma}_t}{\sigma_t} \qquad b_t &= \dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}. \end{aligned} \tag{15}$$

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

$$\boldsymbol{x}_t = \boldsymbol{x}_s + \int_s^t \boldsymbol{u}_\tau^\theta(\boldsymbol{x}_\tau) \, \mathrm{d}\tau, \tag{16}$$

$$\boldsymbol{x}_t = \boldsymbol{x}_s + \int_s^t a_\tau \boldsymbol{x}_\tau + b_\tau \boldsymbol{x}_{1|\tau}^\theta(\boldsymbol{x}_\tau) \, \mathrm{d}\tau. \tag{17}$$

Note the semi-linear form of the integral equation. We can exploit this structure using the technique of *exponential integrators*, [see 21, 49, 86], to simplify Equation (17), under Assumption B.1, to

$$\boldsymbol{x}_t = e^{\int_s^t a_u \, \mathrm{d}u} \boldsymbol{x}_s + \int_s^t e^{\int_\tau^t a_u \, \mathrm{d}u} b_\tau \boldsymbol{x}_{1|\tau}^\theta(\boldsymbol{x}_\tau) \, \mathrm{d}\tau. \tag{18}$$

Now, the integrating factor simplifies quite nicely to

$$e^{\int_s^t a_u \, \mathrm{d}u} = e^{\int_s^t \frac{\dot{\sigma}_u}{\sigma_u} \, \mathrm{d}u} = e^{\int_{\sigma_s}^{\sigma_t} \frac{1}{\sigma} \, \mathrm{d}\sigma} = \frac{\sigma_t}{\sigma_s}, \tag{19}$$

such that Equation (18) becomes

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s} \boldsymbol{x}_s + \sigma_t \int_s^t \frac{b_\tau}{\sigma_\tau} \boldsymbol{x}_{1|\tau}^\theta(\boldsymbol{x}_\tau) \, \mathrm{d}\tau. \tag{20}$$

We can simplify b_t/σ_t to find:

$$\frac{b_t}{\sigma_t} = \frac{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}{\sigma_t^2} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\alpha_t}{\sigma_t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\gamma_t, \tag{21}$$

where $\gamma_t \coloneqq \alpha_t/\sigma_t$, *i.e.*, the signal-to-noise ratio. As such, we can rewrite Equation (20) with a change of variables $\boldsymbol{x}_\gamma = \boldsymbol{x}_{\gamma_t^{-1}(\gamma)} = \boldsymbol{x}_t$,

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s} \boldsymbol{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{x}_{1|\gamma}^\theta(\boldsymbol{x}_\gamma) \, \mathrm{d}\gamma, \tag{22}$$

concluding the proof. □

Remark B.2. This result bears some similarity to Lu et al. [50, Propostion 5.1]; however, they integrate w.r.t. the log-SNR; their result can be recovered, *mutatis mutandis*, with the identity $\lambda_t = \log \gamma_t$.

B.4 Proof of Theorem 3.2

We restate Theorem 3.2 below.

Theorem 3.2 (Greedy as an Euler scheme)**.** *For some trajectory state $\boldsymbol{x}_t$ at time t, the greedy gradient given by $\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}))$ is:*

1. *a DTO scheme with an explicit Euler discretization with step size $h = \gamma_1 - \gamma_t$, and*
2. *an OTD scheme with implicit Euler discretization with step size $h = \gamma_1 - \gamma_t$.*

Proof. We prove both statements invidually as seperate propositions in Sections B.4.1 and B.4.2. □

B.4.1 Proof of Proposition B.1

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

Proof. From Proposition 3.1 we see that using the target prediction model to estimate $\boldsymbol{x}_1$ 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 (6) with:

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t \sum_{n=0}^{k-1} \frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}^\theta_{1|\gamma}(\boldsymbol{x}_\gamma)\right]_{\gamma=\gamma_s} \int_{\gamma_s}^{\gamma_t} \frac{(\gamma-\gamma_s)^n}{n!}\,\mathrm{d}\gamma + \mathcal{O}(h^{k+1}), \tag{23}$$

$$= \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t \sum_{n=0}^{k-1} \frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}^\theta_{1|\gamma}(\boldsymbol{x}_\gamma)\right]_{\gamma=\gamma_s} \frac{h^{n+1}}{(n+1)!} + \mathcal{O}(h^{k+1}), \tag{24}$$

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

$$\boldsymbol{x}_t \approx \tilde{\boldsymbol{x}}_t = \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + (\alpha_t + \frac{\sigma_t\alpha_s}{\sigma_s})\boldsymbol{x}^\theta_{1|s}(\boldsymbol{x}_s). \tag{25}$$

In the limit as $t \to 1$ we have $\tilde{\boldsymbol{x}}_t = \boldsymbol{x}^\theta_{1|s}(\boldsymbol{x}_s)$.[5] Thus, the greedy gradient is a DTO scheme with an explicit Euler discretization with step size $h = \gamma_1 - \gamma_t$. □

B.4.2 Proof of Proposition B.2

We restate Proposition B.2 below.

Proposition B.2 (Greedy as an implicit Euler scheme within OTD)**.** *For some trajectory state $\boldsymbol{x}_t$ at time t, the greedy gradient given by $\nabla_{\boldsymbol{x}_t}\mathcal{L}(\boldsymbol{x}^\theta_{1|t}(\boldsymbol{x}_t))$ 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 $\boldsymbol{u}_\theta \in \mathcal{C}^{1,1}([0,1]\times\mathbb{R}^d;\mathbb{R}^d)$ be a model that models the vector field of some ODE and be Lipschitz continuous in its second argument. Let $\boldsymbol{x} : [0,1] \to \mathbb{R}^d$ be the solution to the ODE with the initial condition $\boldsymbol{x}_0 \in \mathbb{R}^d$, $\dot{\boldsymbol{x}}_t = \boldsymbol{u}_\theta(t, \boldsymbol{x}_t)$. For some scalar-valued loss function $\mathcal{L} \in \mathcal{C}^2(\mathbb{R}^d)$ in $\boldsymbol{x}_1$, let $\boldsymbol{a}_{\boldsymbol{x}} \coloneqq \partial\mathcal{L}/\partial\boldsymbol{x}_t$ denote the gradient. Then $\boldsymbol{a}_{\boldsymbol{x}}$ and related quantity $\boldsymbol{a}_\theta \coloneqq \partial\mathcal{L}/\partial\theta$ can be found by solving an augmented ODE of the form,

$$\begin{aligned} \boldsymbol{a}_{\boldsymbol{x}}(1) &= \frac{\partial\mathcal{L}}{\partial\boldsymbol{x}_1}, & \frac{\mathrm{d}\boldsymbol{a}_{\boldsymbol{x}}}{\mathrm{d}t}(t) &= -\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial\boldsymbol{u}_\theta}{\partial\boldsymbol{x}}(t, \boldsymbol{x}_t), \\ \boldsymbol{a}_\theta(1) &= \mathbf{0}, & \frac{\mathrm{d}\boldsymbol{a}_\theta}{\mathrm{d}t}(t) &= -\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial\boldsymbol{u}_\theta}{\partial\theta}(t, \boldsymbol{x}_t). \end{aligned} \tag{26}$$

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

$$\frac{\mathrm{d}\boldsymbol{a}_{\boldsymbol{x}}}{\mathrm{d}t}(t) = -a_t\boldsymbol{a}_{\boldsymbol{x}}(t) - b_t\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial\boldsymbol{x}^\theta_{1|t}(\boldsymbol{x}_t)}{\partial\boldsymbol{x}_t}. \tag{27}$$

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

$$\begin{aligned} \boldsymbol{a}_{\boldsymbol{x}}(s) &= \boldsymbol{a}_{\boldsymbol{x}}(t) - \int_t^s a_\tau\boldsymbol{a}_{\boldsymbol{x}}(t) + b_\tau\boldsymbol{a}_{\boldsymbol{x}}(\tau)^\top \frac{\partial\boldsymbol{x}^\theta_{1|\tau}(\boldsymbol{x}_\tau)}{\boldsymbol{x}_\tau}\,\mathrm{d}\tau, \\ &= \boldsymbol{a}_{\boldsymbol{x}}(t) + \int_s^t a_\tau\boldsymbol{a}_{\boldsymbol{x}}(t) + b_\tau\boldsymbol{a}_{\boldsymbol{x}}(\tau)^\top \frac{\partial\boldsymbol{x}^\theta_{1|\tau}(\boldsymbol{x}_\tau)}{\partial\boldsymbol{x}_\tau}\,\mathrm{d}\tau. \qquad \text{(time-reversal)} \end{aligned} \tag{28}$$

[5] Note that despite $\sigma_t \to 0$ the asymptotic behavior is well-defined [see 2].

Using the technique of exponential integrators we rewrite the integral as

$$\begin{aligned}
\boldsymbol{a}_{\boldsymbol{x}}(s) &= e^{\int_s^t a_u \, \mathrm{d}u} \boldsymbol{a}_{\boldsymbol{x}}(t) + \int_s^t e^{\int_\tau^t a_u \, \mathrm{d}u} b_\tau \boldsymbol{a}_{\boldsymbol{x}}(\tau)^\top \frac{\partial \boldsymbol{x}_{1|\tau}^\theta(\boldsymbol{x}_\tau)}{\partial \boldsymbol{x}_\tau} \, \mathrm{d}\tau, \\
&= \frac{\sigma_t}{\sigma_s} \boldsymbol{a}_{\boldsymbol{x}}(t) + \sigma_t \int_s^t \frac{b_\tau}{\sigma_\tau} \boldsymbol{a}_{\boldsymbol{x}}(\tau)^\top \frac{\partial \boldsymbol{x}_{1|\tau}^\theta(\boldsymbol{x}_\tau)}{\partial \boldsymbol{x}_\tau} \, \mathrm{d}\tau, \\
&= \frac{\sigma_t}{\sigma_s} \boldsymbol{a}_{\boldsymbol{x}}(t) + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{a}_{\boldsymbol{x}}(\gamma)^\top \frac{\partial \boldsymbol{x}_{1|\gamma}^\theta(\boldsymbol{x}_\gamma)}{\partial \boldsymbol{x}_\gamma} \, \mathrm{d}\gamma. \qquad (29)
\end{aligned}$$

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:

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \frac{\sigma_t}{\sigma_s} \boldsymbol{a}_{\boldsymbol{x}}(t) + (\alpha_t - \frac{\sigma_t}{\sigma_s} \alpha_s) \boldsymbol{a}_{\boldsymbol{x}}(s)^\top \frac{\partial \hat{\boldsymbol{x}}_{1|s}(\boldsymbol{x}_s)}{\partial \boldsymbol{x}_s} + \mathcal{O}(h^2). \qquad (30)$$

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

$$\boldsymbol{a}_{\boldsymbol{x}}(t) = \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}. \qquad (31)$$

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

$$\boldsymbol{a}_{\boldsymbol{x}}(t)^{(1)} = \boldsymbol{a}_{\boldsymbol{x}}(1)^\top \frac{\partial \hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} = \nabla_{\boldsymbol{x}_t} \mathcal{L}(\hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t)). \qquad (32)$$

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

□

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. [2, Proposition 4.1] showed that the gradient of the target prediction model is proportional to the variance of the random variable defined by $p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x})$, 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 $\boldsymbol{x}_{1|t}^\theta(\boldsymbol{x})$ w.r.t. $\boldsymbol{x}$ is proportional to the variance of $p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x})$,* i.e.*,*

$$\nabla_{\boldsymbol{x}} \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}) = \frac{\alpha_t}{\sigma_t^2} \mathrm{Var}_{1|t}(\boldsymbol{x}), \qquad (33)$$

where

$$\mathrm{Var}_{1|t}(\boldsymbol{x}) = \mathbb{E}_{p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x})} \left[(\boldsymbol{x}_1 - \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}))(\boldsymbol{x}_1 - \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}))^\top \right]. \qquad (34)$$

Remark C.1. This can be written more generally in terms of the (pushforward) differential $D_x \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x})$ where the underlying spaces are smooth manifolds and $\boldsymbol{x}_{1|t}^\theta$ is a smooth map between them [2]. 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 $\boldsymbol{x}_0 \in \mathbb{R}^d$ and let $\boldsymbol{f} \in \mathcal{C}^{1,1}([0,T] \times \mathbb{R}^d; \mathbb{R}^d)$ be uniformly Lipschitz in $\boldsymbol{x}$. Let $\boldsymbol{x} : [0,T] \to \mathbb{R}^d$ be the unique solution to*

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)). \qquad (35)$$

Let $\Phi_{s,t}(\boldsymbol{x})$, $s,t \in [0,T]$ denote the flow associated with Equation (35). Then let $\boldsymbol{J}_s(t) \coloneqq \nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x})$ denote the Jacobian matrices, where $\boldsymbol{J}_s : [s,T] \to \mathbb{R}^{d \times d}$ solve the differential equation

$$\boldsymbol{J}_s(s) = \boldsymbol{I}, \qquad \frac{\mathrm{d}\boldsymbol{J}_s}{\mathrm{d}t}(t) = \nabla_{\boldsymbol{x}} \boldsymbol{f}(t, \Phi_{s,t}(\boldsymbol{x}(s)))\boldsymbol{J}_s(t), \tag{36}$$

where $\nabla_{\boldsymbol{x}} \boldsymbol{f}(t,\cdot)$ 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* [19, Theorem 4.4] and *rough differential equations* [19, Theorem 11.3]. Kidger [37, Theorem 5.8] discusses this result for neural ODEs.

C.1 Proof of Theorem C.3

We prove an additional result about the flow of Jacobian matrices for affine Gaussian probability paths which complements our analysis.

Theorem C.3 (Jacobian matrices of affine Gaussian probability paths)**.** *For the standard affine Gaussian probability path with flow model $\Phi^\theta_{s,t}(\boldsymbol{x})$, the Jacobian matrix $\nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x})$ as function of $\boldsymbol{x}$ is given as the solution to*

$$\nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x}) = \frac{\sigma_t}{\sigma_s}\boldsymbol{I} + \sigma_t \int_s^t \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \mathrm{Var}_{1|u}(\Phi^\theta_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) \,\mathrm{d}u, \tag{37}$$

where

$$\mathrm{Var}_{1|t}(\boldsymbol{x}) = \mathbb{E}_{p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x})}\left[(\boldsymbol{x}_1 - \boldsymbol{x}^\theta_{1|t}(\boldsymbol{x}))(\boldsymbol{x}_1 - \boldsymbol{x}^\theta_{1|t}(\boldsymbol{x}))^\top\right]. \tag{38}$$

Remark C.3. From Theorem C.3 we observe the Jacobian-vector product $\nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x})^\top \boldsymbol{v}$ corresponds to an integral of covariance projections applied to $\boldsymbol{v}$.[6]

This proof follows a similar technique to that used by Blasingame and C. Liu [4] 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:

$$\nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x}) = \boldsymbol{I} + \int_s^t \nabla_{\boldsymbol{x}_u} \boldsymbol{u}^\theta_u(\Phi^\theta_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) \,\mathrm{d}u. \tag{39}$$

Now recall the definition of the marginal vector field in terms of the target prediction model (*cf.* Equation (13)) which we use to rewrite Equation (39) as

$$\begin{aligned}\nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x}) &= \boldsymbol{I} + \int_s^t \nabla_{\boldsymbol{x}_u} a_u \Phi^\theta_{s,u}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}_u} b_u \boldsymbol{x}^\theta_{1|u}(\Phi^\theta_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) \,\mathrm{d}u, \\ &\overset{(i)}{=} \boldsymbol{I} + \int_s^t a_u \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) + b_u \nabla_{\boldsymbol{x}_u} \boldsymbol{x}^\theta_{1|u}(\Phi^\theta_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) \,\mathrm{d}u,\end{aligned} \tag{40}$$

where (i) holds by $\nabla_{\boldsymbol{x}_u} \Phi^\theta_{s,u}(\boldsymbol{x}) = \boldsymbol{I}$. Next we can make use of the popular technique of *exponential integrators* to simplify Equation (39) in combination with Equation (13). Thus, the integral equation in Equation (40) becomes

$$\nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x}) = \Lambda_a(s,t)\boldsymbol{I} + \int_s^t \Lambda_a(u,t) b_u \nabla_{\boldsymbol{x}_u} \boldsymbol{x}^\theta_{1|u}(\Phi^\theta_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) \,\mathrm{d}u, \tag{41}$$

where $\Lambda_a(s,t) \coloneqq \exp \int_s^t a_u \,\mathrm{d}u$ is the integrating factor. This simplifies to $\Lambda_a(s,t) = \sigma_t/\sigma_s$. Using this, Equation (41) can be simplified to

$$\nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x}) = \frac{\sigma_t}{\sigma_s}\boldsymbol{I} + \sigma_t \int_s^t \frac{b_u}{\sigma_u} \nabla_{\boldsymbol{x}_u} \boldsymbol{x}^\theta_{1|u}(\Phi^\theta_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^\theta_{s,u}(\boldsymbol{x}) \,\mathrm{d}u. \tag{42}$$

[6] Readers familiar with the work of Ben-Hamu et al. [2] may notice some similarities between our result Theorem C.3 and Ben-Hamu et al. [2, Theorem 4.2]. We discuss this more in Remark C.4.

Now we can apply Lemma C.1 to further simplify Equation (42) to find

$$\nabla_{\boldsymbol{x}} \Phi^{\theta}_{s,t}(\boldsymbol{x}) = \frac{\sigma_t}{\sigma_s}\boldsymbol{I} + \sigma_t \int_s^t \frac{\alpha_u}{\sigma_u^3} b_u \mathrm{Var}_{1|u}(\Phi^{\theta}_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^{\theta}_{s,u}(\boldsymbol{x}) \, \mathrm{d}u. \tag{43}$$

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

$$\begin{aligned} b_t \frac{\alpha_t}{\sigma_t^3} &= \left(\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t} \right) \frac{\alpha_t}{\sigma_t^3}, \\ &= \frac{\dot{\alpha}_t \sigma_t - \dot{\sigma}_t \alpha_t}{\sigma_t^3} \frac{\alpha_t}{\sigma_t^2}, \\ &\overset{(i)}{=} \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\alpha_t}{\sigma_t} \right] \frac{\alpha_t}{\sigma_t} \frac{1}{\sigma_t}, \\ &\overset{(ii)}{=} \dot{\gamma}_t \frac{\gamma_t}{\sigma_t}, \end{aligned} \tag{44}$$

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

$$\nabla_{\boldsymbol{x}} \Phi^{\theta}_{s,t}(\boldsymbol{x}) = \frac{\sigma_t}{\sigma_s}\boldsymbol{I} + \sigma_t \int_s^t \dot{\gamma}_u \frac{\gamma_u}{\sigma_u} \mathrm{Var}_{1|u}(\Phi^{\theta}_{s,u}(\boldsymbol{x})) \nabla_{\boldsymbol{x}} \Phi^{\theta}_{s,u}(\boldsymbol{x}) \, \mathrm{d}u. \tag{45}$$

□

Remark C.4. Readers familiar with the work of Ben-Hamu et al. [2] may notice some similarities between our result Theorem C.3 and Ben-Hamu et al. [2, 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 C.3 is closely related to Ben-Hamu et al. [2, Theorem 4.2] which we restate below within the context of our notational conventions.[7]

Theorem C.4. *For the standard affine Gaussian probability path, the differential of $\Phi^{\theta}_{0,1}(\boldsymbol{x})$ as of function of $\boldsymbol{x}$ is*

$$\nabla_{\boldsymbol{x}} \Phi^{\theta}_{0,1}(\boldsymbol{x}) = \sigma_1 \mathcal{T} \exp\left[\int_0^1 \frac{1}{2} \dot{\gamma}_t^2 \mathrm{Var}_{1|t}(\boldsymbol{x}) \, \mathrm{d}t \right], \tag{46}$$

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

The time-ordered exponential[8] [24] is defined as

$$\begin{aligned} \mathcal{T} \exp\left[\int_t^1 \boldsymbol{A}(s) \, \mathrm{d}s \right] &= \sum_{n=0}^{\infty} \frac{1}{n!} \int_t^1 \mathrm{d}s_1 \cdots \int_t^1 \mathrm{d}s_n \;\; \mathcal{T}\{\boldsymbol{A}(s_1) \ldots \boldsymbol{A}(s_n)\}, \\ &= \sum_{n=0}^{\infty} \int_t^1 \mathrm{d}s_1 \int_t^{s_1} \mathrm{d}s_2 \cdots \int_t^{s_{n-1}} \mathrm{d}s_n \;\; \boldsymbol{A}(s_1)\boldsymbol{A}(s_2) \ldots \boldsymbol{A}(s_n), \end{aligned} \tag{47}$$

and the solution can be found the Dyson series [64] or Magnus expansion [51], which are truncated in practice. The meta-operator $\mathcal{T}$ denotes the time-ordering [16], *e.g.*, consider the time-ordering of two operators $\boldsymbol{A}, \boldsymbol{B}$:

$$\mathcal{T}\{\boldsymbol{A}(s_1)\boldsymbol{B}(s_2)\} \coloneqq \begin{cases} \boldsymbol{A}(s_1)\boldsymbol{B}(s_2) & \text{if } s_1 > s_2, \\ \pm \boldsymbol{B}(s_2)\boldsymbol{A}(s_1) & \text{otherwise.} \end{cases} \tag{48}$$

For more details we refer the reader to Weinberg [83].

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

[8] This is closely related to the *Peano-Baker series* [see 18, Section 7.5].

C.2 Dynamics of gradient guidance

We state this more formally below in Proposition C.5.

Proposition C.5 (Dynamics of gradient guidance)**.** *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\boldsymbol{x}$ at some time $t \in [0,1]$ in the direction of the gradient $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\Phi_{t,1}^{\theta}(\boldsymbol{x})\right)$ is given by*

$$\delta_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x})^{\top} \nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1). \tag{49}$$

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

> Performing gradient guidance with $\mathcal{L}$ at time $t < 1$ amounts to guidance which follows the target distribution $p(\boldsymbol{X}_1)$ by projecting $\nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1)$ 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 (49) we know that applying the gradient at earlier timesteps causes the initial gradient $\nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1)$ 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 $\boldsymbol{x}_1$ change when $\boldsymbol{x}$ is updated with our greedy guidance strategy?

C.3 Proof of Proposition C.5

We restate Proposition C.5 below.

Proposition C.5 (Dynamics of gradient guidance)**.** *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\boldsymbol{x}$ at some time $t \in [0,1]$ in the direction of the gradient $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\Phi_{t,1}^{\theta}(\boldsymbol{x})\right)$ is given by*

$$\delta_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x})^{\top} \nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1). \tag{49}$$

Proof. This can be shown from a straightforward derivation:

$$\begin{aligned}
\delta_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) &\overset{(i)}{=} \frac{\mathrm{d}}{\mathrm{d}\eta}\bigg|_{\eta=0} \Phi_{t,1}^{\theta}\left(\boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} \mathcal{L}\left(\Phi_{t,1}^{\theta}(\boldsymbol{x})\right)\right), \\
&\overset{(ii)}{=} -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}\left(\boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} \mathcal{L}\left(\Phi_{t,1}^{\theta}(\boldsymbol{x})\right)\right) \nabla_{\boldsymbol{x}} \mathcal{L}\left(\Phi_{t,1}^{\theta}(\boldsymbol{x})\right)\bigg|_{\eta=0}, \\
&= -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \mathcal{L}\left(\Phi_{t,1}^{\theta}(\boldsymbol{x})\right), \\
&\overset{(iii)}{=} -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x})^{\top} \nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1),
\end{aligned} \tag{50}$$

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

C.4 Proof of Proposition 3.3

We restate Proposition 3.3 below.

Proposition 3.3 (Dynamics of greedy gradient guidance)**.** *Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\boldsymbol{x}$ at some time $t \in [0,1]$ in the direction of the gradient $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})\right)$ is given by*

$$\delta_{\boldsymbol{x}}^{\mathcal{G}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \Phi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})^{\top} \nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1). \tag{8}$$

Proof. This can be shown from a straightforward derivation:

$$\begin{aligned}
\delta^{\mathcal{G}}_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) &\overset{(i)}{=} \frac{\mathrm{d}}{\mathrm{d}\eta}\bigg|_{\eta=0} \Phi^{\theta}_{t,1}\left(\boldsymbol{x} - \eta\nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})\right)\right), \\
&\overset{(ii)}{=} -\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}\left(\boldsymbol{x} - \eta\nabla_{\boldsymbol{x}}\mathcal{L}\left(\Phi^{\theta}_{t,1}(\boldsymbol{x})\right)\right)\nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})\right)\bigg|_{\eta=0}, \\
&= -\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})\nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})\right), \\
&\overset{(iii)}{=} -\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})\nabla_{\boldsymbol{x}}\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})^{\top}\nabla_{\boldsymbol{x}_1}\mathcal{L}(\boldsymbol{x}_1),
\end{aligned} \tag{51}$$

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

We note an interesting corollary below.

Corollary C.5.1 (Dynamics of gradient vs greedy guidance)**.** *The difference between the dynamics of gradient guidance in Proposition C.5 and greedy gradient guidance in Proposition 3.3 for a point $\boldsymbol{x}$ at time t with guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$ is*

$$\left\|\delta_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) - \delta^{\mathcal{G}}_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})\right\| = \left\|\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})\left(\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}}\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})\right)^{\top}\nabla_{\boldsymbol{x}_1}\mathcal{L}(\boldsymbol{x}_1)\right\|. \tag{52}$$

C.5 Proof of Theorem C.6

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

Theorem C.6 (Dynamics of gradient vs greedy guidance)**.** *The difference between the dynamics of gradient guidance in Proposition C.5 and greedy gradient guidance in Proposition 3.3 for a point $\boldsymbol{x}$ at time t with guidance function $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^d)$ is bounded by $\mathcal{O}(h^2)$ where $h \coloneqq \gamma_1 - \gamma_t$,* i.e.,

$$\left\|\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}}\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})\right\| = \mathcal{O}(h^2). \tag{53}$$

Proof. From Corollary C.5.1 it is clear that the difference between $\delta_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})$ and $\delta^{\mathcal{G}}_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})$ amounts to the difference between the true gradient and gradient of the target prediction model. Recall Theorem C.3 which enables to write the gradient as the solution to an integral equation:

$$\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) = \frac{\sigma_1}{\sigma_t}\boldsymbol{I} + \sigma_1\int_t^1 \dot{\gamma}_u\frac{\gamma_u}{\sigma_u}\mathrm{Var}_{1|u}(\Phi^{\theta}_{s,u}(\boldsymbol{x}))\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,u}(\boldsymbol{x})\,\mathrm{d}u. \tag{54}$$

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

$$\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) = \sigma_1\int_t^1 \dot{\gamma}_u\frac{\gamma_u}{\sigma_u}\mathrm{Var}_{1|u}(\Phi^{\theta}_{t,u}(\boldsymbol{x}))\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,u}(\boldsymbol{x})\,\mathrm{d}u, \tag{55}$$

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

$$\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) = \sigma_1\int_{\gamma_t}^{\gamma_1} \frac{\gamma}{\sigma_\gamma}\mathrm{Var}_{1|\gamma}(\Phi^{\theta}_{\gamma_t,\gamma}(\boldsymbol{x}))\nabla_{\boldsymbol{x}}\Phi^{\theta}_{\gamma_t,\gamma}(\boldsymbol{x})\,\mathrm{d}\gamma. \tag{56}$$

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

$$\frac{\gamma}{\sigma_\gamma}\mathrm{Var}_{1|\gamma}(\Phi^{\theta}_{\gamma_t,\gamma}(\boldsymbol{x}))\nabla_{\boldsymbol{x}}\Phi^{\theta}_{\gamma_t,\gamma}(\boldsymbol{x}) = \frac{\gamma_t}{\sigma_t}\mathrm{Var}_{1|t}(\boldsymbol{x}) + \mathcal{O}(\gamma - \gamma_t). \tag{57}$$

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 (57) into Equation (56)

yields

$$\begin{aligned}\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) &= \sigma_1 \int_{\gamma_t}^{\gamma_1} \gamma \frac{1}{\sigma_t}\mathrm{Var}_{1|t}(\boldsymbol{x}) + \mathcal{O}(\gamma-\gamma_t)\,\mathrm{d}\gamma,\\ &\overset{(i)}{=} \sigma_1 \frac{\gamma_t}{\sigma_t}\mathrm{Var}_{1|t}(\boldsymbol{x}) \int_{\gamma_t}^{\gamma_1} \mathrm{d}\gamma + \mathcal{O}(h^2),\\ &= \sigma_1 \frac{\gamma_t}{\sigma_t}\mathrm{Var}_{1|t}(\boldsymbol{x})\,(\gamma_1-\gamma_t) + \mathcal{O}(h^2), \end{aligned} \tag{58}$$

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

$$\begin{aligned}\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) &= \sigma_1 \frac{\alpha_t}{\sigma_t^2}\,(\gamma_1-\gamma_t)\,\mathrm{Var}_{1|t}(\boldsymbol{x}) + \mathcal{O}(h^2),\\ &= \sigma_1 \frac{\alpha_t}{\sigma_t^2}\left(\frac{\alpha_1}{\sigma_1} - \frac{\alpha_t}{\sigma_t}\right)\mathrm{Var}_{1|t}(\boldsymbol{x}) + \mathcal{O}(h^2),\\ &= \frac{\alpha_t}{\sigma_t^2}\left(\alpha_1 - \sigma_1\frac{\alpha_t}{\sigma_t}\right)\mathrm{Var}_{1|t}(\boldsymbol{x}) + \mathcal{O}(h^2),\\ &\overset{(i)}{=} \frac{\alpha_t}{\sigma_t^2}\mathrm{Var}_{1|t}(\boldsymbol{x}) + \mathcal{O}(h^2), \end{aligned} \tag{59}$$

where (i) holds by the boundary conditions of the schedule (*cf*. Equation (11)). Now recall Lemma C.1 which states:

$$\nabla_{\boldsymbol{x}}\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}) = \frac{\alpha_t}{\sigma_t^2}\mathrm{Var}_{1|t}(\boldsymbol{x}). \tag{60}$$

Thus from Equation (59) and Equation (60) it is easy to see that

$$\left\|\nabla_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}}\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})\right\| = \mathcal{O}(h^2), \tag{61}$$

holds and thus

$$\left\|\delta_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x}) - \delta^{\mathcal{G}}_{\boldsymbol{x}}\Phi^{\theta}_{t,1}(\boldsymbol{x})\right\| = \mathcal{O}(h^2). \tag{62}$$

□

C.6 Proof of Theorem 3.4

We restate Theorem 3.4 below.

Theorem 3.4 (Greedy convergence)**.** *For affine probability paths, if there exists a sequence of states $\boldsymbol{x}_t^{(n)}$ at time t such that it converges to the locally optimal solution $\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}_t^{(n)}) \to \boldsymbol{x}_1^*$. Then the solution, $\Phi^{\theta}_{t,1}(\boldsymbol{x}_t^{(n)})$, converges to a neighborhood of size $\mathcal{O}(h^2)$ centered at $\boldsymbol{x}_1^*$.*

Proof. By Assumption B.2, we can take a $(k-1)$-th order Taylor expansion around γ_t of the flow in Equation (6) to obtain

$$\begin{aligned}\Phi^{\theta}_{t,1}(\boldsymbol{x}_t) &= \frac{\sigma_1}{\sigma_t}\boldsymbol{x}_t + \sigma_1 \int_{\gamma_t}^{\gamma_1} \sum_{n=0}^{k-1} \frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}^{\theta}_{1|\gamma}(\boldsymbol{x}_\gamma)\right]_{\gamma=\gamma_t} \frac{(\gamma-\gamma_t)^n}{n!}\,\mathrm{d}\gamma + \mathcal{O}(h^{k+1}),\\ &= \frac{\sigma_1}{\sigma_t}\boldsymbol{x}_t + \sigma_1 \sum_{n=0}^{k-1} \frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}^{\theta}_{1|\gamma}(\boldsymbol{x}_\gamma)\right]_{\gamma=\gamma_t} \int_{\gamma_t}^{\gamma_1} \frac{(\gamma-\gamma_t)^n}{n!}\,\mathrm{d}\gamma + \mathcal{O}(h^{k+1}),\\ &= \frac{\sigma_1}{\sigma_t}\boldsymbol{x}_t + \sigma_1 \sum_{n=0}^{k-1} \frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}^{\theta}_{1|\gamma}(\boldsymbol{x}_\gamma)\right]_{\gamma=\gamma_t} \frac{h^{n+1}}{(n+1)!} + \mathcal{O}(h^{k+1}), \end{aligned} \tag{63}$$

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

$$\Phi^{\theta}_{t,1}(\boldsymbol{x}_t) = \frac{\sigma_1}{\sigma_t}\boldsymbol{x}_n + \sigma_1\hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t)h + \mathcal{O}(h^2), \tag{64}$$

$$= \frac{\sigma_1}{\sigma_t}\boldsymbol{x}_n + (\alpha_1 - \frac{\sigma_1\alpha_t}{\sigma_t})\hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t) + \mathcal{O}(h^2). \tag{65}$$

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

$$\Phi^{\theta}_{t,1}(\boldsymbol{x}_t) = \hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t) + \mathcal{O}(h^2), \tag{66}$$

which is equivalent to

$$\left\|\Phi^{\theta}_{t,1}(\boldsymbol{x}_t) - \hat{\boldsymbol{x}}_{1|t}(\boldsymbol{x}_t)\right\| \leq C_1 h^2, \tag{67}$$

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

$$\left\|\Phi^{\theta}_{t,1}(\boldsymbol{x}^{(n)}_t) - \boldsymbol{x}^*_1\right\| \leq \left\|\Phi^{\theta}_{t,1}(\boldsymbol{x}^{(n)}_t) - \boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}^{(n)}_t)\right\| + \left\|\boldsymbol{x}^*_1 - \boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}^{(n)}_t)\right\| < \underbrace{\epsilon + C_1 h^2}_{:=C_2}. \tag{68}$$

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

D Beyond Euler

In this section we provide the full proofs and derivations for Section 3.2 in the main paper.

D.1 Proof of Theorem 3.5

Before showing Theorem 3.5 we show a more general version below.

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

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{u}_\theta(t, \boldsymbol{x}(t)), \tag{69}$$

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

$$\left\|\nabla_{\boldsymbol{x}}\Phi^{\theta}_{s,t}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}}\boldsymbol{\Phi}_{s,t}(\boldsymbol{x})\right\| = \mathcal{O}(h^{\alpha+1}). \tag{70}$$

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

$$\boldsymbol{u}_{n,j} = \boldsymbol{u}_\theta\left(t_n + c_j h, \boldsymbol{x}_n + h\sum_{i=1}^{j} a_{j,i}\boldsymbol{u}_{n,i}\right), \qquad j = 1, 2, \ldots, k \tag{71}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\sum_{j=1}^{k} b_j \boldsymbol{u}_{n,j}, \tag{72}$$

where $a_{j,i}, b_j, c_j$ are all given via the *Butcher Tableau* [75, Section 6.1.4]. Now, we consider a single step from time s to time t with initial value $\boldsymbol{x}$ and step size $h := t - s$. Then, the gradient is

$$\begin{aligned}\nabla_{\boldsymbol{x}}\boldsymbol{\Phi}_{s,t}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}}\boldsymbol{x} + h\sum_{j=1}^{k} b_j \nabla_{\boldsymbol{x}}\boldsymbol{u}_\theta\left(s + c_j h, \boldsymbol{x} + h\sum_{i=1}^{j} a_{j,i}\boldsymbol{u}_i\right), \\ &= \boldsymbol{I} + h\sum_{j=1}^{k} b_j \left[\nabla_{\hat{\boldsymbol{x}}_j}\boldsymbol{u}_\theta(s + c_j h, \hat{\boldsymbol{x}}_j)\left(\boldsymbol{I} + h\sum_{i=1}^{j} a_{j,i}\nabla_{\boldsymbol{x}}\boldsymbol{u}_i\right)\right],\end{aligned} \tag{73}$$

where we let

$$\hat{\boldsymbol{x}}_j = \boldsymbol{x} + h\sum_{i=1}^{j} a_{j,i}\boldsymbol{u}_i. \tag{74}$$

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

$$\boldsymbol{J}_s(s) = \boldsymbol{I}, \qquad \frac{\mathrm{d}\boldsymbol{J}_s}{\mathrm{d}t}(t) = \nabla_{\boldsymbol{x}} \boldsymbol{u}_\theta(t, \Phi_{s,t}(\boldsymbol{x}))\boldsymbol{J}_s(t). \tag{75}$$

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

$$\boldsymbol{U}_j = \boldsymbol{I} + h \sum_{j=1}^{k} b_j \left[\nabla_{\hat{\boldsymbol{x}}_j} \boldsymbol{u}_\theta \left(s + c_j h, \boldsymbol{x} + \hat{\boldsymbol{x}}_j \right) \left(\boldsymbol{I} + h \sum_{i=1}^{j} a_{j,i} \nabla_{\boldsymbol{x}} \boldsymbol{u}_i \right) \right]. \tag{76}$$

Clearly, Equation (76) and Equation (73) are equivalent. Now as the underlying numerical solver has local truncation error $\mathcal{O}(h^{\alpha+1})$ we find that

$$\left\| \nabla_{\boldsymbol{x}} \Phi^\theta_{s,t}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \boldsymbol{\Phi}_{s,t}(\boldsymbol{x}) \right\| = \mathcal{O}(h^{\alpha+1}). \tag{77}$$

□

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 [58, Appendix A] made similar observations; however, it is for only of the case of Euler.

Remark D.4. This result bears some resembelmance to work done on using high-resolution ODEs for modelling the continuous-time approximation of momentum gradient descent [67].

Theorem 3.5 (Truncation error of single-step gradients)**.** *Let $\boldsymbol{\Phi}$ be an explict Runge-Kutta solver of order $\alpha > 0$ of a flow model with flow $\Phi^\theta_{s,t}(\boldsymbol{x})$. Then for any $t \in [0, 1]$,*

$$\left\| \nabla_{\boldsymbol{x}} \Phi^\theta_{t,1}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \boldsymbol{\Phi}_{t,1}(\boldsymbol{x}) \right\| = \mathcal{O}(h^{\alpha+1}), \tag{9}$$

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 $\boldsymbol{x}_t^{(n)}$ at time t such that it converges to the locally optimal solution $\boldsymbol{\Phi}^\theta_{t,1}(\boldsymbol{x}_t^{(n)}) \to \boldsymbol{x}_1^*$. Then solution, $\Phi^\theta_{t,1}(\boldsymbol{x}_t^{(n)})$, converges to a neighborhood of size $\mathcal{O}(h^{\alpha+1})$ centered at $\boldsymbol{x}_1^*$.*

Proof. This follows as a straightforward derivation from Theorem D.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 3.1.

Proposition D.2 (Reparameterized for the target prediction model of affine probability paths)**.** *The ODE governed by the vector field in Equation (12) can be reparameterized as*

$$\frac{\mathrm{d}\boldsymbol{y}_\gamma}{\mathrm{d}\gamma} = \sigma_0 \boldsymbol{x}^\theta_{1|\zeta} \left(\frac{\sigma_\gamma}{\sigma_0} \boldsymbol{y}_\zeta \right), \tag{78}$$

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

Proof. The ODE governed by the vector field in Equation (12) can be written as

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = a_t \boldsymbol{x}_t + b_t \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}_t). \tag{79}$$

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

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[e^{\int_0^t -a_u \,\mathrm{d}u}\boldsymbol{x}_t\right] = e^{\int_0^t -a_u \,\mathrm{d}u} b_t \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}_t). \tag{80}$$

The exponential term can be simplified to

$$e^{\int_0^t -a_u \,\mathrm{d}u} = \frac{\sigma_0}{\sigma_t}. \tag{81}$$

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

$$\frac{\mathrm{d}\boldsymbol{y}_t}{\mathrm{d}t} = \frac{\sigma_0}{\sigma_t} b_t \boldsymbol{x}_{1|t}^\theta\left(\frac{\sigma_t}{\sigma_0}\boldsymbol{y}_t\right). \tag{82}$$

Next, recall that $b_t/\sigma_t = \dot{\gamma}_t$ (*cf*. Equation (21)) which enables a change of integration variable:

$$\frac{\mathrm{d}\boldsymbol{y}_\gamma}{\mathrm{d}\gamma} = \sigma_0 \boldsymbol{x}_{1|\gamma}^\theta\left(\frac{\sigma_\gamma}{\sigma_0}\boldsymbol{y}_\gamma\right). \tag{83}$$

□

Remark D.5. Recall that, often, for affine probability paths we let $\sigma_0 = 1$, further simplifying Proposition D.2 to

$$\frac{\mathrm{d}\boldsymbol{y}_\gamma}{\mathrm{d}\gamma} = \boldsymbol{x}_{1|\gamma}^\theta\left(\sigma_\gamma \boldsymbol{y}_\gamma\right). \tag{84}$$

Remark D.6. Proposition D.2 is a tangential result to the prior result of Pan, Liew, et al. [59, 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.[9]

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. [9], 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

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{u}_\theta(t, \boldsymbol{x}(t)), \tag{85}$$

and assume it is defined on the interval $[0, T]$ and the flow model statifies the usual regularity conditions. Then, the continuous adjoint equations [37, Theorem 5.2] are:

$$\begin{aligned} \boldsymbol{a}_{\boldsymbol{x}}(T) &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_T}, & \frac{\mathrm{d}\boldsymbol{a}_{\boldsymbol{x}}}{\mathrm{d}t}(t) &= -\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{u}_\theta}{\partial \boldsymbol{x}}(t, \boldsymbol{x}(t)), \\ \boldsymbol{a}_\theta(T) &= \mathbf{0}, & \frac{\mathrm{d}\boldsymbol{a}_\theta}{\mathrm{d}t}(t) &= -\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{u}_\theta}{\partial \theta}(t, \boldsymbol{x}(t)), \end{aligned} \tag{86}$$

where $\boldsymbol{a}_{\boldsymbol{x}}(t) \coloneqq \partial\mathcal{L}/\partial\boldsymbol{x}(t)$ and $\boldsymbol{a}_\theta(0) \coloneqq \partial\mathcal{L}/\partial\theta$.

[9] Technically forward-time due to the conventions of diffusion models.

Table 3: 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-trival 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 $\{\boldsymbol{x}_{t_i}\}_{i=1}^N$ and the backward trajectory $\{\tilde{\boldsymbol{x}}_{t_i}\}_{i=1}^N$ when performing the backwards solve. *E.g.*, consider an explicit Euler scheme

$$\boldsymbol{x}_{t_{i+1}} = \boldsymbol{x}_{t_i} + (t_{i+1} - t_i)\boldsymbol{u}_\theta(t_i, \boldsymbol{x}_{t_i}). \tag{87}$$

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

$$\tilde{\boldsymbol{x}}_{t_i} = \tilde{\boldsymbol{x}}_{t_{i+1}} + (t_i - t_{i+1})\boldsymbol{u}_\theta(t_{i+1}, \tilde{\boldsymbol{x}}_{t_{i+1}}). \tag{88}$$

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 7, 38, 54] which guarantees that the forward and backward trajectory match *perfectly*. Another option is to store the forward trajectory $\{\boldsymbol{x}_{t_i}\}_{i=1}^N$ in memory and use *interpolated adjoints* if the backward timesteps do not perfectly align with the forward timesteps [see 39].

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 25, 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 8]. For $\boldsymbol{x}_t$ this is given by $\frac{\partial \boldsymbol{u}_\theta}{\partial \boldsymbol{x}}$ and for $\boldsymbol{a}_{\boldsymbol{x}}$ this is given by

$$\frac{\partial}{\partial \boldsymbol{a}_{\boldsymbol{x}}}\left(-\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{u}_\theta}{\partial \boldsymbol{x}}(t, \boldsymbol{x}(t))\right) = -\frac{\partial \boldsymbol{u}_\theta}{\partial \boldsymbol{x}}(t, \boldsymbol{x}(t)). \tag{89}$$

Clearly, the Jacobians for $\boldsymbol{a}_{\boldsymbol{x}}$ and $\boldsymbol{x}_t$ solved in reverse-time are identical, meaning the stability of the backward solve is pushed onto the solve for the adjoint state [see 37, 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 $[-i, i]$ [38]. Recent work by McCallum and Foster [54], 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 [22, 23, 76] 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 [4] and Domingo-Enrich et al. [14] 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 3 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 [48, 80] have explored the guidance from the perspective of optimal control [41]. In essence this technique first injects an additional control signal, $\boldsymbol{z} \in \mathcal{C}^1(\mathbb{R}; \mathbb{R}^d)$, to the vector field, $\boldsymbol{u}_t^\theta$, such that

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{u}_t^\theta(\boldsymbol{x}_t) + \boldsymbol{z}(t). \tag{90}$$

Thus, instead of optimizing $\{\boldsymbol{x}_t\}_{t\in[0,t]}$ directly, this control signal can instead be optimized, serving as one of the key insights in [48, 80]. *I.e.*, suppose we have a neural ODE with vector field $\boldsymbol{u}_t^\theta(\boldsymbol{x})$, then we can write the optimization problem as

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & \mathcal{L}(\boldsymbol{x}_T) + \lambda \int_0^T \|\boldsymbol{z}(t)\| \,\mathrm{d}t, \\ \text{s.t.} \quad & \boldsymbol{x}_T = \boldsymbol{x}_0 + \int_0^T \boldsymbol{u}_t^\theta(\boldsymbol{x}_t) + \boldsymbol{z}(t) \,\mathrm{d}t. \end{aligned} \tag{91}$$

The next natural question then is to ask about the behavior of a greedy strategy applied to $\boldsymbol{z}(t)$. To simplify the analysis, we now consider a control signal applied to the posterior model $\boldsymbol{x}_{1|t}^\theta$ such that it is replaced by $\boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}_t) + \boldsymbol{z}(t)$ which amounts to simply rescaling $\boldsymbol{z}(t)$ from Equation (90) with b_t. From this construction, it should be clear that the greedy gradient for the control signal is merely $\nabla_{\tilde{\boldsymbol{x}}_1}\mathcal{L}(\tilde{\boldsymbol{x}}_1)$. 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 [5]; 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 $\boldsymbol{a}_{\boldsymbol{z}}(t) \coloneqq \partial\mathcal{L}/\partial\boldsymbol{z}(t)$. In Theorem F.1 we show that this gradient is simply an integral of the adjoint state $\boldsymbol{a}_{\boldsymbol{x}}(t)$.

Theorem F.1 (Continuous adjoint equations for the control term)**.** *Let $\boldsymbol{u}_t^\theta \in \mathcal{C}^{1,1}([0,T] \times \mathbb{R}^{d_x}; \mathbb{R}^{d_x})$ be a parameterization of some time-dependent vector field of a neural ODE that is Lipschitz continuous in its second argument, and let $\boldsymbol{z} \in \mathcal{C}^1([0,1]; \mathbb{R}^d)$ be an additional control signal such that the new dynamics are given by Equation (90). Let $\boldsymbol{a}_{\boldsymbol{z}}(t) \coloneqq \partial\mathcal{L}/\partial\boldsymbol{z}(t)$, then*

$$\boldsymbol{a}_{\boldsymbol{z}}(t) = -\int_T^t \boldsymbol{a}_{\boldsymbol{x}}(s) \,\mathrm{d}s. \tag{92}$$

Our proof follows the structure of the modern proof of Pontryagin's original result [61] presented by [9]; and is similar to the form used by Blasingame and C. Liu [4, Theorem 2.2].

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

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{z}\end{bmatrix}(t) = \boldsymbol{f}_{\text{aug}} = \begin{bmatrix}\boldsymbol{u}_\theta(t, \boldsymbol{x}(t)) + \boldsymbol{z}(t)\\ \frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t}(t)\end{bmatrix}, \tag{93}$$

and the augmented adjoint state as

$$\boldsymbol{a}_{\text{aug}}(t) \coloneqq \begin{bmatrix}\boldsymbol{a}_{\boldsymbol{x}}\\ \boldsymbol{a}_{\boldsymbol{z}}\end{bmatrix}(t). \tag{94}$$

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

$$\frac{\partial \boldsymbol{f}_{\text{aug}}}{\partial[\boldsymbol{x}, \boldsymbol{z}]} = \begin{bmatrix}\frac{\partial \boldsymbol{u}_\theta(t, \boldsymbol{x}(t))}{\partial \boldsymbol{x}} & \mathbf{1}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}. \tag{95}$$

The evolution of the adjoint state is given by

$$\frac{\mathrm{d}\boldsymbol{a}_{\text{aug}}}{\mathrm{d}t}(t) = -\begin{bmatrix}\boldsymbol{a}_{\boldsymbol{x}} & \boldsymbol{a}_{\boldsymbol{z}}\end{bmatrix}(t)\frac{\partial \boldsymbol{f}_{\text{aug}}}{\partial[\boldsymbol{x},\boldsymbol{z}]}(t). \tag{96}$$

Therefore, $\boldsymbol{a}_{\boldsymbol{u}}(t)$ evolves with

$$\boldsymbol{a}_{\boldsymbol{u}}(T) = \mathbf{0}, \qquad \frac{\mathrm{d}\boldsymbol{a}_{\boldsymbol{u}}}{\mathrm{d}t}(t) = -\boldsymbol{a}_{\boldsymbol{x}}(t), \tag{97}$$

thereby finishing the proof. □

G A brief introduction to inverse problems

Inverse problems cover a large class of scientific problems [10] that encompass scenarios where a partial measurement $\boldsymbol{y}$ is made of $\boldsymbol{x}$. When the mapping $\boldsymbol{x} \mapsto \boldsymbol{y}$ is not an injection, recovering $\boldsymbol{x}$ from $\boldsymbol{y}$ becomes an ill-posed inverse problem. Generally, the relationship between the underlying sample $\boldsymbol{x}$ and the measurement $\boldsymbol{y}$ is given by

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\eta}, \qquad \boldsymbol{y}, \boldsymbol{\eta} \in \mathbb{R}^{d_y}, \boldsymbol{x} \in \mathbb{R}^{d_x}, \tag{98}$$

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

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

More details on these types of problems can be found in Chung, J. Kim, et al. [10], Moufad et al. [55], and B. Zhang et al. [85].

G.1 Inverse problems and diffusion models

Recall that the ODE formulation of diffusion models is just a particular type of affine Gaussian probability path [45]. Following the conventions of the EDM model [34] we write this ODE formulation, known in the literature as the *probability flow ODE*, below in

$$\mathrm{d}\boldsymbol{x}_t = -\dot{\sigma}_t \sigma_t \nabla_{\boldsymbol{x}_t} \log p(\sigma_t, \boldsymbol{x}_t)\, \mathrm{d}t, \tag{99}$$

where $p(\sigma_t, \boldsymbol{x}_t)$ is the joint distribution of $\boldsymbol{x}_t$ at noise level σ_t.[10] *N.B.*, for diffusion models $\mathrm{d}t$ 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 $\nabla_{\boldsymbol{x}_t} \log p(\sigma_t, \boldsymbol{x}_t)$.

One of the insights of Chung, J. Kim, et al. [10] and Y. Song, Sohl-Dickstein, et al. [72] is to apply Bayes' theorem for inverse problems to score-based generative models, *i.e.*,

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})}, \tag{100}$$

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{y}|\boldsymbol{x}). \tag{101}$$

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

$$\nabla_{\boldsymbol{x}_t} \log p(\sigma_t, \boldsymbol{x}_t|\boldsymbol{y}) = \nabla_{\boldsymbol{x}_t} \log p(\sigma_t, \boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t, \sigma_t). \tag{102}$$

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. [10] is to use an approximation of

$$p(\boldsymbol{y}|\boldsymbol{x}_t, \sigma_t) = \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[p(\boldsymbol{y}|\boldsymbol{x}_0, \sigma_0)], \tag{103}$$

via Tweedie's formula [74] to write

$$p(\boldsymbol{y}|\boldsymbol{x}_t, \sigma_t) \approx p(\boldsymbol{y}|\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t], \sigma_0). \tag{104}$$

The approximation error can be quantified by the Jensen gap [10, Theorem 1].

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

H Experimental details

We provide additional details of the experiments performed in Section 4. *N.B.*, for all experiments we used fixed random seeds between the different software components to ensure a fair comparison.

H.1 Inverse image problems

Inverse problems. The inverse problems are implemented in the same way as in B. Zhang et al. [85]. 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 [69]. The measure for the high dynamic range reconstruction problem is defined as

$$\boldsymbol{y} \sim \mathcal{N}(\text{clip}(\alpha \boldsymbol{x}_0, -1, 1), \beta_{\boldsymbol{y}}^2 \boldsymbol{I}), \tag{105}$$

with $\alpha = 2$.

Diffusion model. We make use of the pre-trained diffusion model from Chung, J. Kim, et al. [10], trained on the FFHQ 256×256 dataset. We focus on the probability flow ODE formulation popularized by Karras, Aittala, et al. [34] known as EDM described as

$$\mathrm{d}\boldsymbol{x}_t = -\dot{\sigma}_t \sigma_t \nabla_{\boldsymbol{x}_t} \log p(\sigma_t, \boldsymbol{x}_t) \, \mathrm{d}t. \tag{106}$$

Following Ben-Hamu et al. [2], 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

$$\sigma_n = \left(\sigma_{\max}^{\frac{1}{\rho}} + \frac{n}{N-1} \left(\sigma_{\min}^{\frac{1}{\rho}} - \sigma_{\max}^{\frac{1}{\rho}} \right) \right)^{\rho}. \tag{107}$$

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 [71].

Hyperparameters. Unlike previous works [85] 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 the `torch.optim.SGD` method for each step of the ODE solve. We set $\beta_{\boldsymbol{y}} = 0.05$ for all tasks.

H.2 Molecule generation for QM9

We follow the experimental methodology taken in previous work [2, 80] and follow the conditional generation pipeline used by Hoogeboom et al. [31]. 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 [73] 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. [80], the test time properties were sampled from the whole training set; in contrast to Ben-Hamu et al. [2].

Following Ben-Hamu et al. [2] we used the L-BFGS algorithm [46] with 5 optimizer steps and 5 inner steps with a linear search, in particular we used the stock PyTorch implementation `torch.opt.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 3.3 states that the greedy gradient is scaled by the covariance projection. This effect is lessened as $t \to 1$, 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 [10, 55, 84] and annealed the learning rate. We chose the following simple scheduler:

$$\eta_t = \begin{cases} \eta(1-t) & t > 0.5 \\ 0 & t \leq 0.5 \end{cases}, \tag{108}$$

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

$$\boldsymbol{k}_1 = \boldsymbol{u}_\theta\left(t_n, \boldsymbol{x}_n\right), \tag{109}$$

$$\boldsymbol{k}_2 = \boldsymbol{u}_\theta\left(t_n + \frac{h}{2}, \boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k}_1\right), \tag{110}$$

$$\boldsymbol{k}_3 = \boldsymbol{u}_\theta\left(t_n + \frac{h}{2}, \boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k}_2\right), \tag{111}$$

$$\boldsymbol{k}_4 = \boldsymbol{u}_\theta\left(t_n + h, \boldsymbol{x}_n + h\boldsymbol{k}_3\right), \tag{112}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4). \tag{113}$$

Using the step size $h = 1 - t$ we encountered large stability issues with the $\boldsymbol{k}_4$ term due to being evaluated at the endpoint of the flow model trajectory. We tried a mixed-solver scheme were we would start with Euler and then switch to RK4, but that did not help. We also tried the common diffusion trick of truncated 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 Ralton's third-order method may fair better.

H.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

$$\boldsymbol{x}_1 = \boldsymbol{x}_t + h\boldsymbol{u}_\theta\left(t + \frac{h}{2}, \boldsymbol{x}_t + \frac{h}{2}\boldsymbol{u}_\theta(t, \boldsymbol{x}_t)\right) \tag{114}$$

with step size $h = 1 - t$.[11]

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

$$\boldsymbol{x}_{t+\frac{h}{2}} = \boldsymbol{x}_t + \frac{h}{2}\boldsymbol{u}_\theta(t, \boldsymbol{x}_t), \tag{115}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_{t+\frac{h}{2}} + \frac{h}{2}\boldsymbol{u}_\theta\left(t + \frac{h}{2}, \boldsymbol{x}_{t+\frac{h}{2}}\right), \tag{116}$$

with step sizes $h = 1 - t$.

H.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.

I Further experimental results

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

[11] This is appropriately adjusted for diffusion models with a terminal time of 0.

I.1 Molecule generation for QM9

In Table 4 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 2) the greedy (midpoint) and (2-step Euler) strategies have slightly better stability than DTO.

Table 4: Stability reported in ASP/MSP per property.

Property	α	$\Delta\varepsilon$	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	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					

I.2 Further results on inverse image problems

To put the results from Section 4.1 into context we present some detailed comparisons to other works from the domain of inverse problems with diffussion models, namely:

1. DAPS [85],
2. DPS [10],
3. DDRM [36],
4. DDNM [81],
5. DCDP [42],
6. FPS-SMC [15],
7. DiffPIR [88], and
8. RED-diff [52].

We present the full comparison in Table 5.

Table 5: Additional results for inverse image problems on FFHQ 256×256.

Task	Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)	FID (↓)
Super resolution 4×	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
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
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	$30.63_{\pm 3.13}$	$0.851_{\pm 0.072}$	$6.139_{\pm 0.060}$	42.71
	DPS	$17.64_{\pm 2.97}$	$0.441_{\pm 0.129}$	$0.410_{\pm 0.090}$	104.52
	RED-diff	$15.60_{\pm 4.48}$	$0.398_{\pm 0.195}$	$0.596_{\pm 0.092}$	167.43
	DCDP	28.65 ± 8.09	$0.781_{\pm 0.217}$	$0.203_{\pm 0.196}$	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	$28.29_{\pm 1.77}$	$0.783_{\pm 0.036}$	$0.155_{\pm 0.032}$	49.38
	DPS	$23.39_{\pm 2.01}$	$0.623_{\pm 0.082}$	$0.278_{\pm 0.060}$	91.31
	RED-diff	$30.86_{\pm 0.51}$	$0.795_{\pm 0.028}$	$0.160_{\pm 0.034}$	43.84
	DCDP	$27.92_{\pm 2.64}$	$0.779_{\pm 0.067}$	$0.183_{\pm 0.051}$	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	$27.12_{\pm 3.53}$	$0.752_{\pm 0.041}$	$0.162_{\pm 0.072}$	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

I.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 $\boldsymbol{x}_1$.

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

Figure 7: Sampling trajectory for greedy (midpoint) solving the HDR inverse problem. Top row is midpoint estimate and the bottom row is $\boldsymbol{x}_t$.

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 $\boldsymbol{x}_t$.

I.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.

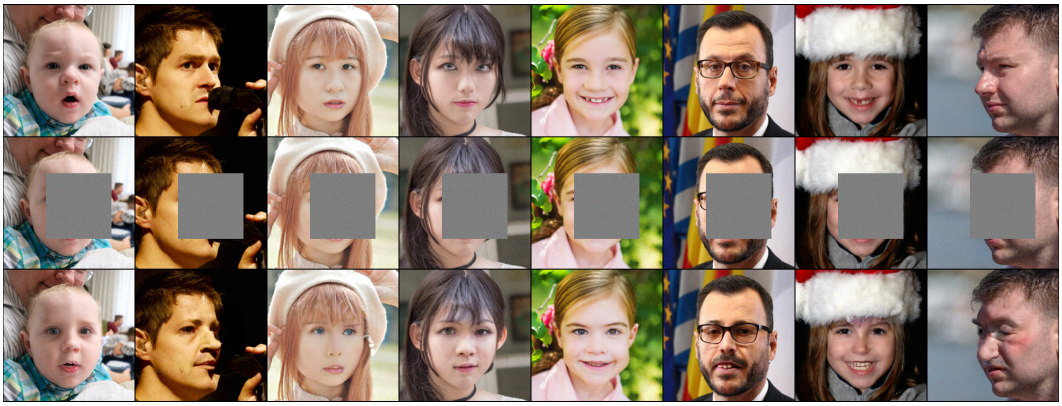

Figure 10: Qualitative visualization of using greedy guidance to solve the super resolution $4\times$ inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.

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.

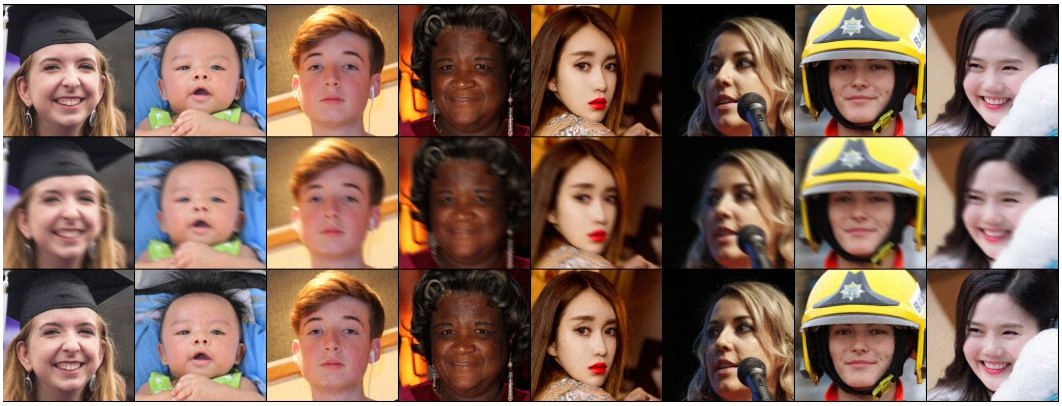

Figure 12: Qualitative visualization of using greedy guidance to solve the motion deblurring inverse problem. Top row is the ground truth, middle row is the measurement, and the bottom row is the reconstruction.

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.

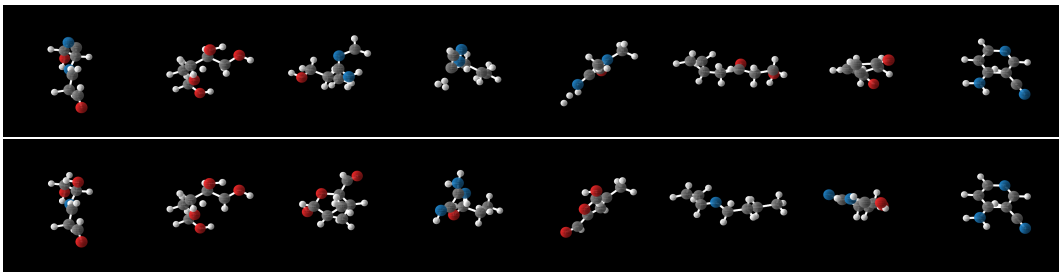

Figure 16: Qualitative visualization of controlled generated molecules for various dipole moments (μ). Top row is generated using a end-to-end guidance with a DTO scheme and the bottom row is generated using posterior guidance.

I.5 More qualitative samples for controlled molecule generation

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

J Discussions

J.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.

J.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*. 55].