AdjointDEIS: Efficient Gradients for Diffusion Models

Zander W. Blasingame
Clarkson University
blasinzw@clarkson.edu

Chen Liu
Clarkson University
cliu@clarkson.edu

Abstract

The optimization of the latents and parameters of diffusion models with respect to some differentiable metric defined on the output of the model is a challenging and complex problem. The sampling for diffusion models is done by solving either the *probability flow* ODE or diffusion SDE wherein a neural network approximates the score function allowing a numerical ODE/SDE solver to be used. However, naïve backpropagation techniques are memory intensive, requiring the storage of all intermediate states, and face additional complexity in handling the injected noise from the diffusion term of the diffusion SDE. We propose a novel family of bespoke ODE solvers to the continuous adjoint equations for diffusion models, which we call *AdjointDEIS*. We exploit the unique construction of diffusion SDEs to further simplify the formulation of the continuous adjoint equations using *exponential integrators*. Moreover, we provide convergence order guarantees for our bespoke solvers. Significantly, we show that continuous adjoint equations for diffusion SDEs actually simplify to a simple ODE. Lastly, we demonstrate the effectiveness of AdjointDEIS for guided generation with an adversarial attack in the form of the face morphing problem. Our code will be released at https://github.com/zblasingame/AdjointDEIS.

1 Introduction

Diffusion models are a large family of state-of-the-art generative models which learn to map samples drawn from white Gaussian noise into the data distribution [1, 2]. These diffusion models have achieved state-of-the-art performance on prominent tasks such as image generation [3–5], audio generation [6, 7], or video generation [8]. Often, the state-of-the-art models are quite large and training them is prohibitively expensive [9]. As such, it is fairly common to adapt a pre-trained model to a specific task for post-training. This way, the generative model can learn new concepts, identities, or tasks without needing to train the whole model [10–12]. Additional work has also proposed algorithms for guiding the generative process of the diffusion models [13, 14].

One method to guide or direct the generative process is to solve an optimization problem w.r.t. some guidance function $\mathcal{L}$ defined on the image space $\mathbb{R}^d$. This guidance function works on the output of the diffusion model and assesses how "good" the output is. However, the diffusion model works by iteratively removing noise until a clean sample is reached. As such, we need to be able to efficiently backpropagate gradients through the entire generative process. As Song et al. [15] showed, the diffusion SDE can be simplified to an associated ODE, and as such, many efficient ODE/SDE solvers have been developed for diffusion models [16–18]. However, naïvely applying backpropagation to the diffusion model is inflexible and memory intensive; moreover, such an approach is not trivial to apply to the diffusion models that used an SDE solver instead of an ODE solver.

AdvML-Frontiers'24: The 3nd Workshop on New Frontiers in Adversarial Machine Learning@NeurIPS'24, Vancouver, CA.

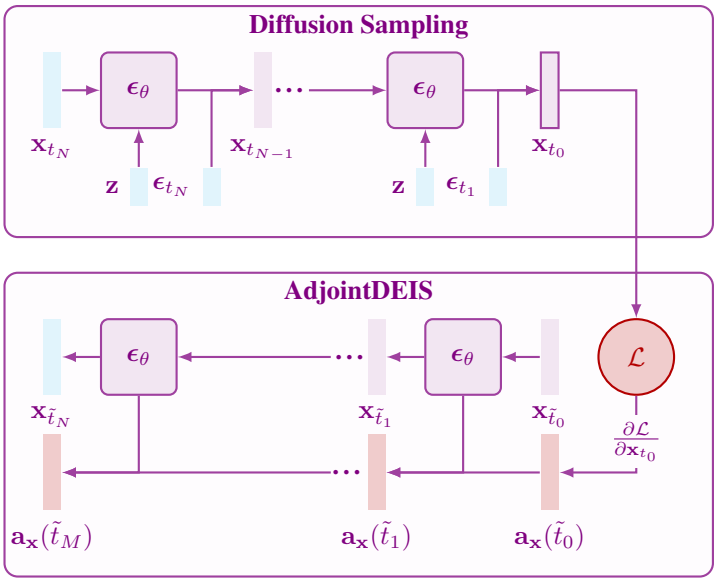


Figure 1: A high-level overview of the AdjointDEIS solver to the continuous adjoint equations for diffusion models. The sampling schedule consists of $\{t_n\}_{n=0}^N$ timesteps for the diffusion model and $\{\tilde{t}_n\}_{n=0}^M$ timesteps for AdjointDEIS. The gradients $\mathbf{a}_\mathbf{x}(T)$ can be used to optimize $\mathbf{x}_T$ to find some optimal $\mathbf{x}_T^*$.

1.1 Contributions

Inspired by the work of Chen et al. [19] we study the application of the continuous adjoint equations to diffusion models, with a focus on training-free guided generation with diffusion models. We introduce several theoretical contributions and technical insights to both improve the ability to perform certain guided generation tasks and to gain insight into guided generation with diffusion models.

First, we introduce *AdjointDEIS* a bespoke family of ODE solvers which can efficiently solve the continuous adjoint equations for both diffusion ODEs and SDEs. To the best of our knowledge, this is the first general backpropagation technique designed for diffusion SDEs. Moreover, we show that the continuous adjoint equations for diffusion SDEs simplify to a mere ODE.

Overall, multiple theoretical contributions and technical insights are provided to bring a new family of techniques for the guided generation of diffusion models, which we evaluate experimentally on the task of face morphing.

2 Diffusion Models

In this section we provide a brief overview of diffusion models. Diffusion models learn a generative process by first perturbing the data distribution into an isotropic Gaussian by progressively adding Gaussian noise to the data distribution, then a neural network is trained to preform denoising steps, allowing for sampling of the data distribution via sampling of a Gaussian distribution [2, 9]. Assume we have an n-dimensional random variable $\mathbf{x} \in \mathbb{R}^n$ with some distribution $p_{\text{data}}(\mathbf{x})$. Then diffusion models begin by diffusing $p_{data}(\mathbf{x})$ according to the diffusion SDE [2], an Itô SDE given as

$$\mathrm{d}\mathbf{x}_t = f(t)\mathbf{x}_t \,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t \tag{2.1}$$

where $t \in [0, T]$ denotes time with fixed constant $T > 0$, $f(\cdot)$ and $g(\cdot)$ denote the drift and diffusion coefficients, and $\mathbf{w}_t$ denotes the standard Wiener process. The trajectories of $\mathbf{x}_t$ follow the distributions $p_t(\mathbf{x}_t)$ with $p_0(\mathbf{x}_0) \equiv p_{data}(\mathbf{x})$ and $p_T(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. Under some regularity conditions Song et al. [15] showed that Equation (2.1) has a reverse process as time runs backwards from T to 0 with initial marginal distribution $p_T(\mathbf{x}_T)$ governed by

$$\mathrm{d}\mathbf{x}_t = [f(t)\mathbf{x}_t - g^2(t)\nabla_\mathbf{x} \log p_t(\mathbf{x}_t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}_t \tag{2.2}$$

where $\bar{\mathbf{w}}_t$ is the standard Wiener process as time runs backwards. Solving Equation (2.2) is what allows diffusion models to draw samples from $p_{data}(\mathbf{x})$ by sampling $p_T(\mathbf{x}_T)$. The unknown term in Equation (2.2) is the *score function* $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$, which in practice is modeled by a neural network that estimates the scaled score function, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \approx -\sigma_t \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$, or some closely related quantity like $\mathbf{x}_0$-prediction [1, 2, 20].

The practical choice of a step size when discretizing SDEs is limited by the randomness of the Wiener process as a large step size, *i.e.*, a small number of steps, can cause non-convergence, particularly in high-dimensional spaces [16]. Sampling an equivalent ODE over an SDE would enable faster sampling. Song et al. [15] showed there exists an ODE whose marginal distribution at time t are identical to that of Equation (2.2). This ODE is known as the *probability flow ODE* [15]. With a noise prediction network, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, trained to model the scaled score function, this ODE can be written as

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = f(t)\mathbf{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \tag{2.3}$$

w.r.t. the noise prediction network.

While there exist several popular choices for the drift and diffusion coefficients, we opt to use the *de facto* choice known as the Variance Preserving (VP) type diffusion SDE [1, 15, 21]. The coefficients for VP-type SDEs are given as

$$f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, \quad g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2 \tag{2.4}$$

which corresponds to sampling $\mathbf{x}_t$ from the distribution $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\alpha_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$.

3 Adjoint Diffusion ODEs

Problem statement. Given the diffusion ODE in Equation (2.3), we wish to solve the following optimization problem:

$$\arg\min_{\mathbf{x}_T, \mathbf{z}, \theta} \mathcal{L}\left(\mathbf{x}_T + \int_T^0 f(t)\mathbf{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)\,\mathrm{d}t\right). \tag{3.1}$$

I.e., we seek to find the optimal $\mathbf{x}_T$, $\mathbf{z}$, and θ which satisfies our guidance function $\mathcal{L}$. *N.B.*, the noise-prediction model is conditioned on additional information $\mathbf{z}$.

Unlike GANs which can update the latent representation through GAN inversion [22, 23], as seen in Equation (3.1) diffusion models require more care as they model an ODE or SDE and require numerical solvers. Therefore, to update the latent representation, model parameters, and conditional information we must backpropagate the gradient of loss defined on the output, $\partial\mathcal{L}(\mathbf{x}_0)/\partial\mathbf{x}_0$ through the whole ODE or SDE.

A key insight of this work is the connection between the adjoint ODE used in neural ODEs by Chen et al. [19] and specialized ODE/SDE solvers by [16–18] for diffusion models. It has been well observed that diffusion models are a type of neural ODE [15, 24]. Since a diffusion model can be thought of as a neural ODE, then we can solve the continuous adjoint equations [25] to find useful gradients for guided generation. We can then exploit the unique structure of diffusion models to develop efficient bespoke ODE solvers for the continuous adjoint equations.

Let $\mathbf{a}_{\mathbf{x}}(t) := \partial\mathcal{L}/\partial\mathbf{x}_t$ and likewise for $\mathbf{a}_{\mathbf{z}}(t)$ and $\mathbf{a}_\theta(t)$. Using exponential integrators we can simplify the formulation of adjoint diffusion ODEs.

Proposition 3.1 (Exact solution of adjoint diffusion ODEs)**.** *Given initial values* $[\mathbf{a}_{\mathbf{x}}(t), \mathbf{a}_{\mathbf{z}}(t), \mathbf{a}_\theta(t)]$ *at time* $t \in (0, T)$*, the solution* $[\mathbf{a}_{\mathbf{x}}(s), \mathbf{a}_{\mathbf{z}}(s), \mathbf{a}_\theta(s)]$ *at time* $s \in (t, T]$ *of adjoint diffusion ODEs is*

$$\mathbf{a}_{\mathbf{x}}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) + \frac{1}{\alpha_s}\int_{\lambda_t}^{\lambda_s} \alpha_\lambda^2 e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial\boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial\mathbf{x}_\lambda}\,\mathrm{d}\lambda, \tag{3.2}$$

$$\mathbf{a}_{\mathbf{z}}(s) = \mathbf{a}_{\mathbf{z}}(t) + \int_{\lambda_t}^{\lambda_s} \alpha_\lambda e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial\boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial\mathbf{z}}\,\mathrm{d}\lambda, \tag{3.3}$$

$$\mathbf{a}_\theta(s) = \mathbf{a}_\theta(t) + \int_{\lambda_t}^{\lambda_s} \alpha_\lambda e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial\boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial\theta}\,\mathrm{d}\lambda. \tag{3.4}$$

The full derivations of Proposition 3.1 can be found in Appendix A.1.

3.1 Numerical Solvers for AdjointDEIS

AdjointDEIS-1. Given an initial augmented adjoint state $[\mathbf{a}_{\mathbf{x}}(t), \mathbf{a}_{\mathbf{z}}(t), \mathbf{a}_{\theta}(t)]$ at time $t \in (0, T)$, the solution $[\mathbf{a}_{\mathbf{x}}(s), \mathbf{a}_{\mathbf{z}}(s), \mathbf{a}_{\theta}(s)]$ at time $s \in (t, T]$ is approximated by

$$\begin{aligned}
\mathbf{a}_{\mathbf{x}}(s) &= \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) + \sigma_s(e^h - 1)\frac{\alpha_t^2}{\alpha_s^2}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t}, \\
\mathbf{a}_{\mathbf{z}}(s) &= \mathbf{a}_{\mathbf{z}}(t) + \sigma_s(e^h - 1)\frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}}, \\
\mathbf{a}_{\theta}(s) &= \mathbf{a}_{\theta}(t) + \sigma_s(e^h - 1)\frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}(\mathbf{x}_t, \mathbf{z}, t)}{\partial \theta}.
\end{aligned} \tag{3.5}$$

We show that AdjointDEIS-k is a k-th order solver, as stated in the following theorem. The proof is in Appendix B.

Theorem 3.1 (AdjointDEIS-k as a k-th order solver)**.** *Assume the function $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)$ and its associated vector-Jacobian products follow the regularity conditions detailed in Appendix B, then for $k = 1, 2$, AdjointDEIS-k is a k-th order solver for adjoint diffusion ODEs, i.e., for the sequence $\{\tilde{\mathbf{a}}_{\mathbf{x}}(t_i)\}_{i=1}^M$ computed by AdjointDEIS-k, the global truncation error at time T satisfies $\tilde{\mathbf{a}}_{\mathbf{x}}(t_M) - \mathbf{a}_{\mathbf{x}}(T) = \mathcal{O}(h_{max}^2)$, where $h_{max} = \max_{1 \le j \le M}(\lambda_{t_i} - \lambda_{t_{i-1}})$. Likewise, AdjointDEIS-$k$ is a k-th order solver for the estimated gradients w.r.t. $\mathbf{z}$ and θ.*

As previous work has shown that higher-order solvers may be unsuitable for large guidance scales [16–18] we do explicitly construct or analyze any solvers for $k > 2$ and leave such explorations for future study.

4 Adjoint Diffusion SDEs

As recent work [26, 27] has shown, diffusion SDEs have useful properties over probability flow ODEs for image manipulation and editing. In particular, it has been shown that probability flow ODEs are invariant in Nie et al. [27, Theorem 3.2] and that diffusion SDEs are contractive in Nie et al. [27, Theorem 3.1], *i.e.*, any gap in the mismatched prior distributions $p_t(\mathbf{x}_t)$ and $\tilde{p}_t(\mathbf{x}_t)$ for the true distribution p_t and edited distribution $\tilde{p}_t$ will remain between $p_0(\mathbf{x}_0)$ and $\tilde{p}_0(\mathbf{x}_0)$, whereas for diffusion SDEs the gap can be reduced between $\tilde{p}_t(\mathbf{x}_t)$ and $p_t(\mathbf{x}_t)$ as t tends towards 0. Motivated by this reasoning, we present a framework for solving the adjoint diffusion SDE using exponential integrators. Full details can be found in Appendix D.

Proposition 4.1 (Exact solution of adjoint diffusion SDEs)**.** *Given initial values $[\mathbf{a}_{\mathbf{x}}(t), \mathbf{a}_{\mathbf{z}}(t), \mathbf{a}_{\theta}(t)]$ at time $t \in (0, T)$, the solution $[\mathbf{a}_{\mathbf{x}}(s), \mathbf{a}_{\mathbf{z}}(s), \mathbf{a}_{\theta}(s)]$ at time $s \in (t, T]$ of adjoint diffusion SDEs is*

$$\mathbf{a}_{\mathbf{x}}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) + \frac{2}{\alpha_s}\int_{\lambda_t}^{\lambda_s} \alpha_\lambda^2 e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\, \mathrm{d}\lambda, \tag{4.1}$$

$$\mathbf{a}_{\mathbf{z}}(s) = \mathbf{a}_{\mathbf{z}}(t) + 2\int_{\lambda_t}^{\lambda_s} \alpha_\lambda e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{z}}\, \mathrm{d}\lambda, \tag{4.2}$$

$$\mathbf{a}_{\theta}(s) = \mathbf{a}_{\theta}(t) + 2\int_{\lambda_t}^{\lambda_s} \alpha_\lambda e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \theta}\, \mathrm{d}\lambda. \tag{4.3}$$

Remark 4.1. *While the adjoint diffusion SDEs evolve with an ODE the same cannot be said for the underlying state, $\mathbf{x}_t$. Rather this evolves with a backwards SDE (more details in Appendix D) which requires the **same** realization of the Wiener process used to sample the image as the one used in the backwards SDE.*

5 Experiments

To illustrate the efficacy of our technique, we examine the application of guided generation for the face morphing attack. The face morphing attack is a new emerging attack on Face Recognition (FR) systems. This attack works by creating a singular morphed face image $\mathbf{x}_0^{(ab)}$ that shares biometric

(a) Identity a (b) Face morphing with AdjointDEIS (c) Identity b

Figure 2: Example of guided morphed face generation with AdjointDEIS on the FRLL dataset.

Figure 3: Comparison of DiM morphs on the FRLL dataset. From left to right, identity a, DiM-A, Fast-DiM, Morph-PIPE, AdjointDEIS (ODE), AdjointDEIS (SDE), and identity b.

information with the two contributing faces $\mathbf{x}_0^{(a)}$ and $\mathbf{x}_0^{(b)}$ [28–30]. A successfully created morphed face image can trigger a false accept with either of the two contributing identities in the targeted Face Recognition (FR) system, see Figure 2 for an illustration. Recent work in this space has explored the use of diffusion models to generate these powerful attacks [28, 31, 32]. All prior work on diffusion-based face morphing used a pre-trained diffusion autoencoder [33] trained on the FFHQ [34] dataset at a 256×256 resolution. We illustrate the use of the AdjointDEIS solvers by modifying the Diffusion Morph (DiM) architecture proposed by Blasingame and Liu [28] to use the AdjointDEIS solvers to find the optimal initial noise $\mathbf{x}_T^{(ab)}$ and conditional $\mathbf{z}_{ab}$. The AdjointDEIS solvers are used to calculate the gradients with respect to the identity loss [32] defined as

$$\begin{aligned} &\mathcal{L}_{ID} = d(v_{ab}, v_a) + d(v_{ab}, v_b), \quad \mathcal{L}_{diff} = \big|d(v_{ab}, v_a) - d(v_{ab}, v_b))\big|, \\ &\mathcal{L}_{ID}^{*} = \mathcal{L}_{ID} + \mathcal{L}_{diff}, \end{aligned} \tag{5.1}$$

where $v_a = F(\mathbf{x}_0^{(a)}), v_b = F(\mathbf{x}_0^{(b)}), v_{ab} = F(\mathbf{x}_0^{(ab)})$, and $F : \mathcal{X} \to V$ is an FR system which embeds images into a vector space V which is equipped with a measure of distance, d. We used the ArcFace [35] FR system for the identity loss.

We compare against three preexisting DiM methods, the original DiM algorithm [28], Fast-DiM [31], and Morph-PIPE [32] as well as a GAN-inversion based face morphing attack, MIPGAN-I and MIPGAN-II [36] based on the StyleGAN [34] and StyleGAN2 [37] architectures respectively. Fast-DiM improves DiM by using higher-order ODE solvers to decrease the number of sampling steps required to create a morph. Morph-PIPE performs a very simple version of guided generation by generating a large batch of morphed images derived from a discrete set of interpolations between $\mathbf{x}_T^{(a)}$ and $\mathbf{x}_T^{(b)}$, and $\mathbf{z}_a$ and $\mathbf{z}_b$. For reference purposes, we compare against a reference GAN-based method [36] which uses GAN-inversion w.r.t.to the identity loss to find the optimal morphed face, and we include prior state-of-the-art Webmorph, a commercial off-the-shelf system [38].

We run our experiments on the SYN-MAD 2022 [38] morphed pairs which are constructed from the Face Research Lab London dataset [39], more details in Appendix G.4. The morphed images are evaluated against three FR systems, the ArcFace [35], ElasticFace [40], and AdaFace [41] models,

further details are found in Appendix G.5. To measure the efficacy of a morphing attack, the Mated Morph Presentation Match Rate (MMPMR) metric [42] is used. The MMPMR metric as proposed by Scherhag et al. [42] is defined as

$$M(\delta) = \frac{1}{M} \sum_{m=1}^{M} \left\{ \left[\min_{n \in \{1,\dots,N_m\}} S_m^n \right] > \delta \right\} \tag{5.2}$$

where δ is the verification threshold, S_m^n is the similarity score of the n-th subject of morph m, N_m is the total number of contributing subjects to morph m, and M is the total number of morphed images.

In our experiments, we used a learning rate of 0.01, $N = 20$ sampling steps, $M = 20$ steps for AdjointDEIS, and 50 optimization steps for gradient descent.

Table 1: Vulnerability of different FR systems across different morphing attacks on the SYN-MAD 2022 dataset. FMR = 0.1%.

		MMPMR(↑)		
Morphing Attack	**NFE(↓)**	**AdaFace**	**ArcFace**	**ElasticFace**
Webmorph [38]	-	97.96	96.93	98.36
MIPGAN-I [36]	-	72.19	77.51	66.46
MIPGAN-II [36]	-	70.55	72.19	65.24
DiM-A [28]	350	92.23	90.18	93.05
Fast-DiM [31]	300	92.02	90.18	93.05
Morph-PIPE [32]	2350	95.91	92.84	95.5
DiM + AdjointDEIS-1 (ODE)	2250	**99.8**	**98.77**	**99.39**
DiM + AdjointDEIS-1 (SDE)	2250	98.57	97.96	97.75

In Table 1 we present the effectiveness of the morphing attacks against the three FR systems. Guided generation with AdjointDEIS massively increases the performance of DiM, supplanting the old state-of-the-art for face morphing. Interestingly, the SDE variant did not fare as well as the ODE variant. This is likely due to the difficulty in discretizing SDEs with large step sizes [15–17]. We present further results in Appendix E which explore the impact of the choice of learning rate and number of discretization steps for AdjointDEIS.

6 Conclusion

We present a unified view on guided generation by updating latent, conditional, and model information of diffusion models with a guidance function using the continuous adjoint equations. We propose AdjointDEIS, a family of solvers for the continuous adjoint equations of diffusion models. We exploit the unique construction of diffusion models to create efficient numerical solvers by using exponential integrators. We prove the convergence order of solvers and show that the continuous adjoint equations for diffusion SDEs evolve with an ODE. Furthermore, we show how to handle conditional information that is scheduled in time, further expanding the generalizability of the proposed technique. Our results in face morphing show that the gradients produced by AdjointDEIS can be used for guided generation tasks.

Limitations. There are several limitations. Empirically, we only explored a small subset of the true potential AdjointDEIS by evaluating on a single scenario, *i.e.*, face morphing. Likewise, we only explored a few different hyperparameter options. In particular, we did not explore much the impact of the number of optimization steps and the number of sampling steps for diffusion SDEs on the visual quality of the generated face morphs.

Broader Impact. Guided generation techniques can be misused for a variety of harmful purposes. In particular, our approach provides a powerful tool for adversarial attacks. However, better knowledge of such techniques should hopefully help direct research in hardening systems against such kinds of attacks.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf. 1, 3, 27

[2] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP. 1, 2, 3

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. 1, 17

[4] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv e-prints*, art. arXiv:2204.06125, April 2022. doi: 10.48550/arXiv.2204.06125.

[5] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ec795aeadae0b7d230fa35cbaf04c041-Paper-Conference.pdf. 1

[6] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. AudioLDM: Text-to-Audio Generation with Latent Diffusion Models. *arXiv e-prints*, art. arXiv:2301.12503, January 2023. doi: 10.48550/arXiv.2301.12503. 1

[7] Scott H. Hawley. Pictures of midi: Controlled music generation via graphical prompts for image-based diffusion inpainting, 2024. URL https://arxiv.org/abs/2407.01499. 1

[8] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. *arXiv e-prints*, art. arXiv:2304.08818, April 2023. doi: 10.48550/arXiv.2304.08818. 1

[9] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022. 1, 2

[10] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242*, 2022. 1

[11] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. *arXiv e-prints*, art. arXiv:2208.01618, August 2022. doi: 10.48550/arXiv.2208.01618.

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI. 1

[14] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal Guidance for Diffusion Models. *arXiv e-prints*, art. arXiv:2302.07121, February 2023. doi: 10.48550/arXiv.2302.07121. 1

[15] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS. 1, 2, 3, 6, 27

[16] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan LI, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5775–5787. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/260a14acce2a89dad36adc8eefe7c59e-Paper-Conference.pdf. 1, 3, 4, 13, 14, 15

[17] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023. 6, 15, 21

[18] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023. 1, 3, 4

[19] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf. 2, 3

[20] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI. 3

[21] Yang Song and Stefano Ermon. *Generative modeling by estimating gradients of the data distribution*. Curran Associates Inc., Red Hook, NY, USA, 2019. 3

[22] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *IEEE/CVF Int'l Conf. on Comp. Vision (ICCV)*, pages 4431–4440, 2019. doi: 10.1109/ICCV.2019.00453. 3

[23] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8296–8305, 2020. 3

[24] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t. 3

[25] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, Oxford University, 2022. 3

[26] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. 4

[27] Shen Nie, Hanzhong Allan Guo, Cheng Lu, Yuhao Zhou, Chenyu Zheng, and Chongxuan Li. The blessing of randomness: SDE beats ODE in general diffusion-based image editing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DesYwmUG00. 4, 21

[28] Zander W. Blasingame and Chen Liu. Leveraging diffusion for strong and high quality face morphing attacks. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 6(1): 118–131, 2024. doi: 10.1109/TBIOM.2024.3349857. 5, 6, 24, 25

[29] R. Raghavendra, K. B. Raja, and C. Busch. Detecting morphed face images. In *IEEE 8th Int'l Conf. on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–7, 2016. doi: 10.1109/BTAS.2016.7791169.

[30] Eklavya Sarkar, Pavel Korshunov, Laurent Colbois, and Sébastien Marcel. Are gan-based morphs threatening face recognition? In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2959–2963, 2022. doi: 10.1109/ICASSP43922.2022.9746477. 5, 25

[31] Zander W. Blasingame and Chen Liu. Fast-dim: Towards fast diffusion morphs. *IEEE Security & Privacy*, 22(4):103–114, June 2024. doi: 10.1109/MSEC.2024.3410112. 5, 6

[32] Haoyu Zhang, Raghavendra Ramachandra, Kiran Raja, and Busch Christoph. Morph-pipe: Plugging in identity prior to enhance face morphing attack based on diffusion model. In *Norwegian Information Security Conference (NISK)*, 2023. 5, 6, 25

[33] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10619–10629, June 2022. 5

[34] 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)*, pages 4396–4405, 2019. doi: 10.1109/CVPR.2019.00453. 5

[35] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 5, 25

[36] Haoyu Zhang, Sushma Venkatesh, Raghavendra Ramachandra, Kiran Raja, Naser Damer, and Christoph Busch. Mipgan—generating strong and high quality morphing attacks using identity prior driven gan. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 3(3): 365–383, 2021. doi: 10.1109/TBIOM.2021.3072349. 5, 6, 25

[37] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020. doi: 10.1109/CVPR42600.2020.00813. 5

[38] Marco Huber, Fadi Boutros, Anh Thi Luu, Kiran Raja, Raghavendra Ramachandra, Naser Damer, Pedro C. Neto, Tiago Gonçalves, Ana F. Sequeira, Jaime S. Cardoso, João Tremoço, Miguel Lourenço, Sergio Serra, Eduardo Cermeño, Marija Ivanovska, Borut Batagelj, Andrej Kronovšek, Peter Peer, and Vitomir Štruc. Syn-mad 2022: Competition on face morphing attack detection based on privacy-aware synthetic training data. In *2022 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–10, 2022. doi: 10.1109/IJCB54206.2022.10007950. 5, 6, 25

[39] Lisa DeBruine and Benedict Jones. Face Research Lab London Set. 5 2017. doi: 10.6084/m9.figshare.5047666.v5. URL https://figshare.com/articles/dataset/Face_Research_Lab_London_Set/5047666. 5, 25

[40] Fadi Boutros, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. Elasticface: Elastic margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1578–1587, June 2022. 5, 25

[41] Minchul Kim, Anil K Jain, and Xiaoming Liu. Adaface: Quality adaptive margin for face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 5, 25

[42] Ulrich Scherhag, Andreas Nautsch, Christian Rathgeb, Marta Gomez-Barrero, Raymond N. J. Veldhuis, Luuk Spreeuwers, Maikel Schils, Davide Maltoni, Patrick Grother, Sebastien Marcel, Ralph Breithaupt, Raghavendra Ramachandra, and Christoph Busch. Biometric systems under morphing attacks: Assessment of morphing techniques and vulnerability reporting. In *2017 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–7, 2017. doi: 10.23919/BIOSIG.2017.8053499. 6

[43] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19: 209–286, 2010. doi: 10.1017/S0962492910000048. 13

[44] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016. 18

[45] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3870–3882. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/li20i.html. 19, 20

[46] Hiroshi Kunita. Stochastic differential equations and stochastic flows. *Stochastic Flows and Jump-Diffusions*, pages 77–124, 2019. 19

[47] Chen Henry Wu and Fernando De la Torre. A latent space of stochastic diffusion models for zero-shot image editing and guidance. In *ICCV*, 2023. 21

[48] Zander W. Blasingame and Chen Liu. Greedy-dim: Greedy algorithms for unreasonably effective face morphs. In *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–10, September 2024. 24, 25, 26

[49] Ionut Cosmin Duta, Li Liu, Fan Zhu, and Ling Shao. Improved residual networks for image and video recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9415–9422, 2021. doi: 10.1109/ICPR48806.2021.9412193. 25

[50] Xiang An, Xuhan Zhu, Yuan Gao, Yang Xiao, Yongle Zhao, Ziyong Feng, Lan Wu, Bin Qin, Ming Zhang, Debing Zhang, and Ying Fu. Partial fc: Training 10 million identities on a single machine. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1445–1449, 2021. doi: 10.1109/ICCVW54120.2021.00166. 25

[51] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 25, 26

[52] Xingchao Liu, Lemeng Wu, Shujian Zhang, Chengyue Gong, Wei Ping, and Qiang Liu. Flowgrad: Controlling the output of generative odes with gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24335–24344, 2023. 25, 26

[53] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance, 2023. 25, 26

[54] Jiachun Pan, Jun Hao Liew, Vincent Tan, Jiashi Feng, and Hanshu 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. 25, 26, 27

[55] Pierre Marion, Anna Korba, Peter Bartlett, Mathieu Blondel, Valentin De Bortoli, Arnaud Doucet, Felipe Llinares-López, Courtney Paquette, and Quentin Berthet. Implicit diffusion: Efficient optimization through stochastic sampling. *arXiv preprint arXiv:2402.05468*, 2024. 26, 27

[56] Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22532–22541, 2023. 26

A Derivation of AdjointDEIS

In this section we provide the full derivations for the family of AdjointDEIS solvers. First recall the full definition of the continuous adjoint equations for the empirical probability flow ODE:

$$\begin{aligned}
\mathbf{a}_{\mathbf{x}}(0) &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_0}, & \frac{\mathrm{d}\mathbf{a}_{\mathbf{x}}}{\mathrm{d}t}(t) &= -\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t}, \\
\mathbf{a}_{\mathbf{z}}(0) &= \mathbf{0}, & \frac{\mathrm{d}\mathbf{a}_{\mathbf{z}}}{\mathrm{d}t}(t) &= -\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}}, \\
\mathbf{a}_{\theta}(0) &= \mathbf{0}, & \frac{\mathrm{d}\mathbf{a}_{\theta}}{\mathrm{d}t}(t) &= -\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \theta}.
\end{aligned} \tag{A.1}$$

We can simplify the equations by explicitly solving gradients of the neural vector field $\boldsymbol{f}_\theta$ for the drift term to obtain

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{a}_{\mathbf{x}}}{\mathrm{d}t}(t) &= -f(t)\mathbf{a}_{\mathbf{x}}(t) - \frac{g^2(t)}{2\sigma_t}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t}, \\
\frac{\mathrm{d}\mathbf{a}_{\mathbf{z}}}{\mathrm{d}t}(t) &= \mathbf{0} - \frac{g^2(t)}{2\sigma_t}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}}, \\
\frac{\mathrm{d}\mathbf{a}_{\theta}}{\mathrm{d}t}(t) &= \mathbf{0} - \frac{g^2(t)}{2\sigma_t}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \theta}.
\end{aligned} \tag{A.2}$$

Remark A.1. *The last two equations in Equations (A.2) the vector fields are independent of $(\mathbf{a}_{\mathbf{z}}, \mathbf{a}_\theta)$, reducing these equations to mere integrals; however, it is often useful to compute the whole system $\mathbf{a}_{aug} = (\mathbf{a}_{\mathbf{x}}, \mathbf{a}_{\mathbf{z}}, \mathbf{a}_\theta)$ as an augmented ODE.*

Remark A.2. *Likewise, the last two equations in Equations (A.2) are functionally identical with a simple swap of $\mathbf{z}$ for θ or vice versa.*

As such, for the sake of brevity, the derivations for the AdjointDEIS solvers for $(\mathbf{a}_{\mathbf{z}}, \mathbf{a}_\theta)$ will only explicitly include the derivations for $\mathbf{a}_{\mathbf{z}}$.

A.1 Simplified Formulation of the Continuous Adjoint Equations

Focusing first on the continuous adjoint equation for $\mathbf{a}_{\mathbf{x}}$ we apply the integrating factor $\exp\left(\int_0^t f(\tau)\,\mathrm{d}\tau\right)$ to Equation (A.2) to find

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[e^{\int_0^t f(\tau)\,\mathrm{d}\tau}\mathbf{a}_{\mathbf{x}}(t)\right] = -e^{\int_0^t f(\tau)\,\mathrm{d}\tau}\frac{g^2(t)}{2\sigma_t}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t}. \tag{A.3}$$

Then, the exact solution at time s given time $t < s$ is found to be

$$\begin{aligned}
e^{\int_0^s f(\tau)\,\mathrm{d}\tau}\mathbf{a}_{\mathbf{x}}(s) &= e^{\int_0^t f(\tau)\,\mathrm{d}\tau}\mathbf{a}_{\mathbf{x}}(t) - \int_t^s e^{\int_0^u f(\tau)\,\mathrm{d}\tau}\frac{g^2(u)}{2\sigma_u}\mathbf{a}_{\mathbf{x}}(u)^\top \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_u, \mathbf{z}, u)}{\partial \mathbf{x}_u}\,\mathrm{d}u \\
\mathbf{a}_{\mathbf{x}}(s) &= e^{\int_s^t f(\tau)\,\mathrm{d}\tau}\mathbf{a}_{\mathbf{x}}(t) - \int_t^s e^{\int_s^u f(\tau)\,\mathrm{d}\tau}\frac{g^2(u)}{2\sigma_u}\mathbf{a}_{\mathbf{x}}(u)^\top \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_u, \mathbf{z}, u)}{\partial \mathbf{x}_u}\,\mathrm{d}u
\end{aligned} \tag{A.4}$$

To simplify Equation (A.4), recall that $f(t)$ is defined as

$$f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, \tag{A.5}$$

for VP type SDEs. Furthermore, let $\lambda_t \coloneqq \log(\alpha_t/\sigma_t)$ be one half of the log-SNR. Then the diffusion coefficient can be simplified using the log-derivative trick such that

$$g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2 = 2\sigma_t^2\left(\frac{\mathrm{d}\log\sigma_t}{\mathrm{d}t} - \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\right) = -2\sigma_t^2\frac{\mathrm{d}\lambda_t}{\mathrm{d}t}. \tag{A.6}$$

Using this updated expression of $g^2(t)$ along with computing the integrating factor in closed form enables us to express Equation (A.4) as

$$\mathbf{a}_{\mathbf{x}}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) + \frac{1}{\alpha_s}\int_t^s \alpha_u\sigma_u\frac{\mathrm{d}\lambda_u}{\mathrm{d}u}\mathbf{a}_{\mathbf{x}}(u)^\top \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_u, \mathbf{z}, u)}{\partial \mathbf{x}_u}\,\mathrm{d}u. \tag{A.7}$$

Lastly, by rewriting the integral in terms of an exponentially weighted integral $\alpha_u\sigma_u = \alpha_u^2\sigma_u/\alpha_u = \alpha_u^2 e^{-\lambda_u}$ we find

$$\mathbf{a}_{\mathbf{x}}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) + \frac{1}{\alpha_s}\int_{\lambda_t}^{\lambda_s} \alpha_\lambda^2 e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\,\mathrm{d}\lambda. \tag{A.8}$$

This change of variables is possible as λ_t is a strictly decreasing function w.r.t. t and therefore it has an inverse function t_λ which satisfies $t_\lambda(\lambda_t) = t$, and, with abuse of notation, we let $\mathbf{x}_\lambda \coloneqq \mathbf{x}_{t_\lambda(\lambda)}$, $\mathbf{a}_{\mathbf{x}}(\lambda) \coloneqq \mathbf{a}_{\mathbf{x}}(t_\lambda(\lambda))$, *&c.*and let the reader infer from context if the function is mapping the log-SNR back into the time domain or already in the time domain.

Now we will show the derivations to find a simplified form of the continuous adjoint equation for the conditional information. Using the continuous adjoint equation from Equations (A.2) for $\mathbf{a}_{\mathbf{z}}(t)$ along with the log-SNR, we can express the evolution of $\mathbf{a}_{\mathbf{z}}(t)$ as

$$\frac{\mathrm{d}\mathbf{a}_{\mathbf{z}}}{\mathrm{d}t}(t) = \sigma_t \frac{\mathrm{d}\lambda_t}{\mathrm{d}t}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}}. \tag{A.9}$$

As we would like to express this as an exponential integrator, we simply multiply σ_t by α_t/α_t to obtain $\alpha_t \cdot \sigma_t/\alpha_t = \alpha_t e^{-\lambda_t}$, as such we can rewrite Equation (A.9) as

$$\frac{\mathrm{d}\mathbf{a}_{\mathbf{z}}}{\mathrm{d}t}(t) = \alpha_t e^{-\lambda_t} \frac{\mathrm{d}\lambda_t}{\mathrm{d}t}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}}. \tag{A.10}$$

Using Equations (A.8) and (A.10), we arrive at Proposition 3.1 from the main paper.

Proposition A.1. *Given initial values* $[\mathbf{a}_{\mathbf{x}}(t), \mathbf{a}_{\mathbf{z}}(t), \mathbf{a}_\theta(t)]$ *at time* $t \in (0, T)$*, the solution* $[\mathbf{a}_{\mathbf{x}}(s), \mathbf{a}_{\mathbf{z}}(s), \mathbf{a}_\theta(s)]$ *at time* $s \in (t, T]$ *of the adjoint empirical probability flow ODE in Equation* (A.4) *is*

$$\mathbf{a}_{\mathbf{x}}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) + \frac{1}{\alpha_s}\int_{\lambda_t}^{\lambda_s} \alpha_\lambda^2 e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\,\mathrm{d}\lambda, \tag{A.11}$$

$$\mathbf{a}_{\mathbf{z}}(s) = \mathbf{a}_{\mathbf{z}}(t) + \int_{\lambda_t}^{\lambda_s} \alpha_\lambda e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{z}}\,\mathrm{d}\lambda, \tag{A.12}$$

$$\mathbf{a}_\theta(s) = \mathbf{a}_\theta(t) + \int_{\lambda_t}^{\lambda_s} \alpha_\lambda e^{-\lambda}\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \theta}\,\mathrm{d}\lambda. \tag{A.13}$$

Then to find the AdjointDEIS solvers we take a k-th order Taylor expansion about λ_t and integrate in the log-SNR domain.

A.2 Taylor Expansion

For $k \geq 1$, the $(k-1)$-th Taylor expansion at λ_t of the inner term of the exponentially weighted integral in Equation (A.11) is

$$\mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda} = \sum_{n=0}^{k-1} \frac{(\lambda - \lambda_t)^n}{n!}\frac{\mathrm{d}^n}{\mathrm{d}\lambda^n}\left[\alpha_\lambda^2 \mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\right]_{\lambda=\lambda_t} + \mathcal{O}((\lambda - \lambda_t)^k). \tag{A.14}$$

Then plugging this into Equation (A.11) yields

$$\begin{aligned}
\mathbf{a}_{\mathbf{x}}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) \quad & + \frac{1}{\alpha_s}\int_{\lambda_t}^{\lambda_s} e^{-\lambda} \sum_{n=0}^{k-1} \frac{(\lambda - \lambda_t)^n}{n!}\frac{\mathrm{d}^n}{\mathrm{d}\lambda^n}\left[\alpha_\lambda^2 \mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\right]_{\lambda=\lambda_t} \mathrm{d}\lambda \\
& + \mathcal{O}(h^{k+1}) \\
= \frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t) \quad & + \frac{1}{\alpha_s}\sum_{n=0}^{k-1} \underbrace{\frac{\mathrm{d}^n}{\mathrm{d}\lambda^n}\left[\alpha_\lambda^2 \mathbf{a}_{\mathbf{x}}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\right]_{\lambda=\lambda_t}}_{\text{estimated}} \underbrace{\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^n}{n!} e^{-\lambda}\,\mathrm{d}\lambda}_{\text{analytically computed}} \\
& + \underbrace{\mathcal{O}(h^{k+1})}_{\text{omitted}},
\end{aligned} \tag{A.15}$$

where $h = \lambda_s - \lambda_t$.

The exponentially weighted integral $\int_{\lambda_t}^{\lambda_s} \frac{(\lambda-\lambda_t)^n}{n!} e^{-\lambda} \, \mathrm{d}\lambda$ can be solved *analytically* by applying n times integration by parts [16, 43] such that

$$\int_{\lambda_t}^{\lambda_s} e^{-\lambda} \frac{(\lambda - \lambda_t)^n}{n!} \, \mathrm{d}\lambda = \frac{\sigma_s}{\alpha_s} h^{n+1} \varphi_{n+1}(h), \tag{A.16}$$

with the special φ-functions [43]. These functions are defined as

$$\varphi_{n+1}(h) \coloneqq \int_0^1 e^{(1-u)h} \frac{u^n}{n!} \, \mathrm{d}u, \qquad \varphi_0(h) = e^h, \tag{A.17}$$

which satisfy the recurrence relation $\varphi_{k+1}(h) = (\varphi_k(h) - \varphi_k(0))/h$ and have closed forms for $k = 1, 2$:

$$\varphi_1(h) = \frac{e^h - 1}{h}, \tag{A.18}$$

$$\varphi_2(h) = \frac{e^h - h - 1}{h^2}. \tag{A.19}$$

Likewise, the Taylor expansion of the exponentially weighted integral in Equation (A.12) yields

$$\begin{aligned} \mathbf{a}_\mathbf{z}(s) &= \mathbf{a}_\mathbf{z}(t) + \int_{\lambda_t}^{\lambda_s} e^{-\lambda} \sum_{n=0}^{k-1} \frac{(\lambda - \lambda_t)^n}{n!} \frac{\mathrm{d}^n}{\mathrm{d}\lambda^n} \left[\alpha_\lambda \mathbf{a}_\mathbf{x}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{z}} \right]_{\lambda=\lambda_t} \mathrm{d}\lambda + \mathcal{O}(h^{k+1}) \\ &= \mathbf{a}_\mathbf{z}(t) + \sum_{n=0}^{k-1} \underbrace{\frac{\mathrm{d}^n}{\mathrm{d}\lambda^n} \left[\alpha_\lambda \mathbf{a}_\mathbf{x}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{z}} \right]_{\lambda=\lambda_t}}_{\text{estimated}} \underbrace{\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^n}{n!} e^{-\lambda} \, \mathrm{d}\lambda}_{\text{analytically computed}} + \underbrace{\mathcal{O}(h^{k+1})}_{\text{omitted}}. \end{aligned} \tag{A.20}$$

A.3 AdjointDEIS-1

For $k = 1$ and omitting the higher-order error term, Equation (A.15) becomes:

$$\begin{aligned} \mathbf{a}_\mathbf{x}(s) &= \frac{\alpha_t}{\alpha_s} \mathbf{a}_\mathbf{x}(t) + \frac{1}{\alpha_s} \alpha_t^2 \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t} \int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^0}{0!} e^{-\lambda} \, \mathrm{d}\lambda \\ &= \frac{\alpha_t}{\alpha_s} \mathbf{a}_\mathbf{x}(t) + \sigma_s (e^h - 1) \frac{\alpha_t^2}{\alpha_s^2} \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t} \qquad \text{By Equation (A.16).} \end{aligned} \tag{A.21}$$

Likewise, the continuous adjoint equation for $\mathbf{z}$, Equation (A.20), becomes when $k = 1$ by omitting the higher-order error term:

$$\begin{aligned} \mathbf{a}_\mathbf{z}(s) &= \mathbf{a}_\mathbf{z}(t) + \alpha_t \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}} \int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^0}{0!} e^{-\lambda} \, \mathrm{d}\lambda \\ &= \mathbf{a}_\mathbf{z}(t) + \sigma_s (e^h - 1) \frac{\alpha_t}{\alpha_s} \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}} \qquad \text{By Equation (A.16).} \end{aligned} \tag{A.22}$$

And the first-order solver for $\mathbf{a}_\theta(t)$ can be found in a similar fashion, thus we have derived the AdjointDEIS-1 solvers.

A.4 AdjointDEIS-2M

For notational convenience let $\mathbf{V}(\mathbf{x}; t)$ denote the scaled vector-Jacobian product of the adjoint state $\mathbf{a}_\mathbf{x}(t)$ and the gradient of the model w.r.t. $\mathbf{x}_t$, *i.e.*,

$$\mathbf{V}(\mathbf{x}; t) = \alpha_t^2 \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t}. \tag{A.23}$$

Consider the following definition of the limit in the log-SNR domain

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\left[\alpha_\lambda^2 \mathbf{a}_\mathbf{x}(\lambda)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_\lambda, \mathbf{z}, \lambda)}{\partial \mathbf{x}_\lambda}\right] = \lim_{\lambda_r \to \lambda_t} \frac{\mathbf{V}(\mathbf{x}; \lambda_t) - \mathbf{V}(\mathbf{x}; \lambda_r)}{\rho h}, \tag{A.24}$$

where $\rho = \frac{\lambda_t - \lambda_r}{h}$ with $h = \lambda_s - \lambda_t$ and where r is some previous step $r < t < s$. Again $\mathbf{V}(\mathbf{x}; \lambda_t)$ is overloaded to mean $\mathbf{V}(\mathbf{x}; t_\lambda(\lambda_t))$. Then by omitting higher-order error $\mathcal{O}(h^{k+1})$, Equation (A.15) becomes:

$$\begin{aligned}
\mathbf{a}_\mathbf{x}(s) &= \frac{\alpha_t}{\alpha_s}\mathbf{a}_\mathbf{x}(t) + \frac{1}{\alpha_s}\left[\mathbf{V}(\mathbf{x}; \lambda_t)\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^0}{0!}\,\mathrm{d}\lambda + \mathbf{V}^{(1)}(\mathbf{x}; \lambda_t)\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^1}{1!}\,\mathrm{d}\lambda\right] \\
&= \frac{\alpha_t}{\alpha_s}\mathbf{a}_\mathbf{x}(t) + \frac{1}{\alpha_s}\left[\frac{\sigma_s}{\alpha_s}(e^h - 1)\mathbf{V}(\mathbf{x}; \lambda_t) + \frac{\sigma_s}{\alpha_s}(e^h - h - 1)\mathbf{V}^{(1)}(\mathbf{x}; \lambda_t)\right].
\end{aligned} \tag{A.25}$$

By applying the same approximation used in Lu et al. [16] of

$$\frac{e^h - h - 1}{h} \approx \frac{e^h - 1}{2}, \tag{A.26}$$

then we can rewrite the second term of the Taylor expansion as

$$\begin{aligned}
\frac{\sigma_s}{\alpha_s}(e^h - h - 1)\mathbf{V}^{(1)}(\mathbf{x}; \lambda_t) &\approx \frac{\sigma_s}{\alpha_s}(e^h - h - 1)\frac{\mathbf{V}(\mathbf{x}; \lambda_t) - \mathbf{V}(\mathbf{x}; \lambda_r)}{\rho h} && \text{By Equation (A.24)} \\
&\approx \frac{\sigma_s}{\alpha_s}\frac{e^h - 1}{2\rho}\big(\mathbf{V}(\mathbf{x}; \lambda_t) - \mathbf{V}(\mathbf{x}; \lambda_r)\big) && \text{By Equation (A.26)} \\
&= \frac{\sigma_s}{\alpha_s}\frac{e^h - 1}{2\rho}\left(\alpha_t^2 \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t} - \alpha_r^2 \mathbf{a}_\mathbf{x}(r)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_r, \mathbf{z}, r)}{\partial \mathbf{x}_r}\right).
\end{aligned} \tag{A.27}$$

Then Equation (A.25) becomes

$$\begin{aligned}
\mathbf{a}_\mathbf{x}(s) = \frac{\alpha_t}{\alpha_s}\mathbf{a}_\mathbf{x}(t) &+ \sigma_s(e^h - 1)\frac{\alpha_t^2}{\alpha_s^2}\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t} \\
&+ \sigma_s \frac{e^h - 1}{2\rho}\left(\frac{\alpha_t^2}{\alpha_s^2}\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t} - \frac{\alpha_r^2}{\alpha_s^2}\mathbf{a}_\mathbf{x}(r)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_r, \mathbf{z}, r)}{\partial \mathbf{x}_r}\right).
\end{aligned} \tag{A.28}$$

Likewise, consider the scaled vector-Jacobian product of the adjoint state $\mathbf{a}_\mathbf{x}(t)$ and the gradient of the model w.r.t. $\mathbf{z}$, *i.e.*,

$$\mathbf{V}(\mathbf{z}; t) = \alpha_t \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}}, \tag{A.29}$$

along with a corresponding definition of first-derivative w.r.t. λ as defined in Equation (A.24). As such Equation (A.20), when $k = 2$, becomes the following when omitting the higher-order error term:

$$\begin{aligned}
\mathbf{a}_\mathbf{z}(s) &= \mathbf{a}_\mathbf{z}(t) + \mathbf{V}(\mathbf{z}; \lambda_t)\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^0}{0!}\,\mathrm{d}\lambda + \mathbf{V}^{(1)}(\mathbf{z}; \lambda_t)\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^1}{1!}\,\mathrm{d}\lambda \\
&= \mathbf{a}_\mathbf{z}(t) + \frac{\sigma_s}{\alpha_s}(e^h - 1)\mathbf{V}(\mathbf{z}; \lambda_t) + \frac{\sigma_s}{\alpha_s}(e^h - h - 1)\mathbf{V}^{(1)}(\mathbf{z}; \lambda_t).
\end{aligned} \tag{A.30}$$

The second term of the Taylor expansion can be rewritten as

$$\begin{aligned}
\frac{\sigma_s}{\alpha_s}(e^h - h - 1)\mathbf{V}^{(1)}(\mathbf{z}; \lambda_t) &\approx \frac{\sigma_s}{\alpha_s}(e^h - h - 1)\frac{\mathbf{V}(\mathbf{z}; \lambda_t) - \mathbf{V}(\mathbf{z}; \lambda_r)}{\rho h} \\
&\approx \frac{\sigma_s}{\alpha_s}\frac{e^h - 1}{2\rho}\big(\mathbf{V}(\mathbf{z}; \lambda_t) - \mathbf{V}(\mathbf{z}; \lambda_r)\big) && \text{By Equation (A.26)} \\
&= \frac{\sigma_s}{\alpha_s}\frac{e^h - 1}{2\rho}\left(\alpha_t \mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}} - \alpha_r \mathbf{a}_\mathbf{x}(r)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_r, \mathbf{z}, r)}{\partial \mathbf{z}}\right).
\end{aligned} \tag{A.31}$$

Then Equation (A.30) becomes

$$\mathbf{a}_{\mathbf{z}}(s) = \mathbf{a}_{\mathbf{z}}(t) + \sigma_s(e^h - 1)\frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}} + \sigma_s \frac{e^h - 1}{2\rho}\left(\frac{\alpha_t}{\alpha_s}\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{z}} - \frac{\alpha_r}{\alpha_s}\mathbf{a}_{\mathbf{x}}(r)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_r, \mathbf{z}, r)}{\partial \mathbf{z}}\right), \tag{A.32}$$

and the corresponding second-order solver for $\mathbf{a}_\theta(t)$ can be found in a similar manner.

B Proof of Theorem 3.1

For notational brevity we denote the *scaled* vector-Jacobian products of the solution trajectory of AdjointDEIS as

$$\tilde{\mathbf{V}}(\mathbf{x}; t) = \alpha_t^2 \tilde{\mathbf{a}}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_t, \mathbf{z}, t)}{\partial \tilde{\mathbf{x}}_t}, \tag{B.1}$$

$$\tilde{\mathbf{V}}(\mathbf{z}; t) = \alpha_t \tilde{\mathbf{a}}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_t, \mathbf{z}, t)}{\partial \mathbf{z}}. \tag{B.2}$$

B.1 Assumptions

For the AdjointDEIS solvers, we make similar assumptions to Lu et al. [16].

Assumption B.1. *The total derivatives of the vector-Jacobian products $\mathbf{V}^{(k)}(\{\mathbf{x}_\lambda, \mathbf{z}, \theta\}, \lambda)$ as a function of λ exist and are continuous for $0 \leq j \leq k+1$ (and hence bounded).*

Assumption B.2. *The function $\boldsymbol{\epsilon}_\theta(\mathbf{x}, \mathbf{z}, t)$ is continuous in t and uniformly Lipschitz and continuously differentiable w.r.t. its first parameter $\mathbf{x}$.*

Assumption B.3. $h_{max} \coloneqq \max_{1 \leq j \leq M} h_j = \mathcal{O}(1/M)$.

Assumption B.4. $\rho_i > c > 0$ *for all* $i = 1, \ldots, M$ *and some constant* c.

The first assumption is required by Taylor's theorem. The second assumption is a mild assumption to ensure Theorem B.1 holds, which is used to replace $\tilde{\mathbf{V}}(\{\mathbf{x}_t, \mathbf{z}, \theta\}, t)$ with $\mathbf{V}(\{\mathbf{x}_t, \mathbf{z}, \theta\}, t) + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t) - \mathbf{a}_{\mathbf{x}}(t))$ so the Taylor expansion w.r.t. λ_s is applicable. The third assumption is a technical assumption to exclude a significantly large step size. The last assumption is necessary for the case when $k = 2$. For our proofs we follow a similar outline to that taken by Lu et al. [17, Appendix A].

B.2 The Vector-Jacobian Product is Lipschitz

Lemma B.1 (Vector-Jacobian Product is Lipschitz.)**.** *Let $\boldsymbol{f}_\theta : \mathbb{R}^d \times \mathbb{R}^z \times [0, T] \to \mathbb{R}^d$ be continuous in t and uniformly Lipschitz and continuously differentiable in $\mathbf{x}$. Let $\mathbf{x} : [0, T] \to \mathbb{R}^d$ be the unique solution to*

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)$$

with initial condition $\mathbf{x}_0$. Then the following map

$$(\mathbf{a}, t) \mapsto -\mathbf{a}^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial [\mathbf{x}_t, \mathbf{z}, \theta]}$$

is Lipschitz in $\mathbf{a}$. Moreover, the Lipschitz constant $L > 0$ is given by

$$L = \sup_{t \in [0,T]} \left| \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial \mathbf{x}_t} \right|. \tag{B.3}$$

Proof. Now as $\mathbf{x}_t$ is continuous and $\boldsymbol{f}_\theta$ is continuously differentiable in $\mathbf{x}$, so $t \mapsto \frac{\partial \boldsymbol{f}_\theta}{\partial [\mathbf{x}_t, \mathbf{z}, \theta]}(\mathbf{x}_t, \mathbf{z}, t)$ is a continuous function on the compact set $[0, T]$, so it is bounded by some $L > 0$. Likewise, for $\mathbf{a} \in \mathbb{R}^d$ the map $(\mathbf{a}, t) \mapsto -\mathbf{a}^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\partial [\mathbf{x}_t, \mathbf{z}, \theta]}$ is Lipschitz in $\mathbf{a}$ with Lipschitz constant L and this constant is independent of t. □

B.3 Proof of Theorem 3.1 when $k = 1$

Proof. First we consider the case of the adjoint state $\mathbf{a}_{\mathbf{x}}(t)$. Recall that the AdjointDEIS-1 solver for $\mathbf{a}_{\mathbf{x}}$ with higher-order error terms is given by

$$\mathbf{a}_{\mathbf{x}}(t_{i+1}) = \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\mathbf{a}_{\mathbf{x}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}^2}{\alpha_{t_{i+1}}^2}\mathbf{a}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{x}_{t_i}} + \mathcal{O}(h_i^2), \tag{B.4}$$

where we let $t_i = t$, $t_{i+1} = s$, $h_i = \lambda_{t_{i+1}} - \lambda_{t_i}$ from Equation (A.21). By Theorem B.1 and Equation (A.21) it holds that

$$\begin{aligned}
\tilde{\mathbf{a}}_{\mathbf{x}}(t_{i+1}) &= \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}^2}{\alpha_{t_{i+1}}^2}\tilde{\mathbf{a}}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_{t_i}, \mathbf{z}, t)}{\partial \tilde{\mathbf{x}}_{t_i}} \\
&= \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}^2}{\alpha_{t_{i+1}}^2}\left(\mathbf{a}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{x}_{t_i}} + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i))\right) \\
&= \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\mathbf{a}_{\mathbf{x}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}^2}{\alpha_{t_{i+1}}^2}\mathbf{a}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{x}_{t_i}} + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i)) \\
&= \mathbf{a}_{\mathbf{x}}(t_{i+1}) + \mathcal{O}(h_{max}^2) + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i)).
\end{aligned} \tag{B.5}$$

Repeat, this argument, from $\tilde{\mathbf{a}}_{\mathbf{x}}(t_0) = \mathbf{a}_{\mathbf{x}}(0)$ then we find

$$\tilde{\mathbf{a}}_{\mathbf{x}}(t_M) = \mathbf{a}_{\mathbf{x}}(T) + \mathcal{O}(Mh_{max}^2) = \mathbf{a}_{\mathbf{x}}(T) + \mathcal{O}(h_{max}). \tag{B.6}$$

Although the argument for the adjoint state $\mathbf{a}_{\mathbf{z}}(t)$ follows an analogous form to the one above we explicitly state it for completeness. Recall that the AdjointDEIS-1 solver for $\mathbf{a}_{\mathbf{z}}$ with higher-order error terms is given by

$$\mathbf{a}_{\mathbf{z}}(t_{i+1}) = \mathbf{a}_{\mathbf{z}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\mathbf{a}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{z}} + \mathcal{O}(h_i^2). \tag{B.7}$$

By Theorem B.1 and Equation (A.22) it holds that

$$\begin{aligned}
\tilde{\mathbf{a}}_{\mathbf{z}}(t_{i+1}) &= \tilde{\mathbf{a}}_{\mathbf{z}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\tilde{\mathbf{a}}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{z}} \\
&= \tilde{\mathbf{a}}_{\mathbf{z}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\left(\mathbf{a}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{z}} + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i))\right) \\
&= \mathbf{a}_{\mathbf{z}}(t_i) + \sigma_{t_{i+1}}(e^{h_i} - 1)\frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\mathbf{a}_{\mathbf{x}}(t_i)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t)}{\partial \mathbf{z}} + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i) \\
&= \mathbf{a}_{\mathbf{z}}(t_{i+1}) + \mathcal{O}(h_{max}^2) + \mathcal{O}(\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i).
\end{aligned} \tag{B.8}$$

Repeat, this argument, from $\tilde{\mathbf{a}}_{\mathbf{z}}(t_0) = \mathbf{0}$ then we find

$$\tilde{\mathbf{a}}_{\mathbf{z}}(t_M) = \mathbf{a}_{\mathbf{z}}(T) + \mathcal{O}(Mh_{max}^2) = \mathbf{a}_{\mathbf{z}}(T) + \mathcal{O}(h_{max}). \tag{B.9}$$

An identical argument can be constructed for $\mathbf{a}_\theta$ thereby finishing the proof. □

B.4 Proof of Theorem 3.1 when $k = 2$

We prove the discretization error of the AdjointDEIS-2M solver. Note for the AdjointDEIS-2M solver we have $h_i = \lambda_{t_{i+1}} - \lambda_{t_{i-1}}$ and $\rho_i = \frac{\lambda_{t_i} - \lambda_{t_{i-1}}}{h_i}$. Furthermore, let $\Delta_i = \|\tilde{\mathbf{a}}_{\mathbf{x}}(t_i) - \mathbf{a}_{\mathbf{x}}(t_i)\|$. Without loss of generality, we will prove this only for $\mathbf{a}_{\mathbf{x}}$; the derivation for $\mathbf{a}_{\mathbf{z}}$ and $\mathbf{a}_\theta$ is analogous.

Proof. First we consider the case of the adjoint state $\mathbf{a}_{\mathbf{x}}(t)$. Recall that the AdjointDEIS-2, see Equation (A.25), solver for $\mathbf{a}_{\mathbf{x}}$ with higher-order error terms is given by

$$\mathbf{a}_{\mathbf{x}}(t_{i+1}) = \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\mathbf{a}_{\mathbf{x}}(t_i) + \frac{1}{\alpha_{t_{i+1}}}\left[\frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}}(e^{h_i} - 1)\mathbf{V}(\mathbf{x}; t_i) + \frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}}(e^{h_i} - h_i - 1)\mathbf{V}^{(1)}(\mathbf{x}; t_i)\right] + \mathcal{O}(h_i^3). \tag{B.10}$$

Taylor's expansion yields

$$\left\|\mathbf{a}_{\mathbf{x}}(t_{i+1})-\Big(\frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\mathbf{a}_{\mathbf{x}}(t_i)+\frac{1}{\alpha_{t_{i+1}}}\Big[\frac{\sigma_{t_{i+1}}}{\sigma_{t_{i+1}}}(e^{h_i}-1)\mathbf{V}(\mathbf{x};t_i)+\frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}}(e^{h_i}-h_i-1)\mathbf{V}^{(1)}(\mathbf{x};t_i)\Big]\Big)\right\| \leq Ch_i^3, \tag{B.11}$$

where C is a constant that depends on $\mathbf{V}^{(2)}(\mathbf{x}_t, t)$. Also note that

$$\left\|\mathbf{V}^{(1)}(\mathbf{x};t_i)-\frac{1}{\rho_i h_i}\big(\mathbf{V}(\mathbf{x};t_i)-\mathbf{V}(\mathbf{x};t_{i-1})\big)\right\| \leq Ch_i. \tag{B.12}$$

Since ρ_i is bounded away from zero, and $e^{-h_i} = 1 - h_i + h_i^2/2 + \mathcal{O}(h_i^3)$, we know

$$\begin{aligned}
&\left\|(e^{h_i}-h_i+1)\mathbf{V}^{(1)}(\mathbf{x};t_i)-\frac{e^{h_i}-1}{2\rho_i}\big(\tilde{\mathbf{V}}(\mathbf{x};t_i)-\tilde{\mathbf{V}}(\mathbf{x};t_{i-1})\big)\right\| \\
&\leq CLh_i(\Delta_i+\Delta_{i-1})+Ch_i^3+\frac{1}{\rho_i}\left|\frac{e^{h_i}-1}{2}-\frac{e^{h_i}-h_i-1}{h_i}\right|\|\mathbf{V}(\mathbf{x};t_i)-\mathbf{V}(\mathbf{x};t_{i-1})\| \\
&\leq CLh_i(\Delta_i+\Delta_{i-1})+Ch_i^3+Ch_i^2\|\mathbf{V}(\mathbf{x};t_i)-\mathbf{V}(\mathbf{x};t_{i-1})\| \\
&\leq CLh_i(\Delta_i+\Delta_{i-1})+CM_ih_i^3,
\end{aligned} \tag{B.13}$$

where $M_i = 1 + \sup_{t_i \leq t \leq t_{i+1}} \|\mathbf{V}^{(1)}(\mathbf{x};t)\|$ and L are the Lipschitz constants of $\mathbf{V}(\mathbf{x};t)$ by Theorem B.1. Then, Δ_{i+1} can be estimated as

$$\begin{aligned}
\Delta_{i+1} &\leq \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\Delta_i+\frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}^2}L\Delta_i+\frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}^2}(CM_ih_i^3+CLh_i(\Delta_i+\Delta_{i+1}))+Ch_i^3 \\
&\leq \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}}\Delta_i+\tilde{C}h_i(\Delta_i+\Delta_{i+1}+h_i^2).
\end{aligned} \tag{B.14}$$

Thus, $\Delta_{i+1} = \mathcal{O}(h_{max}^2)$ as long as h_{max} is sufficiently small and $\Delta_0 + \Delta_1 = \mathcal{O}(h_{max}^2)$, which can be verified via Taylor expansion, thereby finishing the proof. □

C Scheduled Conditional Information

Thus far, we have held the conditional information constant across time, *i.e.*, the same text conditioning like a prompt "fire dragon" can be fed to the noise prediction network. In guided generation tasks it is not uncommon to take advantage of the iterative nature of diffusion models by scheduling the conditional information to exhibit different values at different timesteps. *E.g.*, blending concepts by alternating between the prompt for "fire dragon" and the prompt for "ice dragon" at each timestep to create a picture of a dragon with both the qualities of ice and fire—a technique which has been popularized by Stable Diffusion [3].

We show that converting a constant $\mathbf{z}$ into a scheduled $\mathbf{z}_t$ does not actually change the continuous adjoint equation for $\mathbf{z}_t$, meaning we can still apply the AdjointDEIS-k solvers derived above to find $\mathbf{a}_{\mathbf{z}}(t)$ by simply replacing $\mathbf{z}$ with $\mathbf{z}_t$. We state this observation more formally in the following theorem. The proof is in Appendix C.

Theorem C.1 (Continuous adjoint equations for time-dependent conditional information)**.** *Suppose there exists a function $\mathbf{z} : \mathbb{R} \to \mathbb{R}^z$ which is continuously differentiable in t and is the conditional input into a parameterized vector field, $\boldsymbol{f}_\theta$. Let $\boldsymbol{f}_\theta : \mathbb{R}^d \times \mathbb{R}^z \times \mathbb{R} \to \mathbb{R}^d$ be continuous in t, uniformly Lipschitz in $\mathbf{x}$, and continuously differentiable in $\mathbf{x}$. Let $\mathbf{x} : \mathbb{R} \to \mathbb{R}^d$ be the unique solution for the ODE*

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}_t, t)$$

with initial condition $\mathbf{x}_0$. Then there exists a unique solution $\mathbf{a}_{\mathbf{z}} : \mathbb{R} \to \mathbb{R}^z$ to the following IVP:

$$\mathbf{a}_{\mathbf{z}}(T) = \mathbf{0}, \qquad \frac{\mathrm{d}\mathbf{a}_{\mathbf{z}}}{\mathrm{d}t}(t) = -\mathbf{a}_{\mathbf{x}}(t)^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}_t, t)}{\partial \mathbf{z}_t}.$$

Proof. As $\mathbf{z}(t)$ is continuously differentiable w.r.t. t there exists some function $\mathbf{z}'(t)$ such that

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t}(t) = \mathbf{z}'(t). \tag{C.1}$$

Consider the augmented state defined by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{x}\\ \mathbf{z}\end{bmatrix}(t) = \boldsymbol{f}_{\text{aug}} = \begin{bmatrix}\boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}_t, t)\\ \mathbf{z}'(t)\end{bmatrix}, \tag{C.2}$$

and the associated augmented adjoint state

$$\mathbf{a}_{\text{aug}}(t) \coloneqq \begin{bmatrix}\mathbf{a}_\mathbf{x}\\ \mathbf{a}_\mathbf{z}\end{bmatrix}(t). \tag{C.3}$$

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

$$\frac{\partial \boldsymbol{f}_{\text{aug}}}{\partial[\mathbf{x},\mathbf{z}]} = \begin{bmatrix}\frac{\partial \boldsymbol{f}_\theta(\mathbf{x},\mathbf{z},t)}{\partial \mathbf{x}} & \frac{\partial \boldsymbol{f}_\theta(\mathbf{x},\mathbf{z},t)}{\partial \mathbf{z}}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}. \tag{C.4}$$

Recall that $\mathbf{a}_\mathbf{x}(t)$ evolves with

$$\frac{\mathrm{d}\mathbf{a}_\mathbf{x}}{\mathrm{d}t}(t) = -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}(t), \mathbf{z}(t), t)}{\partial \mathbf{x}(t)}. \tag{C.5}$$

Using Equations (C.3) and (C.4) we can define the evolution of the adjoint augmented state as

$$\frac{\mathrm{d}\mathbf{a}_{\text{aug}}}{\mathrm{d}t}(t) = -\begin{bmatrix}\mathbf{a}_\mathbf{x} & \mathbf{a}_\mathbf{z}\end{bmatrix}(t)\frac{\partial \boldsymbol{f}_{\text{aug}}}{\partial[\mathbf{x},\mathbf{z}]}(t) = -\begin{bmatrix}\mathbf{a}_\mathbf{x}\frac{\partial \boldsymbol{f}_\theta(\mathbf{x},\mathbf{z},t)}{\partial \mathbf{x}} & \mathbf{a}_\mathbf{x}\frac{\partial \boldsymbol{f}_\theta(\mathbf{x},\mathbf{z},t)}{\partial \mathbf{z}}\end{bmatrix}(t). \tag{C.6}$$

Therefore, $\mathbf{a}_\mathbf{z}(t)$ evolves with the ODE

$$\mathbf{a}_\mathbf{z}(T) = 0, \qquad \frac{\mathrm{d}\mathbf{a}_\mathbf{z}}{\mathrm{d}t}(t) = -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{f}_\theta(\mathbf{x}(t), \mathbf{z}(t), t)}{\partial \mathbf{z}(t)}. \tag{C.7}$$

We have thus shown the evolution of $\mathbf{a}_\mathbf{z}(t)$ for some continuously differentiable function $\mathbf{z}(t)$.

Now we prove the solution is unique and exists. As $\mathbf{x}(t)$ is continuous and $\boldsymbol{f}_\theta$ is continuously differentiable in $\mathbf{x}$, it follows that the map $t \mapsto \frac{\partial \boldsymbol{f}_\theta}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{z}(t), t)$ is a continuous function on the compact set $[0, T]$, and therefore it is bounded by some $L > 0$. Correspondingly, for $\mathbf{a} \in \mathbb{R}^d$ it follows that the map $(\mathbf{a}, t) \mapsto -\mathbf{a}^\top \frac{\partial \boldsymbol{f}_\theta}{\partial[\mathbf{x},\mathbf{z}]}(\mathbf{x}(t), \mathbf{z}(t), t)$ is Lipschitz in $\mathbf{a}$ with Lipschitz constant L and this constant is independent of t. Therefore, by the Picard-Lindelölf theorem [44, Theorem 110C] the solution $\mathbf{a}_\mathbf{z}(t)$ exists and is unique. □

While motivated by the case of scheduled conditional information in guided generation with diffusion models, this result applies to neural ODEs more generally, which could open future research directions. For additional clarity, we let $\mathbf{x}(t) \equiv \mathbf{x}_t$ and likewise, $\mathbf{z}(t) \equiv \mathbf{z}_t$.

D Details on Adjoints for SDEs

In this section, we provide further details on the continuous adjoint equations for diffusion SDEs that we omitted from the main paper due to their technical nature and for the purpose of brevity.

Consider the Itô integral given by

$$\mathbf{x}_T = \int_0^T \mathbf{x}_t \, \mathrm{d}\mathbf{w}_t, \tag{D.1}$$

where $\mathbf{x}_t$ is a continuous semi-martingale adapted to the filtration generated by the Wiener process $\{\mathbf{w}_t\}_{t\in[0,T]}$, $\{\mathcal{F}_t\}_{t\in[0,T]}$. The following quantity, however, is not defined

$$\int_T^0 \mathbf{x}_t \, \mathrm{d}\mathbf{w}_t. \tag{D.2}$$

This is because $\mathbf{x}_t$ and $\mathbf{w}_t$ are adapted to $\{\mathcal{F}_t\}_{t\in[0,T]}$ which is defined in forwards time. This means $\mathbf{x}_t$ does not anticipate future events only depends on *past* events. While this is generally sufficient when we wish to integrate backwards in time we want *future* events to inform *past* events.

D.1 Stratonovich Symmetric Integrals and Two-sided Filtration

Clearly, we need a different tool to model this backwards SDE. As such, taking inspiration from the work on neural SDEs [45], we follow the treatment of Kunita [46] for the forward and backward Fisk-Stratonovich integrals using *two-sided filtration*. Let $\{\mathcal{F}_{s,t}\}_{s\leq t;s,t\in[0,T]}$ be a two-sided filtration, where $\mathcal{F}_{s,t}$ is the σ-algebra generated by $\{\mathbf{w}_v - \mathbf{w}_u : s \leq u \leq v \leq t\}$ for $s, t \in [0, T]$ such that $s \leq t$.

Forward time. For a continuous semi-martingale $\{\mathbf{x}_t\}_{t\in[0,T]}$ adapted to the forward filtration $\{\mathcal{F}_{0,t}\}_{t\in[0,t]}$, the Stratonovich stochastic integral is given as

$$\int_0^T \mathbf{x}_t \circ \mathrm{d}\mathbf{w}_t = \lim_{|\Pi|\to 0} \sum_{k=1}^{N} \frac{\mathbf{x}_{t_k} + \mathbf{x}_{t_{k-1}}}{2}(\mathbf{w}_{t_k} - \mathbf{w}_{t_{k-1}}) \tag{D.3}$$

where $\Pi = \{0 = t_0 < \cdots < t_N = T\}$ is a partition of the interval $[0, T]$ and $|\Pi| = \max_k t_k - t_{k-1}$. The forward filtration $\{\mathcal{F}_{0,t}\}_{t\in[0,t]}$ is analogous to the filtration defined in the prior section; therefore, any continuous semi-martingale adapted to it only considers *past* events and does not anticipate *future* events.

Reverse time. Consider the backwards Wiener process $\breve{\mathbf{w}}_t = \mathbf{w}_t - \mathbf{w}_T$ that is adapted to the backward filtration $\{\mathcal{F}_{s,T}\}_{s\in[0,T]}$, then for a continuous semi-martingale $\{\breve{\mathbf{x}}_t\}_{t\in[0,T]}$ adapted to the backward filtration, the backward Stratonovich integral is

$$\int_0^T \breve{\mathbf{x}}_t \circ \mathrm{d}\breve{\mathbf{w}}_t = \lim_{|\Pi|\to 0} \sum_{k=1}^{N} \frac{\breve{\mathbf{x}}_{t_k} + \breve{\mathbf{x}}_{t_{k-1}}}{2}(\breve{\mathbf{w}}_{t_{k-1}} - \breve{\mathbf{w}}_{t_k}) \tag{D.4}$$

The backward filtration $\{\mathcal{F}_{s,T}\}_{s\in[0,T]}$ is the opposite of the forward filtration in the sense that continuous semi-martingales adapted to it only depend on *future* events and do not anticipate *past* events. As such, time is effectively reversed.

Remark D.1. *While the Stratonovich symmetric integrals give us a powerful tool for integrating forwards and backwards in time with stochastic integrals, it is important that we use the* ***same*** *realization of the Wiener process.*

D.2 Stochastic Flow of Diffeomorphisms

Consider the Stratonovich SDE defined as

$$\mathbf{x}_T = \mathbf{x}_0 + \int_0^T \boldsymbol{f}(\mathbf{x}_t, t)\,\mathrm{d}t + \int_0^T \boldsymbol{g}(\mathbf{x}_t, t) \circ \mathrm{d}\mathbf{w}_t, \tag{D.5}$$

where $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{C}_b^{\infty,1}$, *i.e.*, they belong to the class of functions with infinitely many bounded derivatives w.r.t. the state and bounded first derivatives w.r.t. time. Thus, the SDE has a unique strong solution. Given a realization of the Wiener process, there exists a smooth mapping Φ called the *stochastic flow* such that $\Phi_{s,t}(\mathbf{x}_s)$ is the solution at time t of the process described in Equation (D.5) started at $\mathbf{x}_s$ at time $s \leq t$. This then defines a collection of continuous maps $\mathcal{S} = \{\Phi_{s,t}\}_{s\leq t;s,t\in[0,T]}$ from $\mathbb{R}^d$ to itself.

Kunita [46, Theorem 3.7.1] shows that with probability 1 this collection $\mathcal{S}$ satisfies the flow property

$$\Phi_{s,t}(\mathbf{x}_s) = \Phi_{u,t}(\Phi_{s,u}(\mathbf{x}_s)) \quad s \leq u \leq t, \mathbf{x}_s \in \mathbb{R}^d, \tag{D.6}$$

and that each $\Phi_{s,t}$ is a smooth diffeomorphism from $\mathbb{R}^d$ to itself. Hence, $\mathcal{S}$ is the stochastic flow of diffeomorphisms generated by Equation (D.5). Moreover, the backward flow $\breve{\Psi}_{s,t} \coloneqq \Phi_{s,t}^{-1}$ satisfies the backwards SDE:

$$\breve{\Psi}_{s,t}(\mathbf{x}_t) = \mathbf{x}_t - \int_s^t \boldsymbol{f}(\breve{\Psi}_{u,t}(\mathbf{x}_t), u)\,\mathrm{d}u - \int_s^t \boldsymbol{g}(\breve{\Psi}_{u,t}(\mathbf{x}_t), u) \circ \mathrm{d}\breve{\mathbf{w}}_u, \tag{D.7}$$

for all $s, t \in [0, T]$ such that $s \leq t$. This formulation makes intuitive sense as the *backwards* SDE differs only from the *forwards* SDE by a negative sign.

D.3 Continuous Adjoint Equations

Now consider the adjoint flow $\mathbf{A}_{s,t}(\mathbf{x}_s) = \partial \mathcal{L}(\Phi_{s,t}(\mathbf{x}_s))/\partial \mathbf{x}_s$, then $\breve{\mathbf{A}}_{s,t}(\mathbf{x}_t) = \mathbf{A}_{s,t}(\breve{\Psi}_{s,t}(\mathbf{x}_t))$. Li et al. [45] show that $\breve{\mathbf{A}}_{s,t}(\mathbf{x}_t)$ satisfies the backward SDE:

$$\breve{\mathbf{A}}_{s,t}(\mathbf{x}_t) = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} + \int_s^t \breve{\mathbf{A}}_{u,t}(\mathbf{x}_t) \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}_u}(\breve{\Psi}_{u,t}(\mathbf{x}_t), u)\, \mathrm{d}u + \int_s^t \breve{\mathbf{A}}_{u,t}(\mathbf{x}_t) \frac{\partial \boldsymbol{g}}{\partial \mathbf{x}_u}(\breve{\Psi}_{u,t}(\mathbf{x}_t), u) \circ \mathrm{d}\breve{\mathbf{w}}_u. \tag{D.8}$$

As the drift and diffusion coefficient of this SDE are in $\mathcal{C}_b^{\infty,1}$, the system has a unique strong solution.

D.4 Adjoint Diffusion SDE is Actually an ODE

Theorem D.1. *Let $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ be in $\mathcal{C}_b^{\infty,1}$ and $\boldsymbol{g} : \mathbb{R} \to \mathbb{R}^{d\times w}$ be in $\mathcal{C}_b^1$. Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be a scalar-valued differentiable function. Let $\mathbf{w}_t : [0,T] \to \mathbb{R}^w$ be a w-dimensional Wiener process. Let $\mathbf{x} : [0,T] \to \mathbb{R}^d$ solve the Stratonovich SDE*

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\, \mathrm{d}t + \boldsymbol{g}(t) \circ \mathrm{d}\mathbf{w}_t,$$

with initial condition $\mathbf{x}_0$. Then the adjoint process $\mathbf{a}_\mathbf{x}(t) \coloneqq \partial \mathcal{L}(\mathbf{x}_T)/\partial \mathbf{x}_t$ is a strong solution to the backwards-in-time ODE

$$\mathrm{d}\mathbf{a}_\mathbf{x}(t) = -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}_t}(\mathbf{x}_t, t)\, \mathrm{d}t. \tag{D.9}$$

Proof.

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\, \mathrm{d}t + \boldsymbol{g}(t) \circ \mathrm{d}\mathbf{w}_t. \tag{D.10}$$

By Equation (D.8) the adjoint state admitted by the flow of diffeomorphisms generated by Equation (D.10) evolves with the SDE

$$\begin{aligned}\breve{\mathbf{A}}_{s,t}(\mathbf{x}_t) &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} + \int_s^t \breve{\mathbf{A}}_{u,t}(\mathbf{x}_t) \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}_u}(\breve{\Psi}_{u,t}(\mathbf{x}_t), u)\, \mathrm{d}u + \underbrace{\int_s^t \breve{\mathbf{A}}_{u,t}(\mathbf{x}_t) \frac{\partial \boldsymbol{g}}{\partial \mathbf{x}_u}(u) \circ \mathrm{d}\breve{\mathbf{w}}_u}_{=\mathbf{0}} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} + \int_s^t \breve{\mathbf{A}}_{u,t}(\mathbf{x}_t) \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}_u}(\breve{\Psi}_{u,t}(\mathbf{x}_t), u)\, \mathrm{d}u. \end{aligned} \tag{D.11}$$

Clearly, the adjoint state evolves with an ODE revolving around only the drift coefficient, *i.e.*, $\boldsymbol{f}$. Therefore, we can rewrite the evolution of the adjoint state as

$$\mathrm{d}\mathbf{a}_\mathbf{x}(t) = -\mathbf{a}_\mathbf{x}(t)^\top \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}_t}(\mathbf{x}_t, t)\, \mathrm{d}t. \tag{D.12}$$

□

D.5 Converting the Itô SDE to Stratonovich

The diffusion SDE in **??** is defined as an Itô SDE. However, Theorem D.1 is defined as Stratonovich SDEs. However, an Itô SDE can be easily converted into the Stratonovich form, *i.e.*, for some Itô SDE of the form

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\, \mathrm{d}t + \boldsymbol{g}(\mathbf{x}_t, t)\, \mathrm{d}\mathbf{w}_t \tag{D.13}$$

with a differentiable function σ, there exists a corresponding Stratonovich SDE of the form

$$\mathrm{d}\mathbf{x}_t = [\boldsymbol{f}(\mathbf{x}_t, t) + \frac{1}{2} \frac{\partial \boldsymbol{g}}{\partial \mathbf{x}}(\mathbf{x}_t, t) \cdot \boldsymbol{g}(\mathbf{x}_t, t)]\, \mathrm{d}t + \boldsymbol{g}(\mathbf{x}_t, t) \circ \mathrm{d}\mathbf{w}_t. \tag{D.14}$$

As **??** is defined such that $\boldsymbol{g}(\mathbf{x}_t, t) = g(t)$ and is independent of the state $\mathbf{x}_t$, then the SDE may be written in Stratonovich form as

$$\mathrm{d}\mathbf{x}_t = \left[f(t)\mathbf{x}_t + \frac{g^2(t)}{\sigma_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t) \right] \mathrm{d}t + g(t) \circ \mathrm{d}\bar{\mathbf{w}}_t. \tag{D.15}$$

D.6 Solving Backwards Diffusion SDEs

Lu et al. [17] propose the following first-order solver for diffusion SDEs

$$\mathbf{x}_t = \frac{\alpha_t}{\alpha_s}\mathbf{x}_s - 2\sigma_t(e^h - 1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_s, s) + \sigma_t\sqrt{e^{2h} - 1}\boldsymbol{\epsilon}_s, \tag{D.16}$$

where $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To solve the SDE backwards in time we follow the approach initially proposed by Wu and la Torre [47] and used by later works [27]. Given a particular realization of the Wiener process that admits $\mathbf{x}_t \sim \mathcal{N}(\alpha_t\mathbf{x}_0 \mid \sigma_t^2\mathbf{I})$, then for two samples $\mathbf{x}_t$ and $\mathbf{x}_s$ the noise $\boldsymbol{\epsilon}_s$ can be calculated by rearranging Equation (D.16) to find

$$\boldsymbol{\epsilon}_s = \frac{\mathbf{x}_t - \frac{\alpha_t}{\alpha_s}\mathbf{x}_s + 2\sigma_t(e^h - 1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_s, \mathbf{z}, s)}{\sigma_t\sqrt{e^{2h} - 1}} \tag{D.17}$$

With this the sequence $\{\boldsymbol{\epsilon}_{t_i}\}_{i=1}^N$ of added noises can be calculated which will **exactly** reconstruct the original input from the initial realization of the Wiener process. This technique is referred to as *Cycle-SDE* after the CycleDiffusion paper [47].

E Additional Experiments

In this section, we include some additional experiments that did not fit within the main paper.

(a) $M = 5$ (b) $M = 10$ (c) $M = 15$ (d) $M = 20$

Figure 4: Morphed faces created by guided generation with AdjointDEIS with differing number of discretization steps.

E.1 Impact of Discretization Steps

One of the advantages of AdjointDEIS is that the solver for the diffusion ODE and continuous adjoint equations are distinct. This means that we don't have to force $N = M$ enabling greater flexibility when using AdjointDEIS. As such we explore the impact of using fewer steps to estimate the gradient while keeping the number of sampling steps $N = 20$ fixed. In Figure 4 we illustrate the impact of the change in the number of discretization steps when estimating the gradients. Unsurprisingly, the fewer steps we take the less accurate the gradients are. This matches the empirical data presented in Table 2 which measures the impact of face morphing performance measured in MMPMR.

Table 2: Impact of number of discretization steps, M, on face morphing with AdjointDEIS. FMR = 0.1%.

	MMPMR(↑)		
M (↓)	**AdaFace**	**ArcFace**	**ElasticFace**
20	**99.8**	**98.77**	**99.39**
15	94.89	90.59	94.07
10	94.27	91.21	92.84
05	69.94	60.74	64.21

(a) $\eta = 0.01$ (b) $\eta = 0.1$ (c) $\eta = 1$

Figure 5: Morphed faces created by guided generation with AdjointDEIS with different learning rates. All used $M = 20$ the ODE variant.

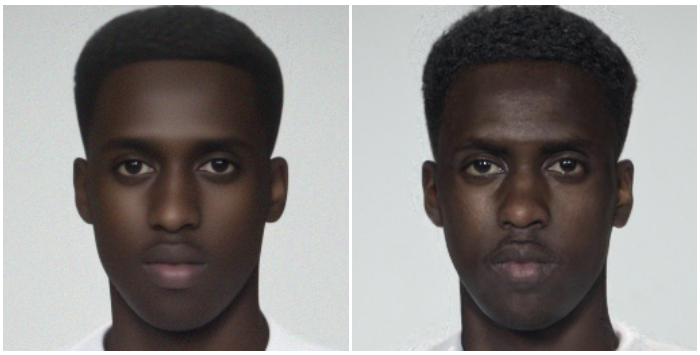
E.2 Impact of Learning Rate

We measure the impact of the learning rate on guided generation with AdjointDEIS in Table 2. Unsurprisingly, large learning rates lower performance, especially for less accurate gradients. *I.e.*, when M is small. We illustrate an example of the impact in Figure 5. Clearly, the learning rate of $\eta = 1$ starts to distort the images even if it still fools the FR system.

Table 3: Impact of learning rate, η, on face morphing with AdjointDEIS. FMR = 0.1%.

			MMPMR(↑)		
SDE	η	M (↓)	**AdaFace**	**ArcFace**	**ElasticFace**
✗	1	20	98.77	98.98	98.77
✗	0.1	20	**99.8**	98.77	**99.39**
✗	0.01	20	95.5	92.64	95.91
✗	1	10	50.92	49.69	50.92
✗	0.1	10	94.27	91.21	92.84
✗	1	05	2.66	2.04	1.84
✗	0.1	05	69.94	60.74	64.21
✓	1	20	98.57	**99.59**	98.98
✓	0.1	20	98.57	97.96	97.75

E.3 Number of Steps



(a) $N = 20$ (b) $N = 50$

Figure 6: Morphed faces created by guided generation with AdjointDEIS with different number of sampling steps. SDE solver, $M = N$.

As alluded to in the main paper, one of the drawbacks of diffusion SDEs is that they require small step sizes to work properly. We observe that the missing high frequency content is added back in when the step size is increased, see Figure 6.

F Implementation Details

F.1 AdjointDEIS-2M Algorithm

For completeness we have the full AdjointDEIS-2M solver implemented in Algorithm 1 for solving the continuous adjoint equations for diffusion ODEs. We assume there is another solver which solves the backwards ODE to yield $\{\tilde{\mathbf{x}}_{t_i}\}_{i=0}^M$. Remark that $\mathbf{a}_{\text{aug}} := [\mathbf{a}_{\mathbf{x}}, \mathbf{a}_{\mathbf{z}}, \mathbf{a}_\theta]$. Also Algorithm 1 can be used to solve the continuous adjoint equations for diffusion SDEs by simply adding the factor of 2 into the update equations.

Algorithm 1 AdjointDEIS-2M.

Require: Initial values $\mathbf{a}_{\mathbf{x}}(0)$, monotonically increasing time steps $\{t_i\}_{i=0}^M$, and noise prediction model $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)$.

Denote $h_i := \lambda_{t_{i+1}} - \lambda_{t_i}$, for $i = 0, \dots, M-1$.
$\tilde{\mathbf{a}}_{\mathbf{x}}(t_0) \leftarrow \mathbf{a}_{\mathbf{x}}(0)$ ▷ Initialize an empty buffer Q.
$\tilde{\mathbf{a}}_{\mathbf{z}}(t_0) \leftarrow \mathbf{0}, \tilde{\mathbf{a}}_\theta(t_0) \leftarrow \mathbf{0}$.
$Q \xleftarrow{\text{buffer}} [\tilde{\mathbf{V}}(\mathbf{x}, t_0), \tilde{\mathbf{V}}(\mathbf{z}, t_0), \tilde{\mathbf{V}}(\theta, t_0)]$
$\tilde{\mathbf{a}}_{\mathbf{x}}(t_1) \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_1}} \tilde{\mathbf{a}}_{\mathbf{x}}(t_0) + \sigma_{t_1}(e^{h_0} - 1) \frac{\alpha_{t_0}^2}{\alpha_{t_1}^2} \tilde{\mathbf{a}}_{\mathbf{x}}(t_0)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_{t_0}, \mathbf{z}, t_0)}{\partial \tilde{\mathbf{x}}_{t_0}}$
$\tilde{\mathbf{a}}_{\mathbf{z}}(t_1) \leftarrow \tilde{\mathbf{a}}_{\mathbf{z}}(t_0) + \sigma_{t_1}(e^{h_0} - 1) \frac{\alpha_{t_0}}{\alpha_{t_1}} \tilde{\mathbf{a}}_{\mathbf{x}}(t_0)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_{t_0}, \mathbf{z}, t_0)}{\partial \mathbf{z}}$
$\tilde{\mathbf{a}}_\theta(t_1) \leftarrow \tilde{\mathbf{a}}_\theta(t_0) + \sigma_{t_1}(e^{h_0} - 1) \frac{\alpha_{t_0}}{\alpha_{t_1}} \tilde{\mathbf{a}}_{\mathbf{x}}(t_0)^\top \frac{\partial \boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_{t_0}, \mathbf{z}, t_0)}{\partial \theta}$
$Q \xleftarrow{\text{buffer}} [\tilde{\mathbf{V}}(\mathbf{x}, t_1), \tilde{\mathbf{V}}(\mathbf{z}, t_1), \tilde{\mathbf{V}}(\theta, t_1)]$
for $i \leftarrow 1, 2, \dots, M-1$ **do**
    $\rho_i \leftarrow \frac{h_{i-1}}{h_i}$
    $\mathbf{D}_i \leftarrow \left(1 + \frac{1}{2\rho_i}\right) \tilde{\mathbf{V}}(\mathbf{x}; t_i) - \frac{1}{2\rho_i} \tilde{\mathbf{V}}(\mathbf{x}; t_{i-1})$
    $\mathbf{E}_i \leftarrow \left(1 + \frac{1}{2\rho_i}\right) \tilde{\mathbf{V}}(\mathbf{z}; t_i) - \frac{1}{2\rho_i} \tilde{\mathbf{V}}(\mathbf{z}; t_{i-1})$
    $\mathbf{F}_i \leftarrow \left(1 + \frac{1}{2\rho_i}\right) \tilde{\mathbf{V}}(\theta; t_i) - \frac{1}{2\rho_i} \tilde{\mathbf{V}}(\theta; t_{i-1})$
    $\tilde{\mathbf{a}}_{\mathbf{x}}(t_{i+1}) \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i+1}}} \tilde{\mathbf{a}}_{\mathbf{x}}(t_i) + \frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}^2}(e^{h_i} - 1)\mathbf{D}_i$
    $\tilde{\mathbf{a}}_{\mathbf{z}}(t_{i+1}) \leftarrow \tilde{\mathbf{a}}_{\mathbf{z}}(t_i) + \frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}}(e^{h_i} - 1)\mathbf{E}_i$
    $\tilde{\mathbf{a}}_\theta(t_{i+1}) \leftarrow \tilde{\mathbf{a}}_\theta(t_i) + \frac{\sigma_{t_{i+1}}}{\alpha_{t_{i+1}}}(e^{h_i} - 1)\mathbf{F}_i$
    if $i < M-1$ **then**
        $Q \xleftarrow{\text{buffer}} [\tilde{\mathbf{V}}(\mathbf{x}, t_{i+1}), \tilde{\mathbf{V}}(\mathbf{z}, t_{i+1}), \tilde{\mathbf{V}}(\theta, t_{i+1})]$
    end if
end for
return $\tilde{\mathbf{a}}_{\mathbf{x}}(t_M), \tilde{\mathbf{a}}_{\mathbf{z}}(t_M), \tilde{\mathbf{a}}_\theta(t_M)$.

F.2 Code

Our code for AdjointDEIS will soon be available here at https://github.com/zblasingame/AdjointDEIS.

F.3 Repositories Used

For reproducibility purposes, we provide a list of links to the official repositories of other works used in this paper.

1. The SYN-MAD 2022 dataset used in this paper can be found at https://github.com/marcohuber/SYN-MAD-2022.
2. The ArcFace models, MS1M-RetinaFace dataset, and MS1M-ArcFace dataset can be found at https://github.com/deepinsight/insightface.
3. The ElasticFace model can be found at https://github.com/fdbtrs/ElasticFace.
4. The AdaFace model can be found at https://github.com/mk-minchul/AdaFace.
5. The official Diffusion Autoencoders repository can be found at https://github.com/phizaz/diffae.
6. The official MIPGAN repository can be found at https://github.com/ZHYYYYYYYYYYY/MIPGAN-face-morphing-algorithm.

G Experimental Details

In this section, we outline the details for the experiments run in Section 5.

G.1 DiM Algorithm

For completeness, we provide the DiM algorithm from [28] following the notation used in [48]. The original bona fide images are denoted $\mathbf{x}_0^{(a)}$ and $\mathbf{x}_0^{(b)}$. The conditional encoder is $\mathcal{E} : \mathcal{X} \to \mathcal{Z}$, Φ is the numerical diffusion ODE solver, Φ^+ is the numerical diffusion ODE solver as time runs forwards from 0 to T. The algorithm is presented in Algorithm 2.

Algorithm 2 DiM Framework.

Require: Blend parameter $w = 0.5$. Time schedule $\{t_i\}_{i=1}^N \subseteq [0, T], t_i < t_{i+1}$.

$\mathbf{z}_a \leftarrow \mathcal{E}(\mathbf{x}_0^{(a)})$ ▷ Encoding bona fides into conditionals.
$\mathbf{z}_b \leftarrow \mathcal{E}(\mathbf{x}_0^{(b)})$
for $i \leftarrow 1, 2, \ldots, N-1$ **do**
    $\mathbf{x}_{t_{i+1}}^{(a)} \leftarrow \Phi^+(\mathbf{x}_{t_i}^{(a)}, \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}^{(a)}, \mathbf{z}_a, t_i), t_i)$ ▷ Solving the probability flow ODE as time runs from 0 to T.
    $\mathbf{x}_{t_{i+1}}^{(b)} \leftarrow \Phi^+(\mathbf{x}_{t_i}^{(b)}, \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}^{(b)}, \mathbf{z}_b, t_i), t_i)$
end for
$\mathbf{x}_T^{(ab)} \leftarrow \text{slerp}(\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)}; w)$ ▷ Morph initial noise.
$\mathbf{z}_{ab} \leftarrow \text{lerp}(\mathbf{z}_a, \mathbf{z}_b; w)$ ▷ Morph conditionals.
for $i \leftarrow N, N-1, \ldots, 2$ **do**
    $\mathbf{x}_{t_{i-1}}^{(ab)} \leftarrow \Phi(\mathbf{x}_{t_i}^{(ab)}, \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}^{(ab)}, \mathbf{z}_{ab}, t_i), t_i)$ ▷ Solving the probability flow ODE as time runs from T to 0.
end for
return $\mathbf{x}_0^{(ab)}$

G.2 NFE

In our reporting of the NFE we record the number of times the diffusion noise prediction U-Net is evaluated both during the encoding phase, N_E, and solving of the PF-ODE or diffusion SDE, N. We chose to report $N + N_E$ over $N + 2N_E$ as even though two bona fide images are encoded resulting in $2N_E$ NFE during encoding, this process can simply be batched together, reducing the NFE down to N_E. When reporting the NFE for the Morph-PIPE model, we report $N_E + BN$ where B is the number of blends. While a similar argument can be made that the morphed candidates could be generated in a large batch of size B, reducing the NFE of the sampling process down to N, we chose to report BN as the number of blends, $B = 21$, used in the Morph-PIPE is quite large, potentially resulting in Out Of Memory (OOM) errors, especially if trying to process a mini-batch of morphs. Using $N_E + N$ reporting over $N_E + BN$, the NFE of Morph-PIPE is 350, which is comparable to DiM. The reporting of NFE for AdjointDEIS was calculated as $N_E + n_{opt}(N + M)$ where n_opt is the number of optimization steps and M is the number of discretization steps for the continuous adjoint equations.

G.3 Hardware

All of the main experiments were done on a single NVIDIA Tesla V100 32GB GPU. On average the guided generation experiments for our approach took between 6 - 8 hours for the whole dataset of face morphs with a batch size of 8. Some additional follow-up work for the camera-ready version used an NVIDIA H100 Tensor Core 80GB GPU with a batch size of 16.

G.4 Datasets

The SYN-MAD 2022 dataset is derived from the Face Research Lab London (FRLL) dataset [39]. FRLL is a dataset of high-quality captures of 102 different individuals with frontal images and neutral lighting. There are two images per subject, an image of a "neutral" expression and one of a "smiling" expression. The ElasticFace [40] FR system was used to select the top 250 most similar pairs, in terms of cosine similarity, of bona fide images for both genders, resulting in a total of 489 bona fide image pairs for face morphing [38], as some pairs did not generate good morphs on the reference set; we follow this minimal subset.

G.5 FR Systems

All three FR systems use the Improved ResNet (IResNet-100) architecture [49] as the neural net backbone for the FR system. The ArcFace model is a widely used FR system [28, 30, 32, 36]. It employs an additive angular margin loss to enforce intra-class compactness and inter-class distance, which can enhance the discriminative ability of the feature embeddings [35]. ElasticFace builds upon the ArcFace model by using an elastic penalty margin over the fixed penalty margin used by ArcFace. This change results in an FR system with state-of-the-art performance [40]. Lastly, the AdaFace model employs an adaptive margin loss by weighting the loss relative to an approximation of the image quality [41]. The image quality is approximated via feature norms and is used to give less weight to misclassified images, reducing the impact of "low" quality images on training. This improvement allows the AdaFace model to achieve state-of-the-art performance in FR tasks.

The AdaFace and ElasticFace models are trained on the MS1M-ArcFace dataset, whereas the ArcFace model is trained on the MS1M-RetinaFace dataset. *N.B.*, the ArcFace model used in the identity loss is not the same ArcFace model used during evaluation. The model used in the identity loss is an IResNet-100 trained on the Glint360k dataset [50] with the ArcFace loss. We use the cosine distance to measure the distance between embeddings from the FR models. All three FR systems require images of 112×112 pixels. We resize every image, post alignment from dlib which ensures the images are square, to 112×112 using bilinear down-sampling. The image tensors are then normalized such that they take values in $[-1, 1]$. Lastly, the AdaFace FR system was trained on BGR images so the image tensor is shuffled from the RGB format to the BGR format.

H Additional Related Work

In this section we compare several recent methods for training-free guided generation. We broadly classify these techniques into two categories:

1. Techniques which directly optimize the solution trajectory during sampling [48, 51, 52]
2. Techniques which search for the optimal latents $\mathbf{x}_T$ and or $\mathbf{z}$ (this can include optimizing the solution trajectory as well) [53, 54].

In Table 4 we compare several different techniques for training-free guided diffusion along this category along with whether the formulation is for diffusion ODEs, SDEs, or both.

FlowGrad [52] controls the generative process by solving the following optimal control problem

$$\min_{\boldsymbol{u}} \quad \mathcal{L}(\mathbf{x}_0) + \lambda \int_T^0 \|\boldsymbol{u}(t)\|^2 \, \mathrm{d}t, \tag{H.1}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \mathbf{x}_T + \int_T^0 \boldsymbol{f}_\theta(\mathbf{x}_t, \mathbf{z}, t) + \boldsymbol{u}(t) \, \mathrm{d}t \tag{H.2}$$

Table 4: Comparison of different guidance methods for diffusion models.

Method	**ODE**	**SDE**	**Optimize** $\mathbf{x}_T$	**Optimize** $(\mathbf{z}, \theta)$
FlowGrad [52]	✓	✗	✗	✗
FreeDoM [51]	✗	✓	✗	✗
Greedy [48]	✓	✓	✗	✗
DOODL [53]	✗	✓	✓	✗
AdjointDPM [54]	✓	✗	✓	✓
Implicit Diffusion [55]	✓	✓	✓	✓
AdjointDEIS	✓	✓	✓	✓

where $\boldsymbol{u}$ is the control function. This optimization objective learns to alter the flow, $\boldsymbol{f}_\theta$, by $\boldsymbol{u}$. In practice, this amounts to injecting a control step governed by $\boldsymbol{u}(t)$ for a discretized schedule. This technique does not allow for learning an optimal $\mathbf{x}_T$, $\mathbf{z}$, or θ.

FreeDoM [51] looks at gradient guided generation of images by calculating the gradient w.r.t. $\mathbf{x}_t$ by using the approximated clean image

$$\mathbf{x}_0 \approx \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\alpha_t} \tag{H.3}$$

at each timestep. Let $h_i = \lambda_{t_i} - \lambda_{t_{i-1}}$. The strategy can be described as

$$\mathbf{x}_{t_{i-1}} = \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}} \mathbf{x}_{t_i} - 2\sigma_{t_{i-1}}(e^{h_i} - 1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t_i) + \sigma_{t_{i-1}}\sqrt{e^{2h_i} - 1}\boldsymbol{\epsilon}_{t_i}, \tag{H.4}$$

$$\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_{t_i} - \sigma_{t_i}\boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t_i)}{\alpha_{t_i}}, \tag{H.5}$$

$$\mathbf{g}_{t_i} = \frac{\partial \mathcal{L}(\hat{\mathbf{x}}_0)}{\partial \mathbf{x}_t}, \tag{H.6}$$

$$\mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_{i-1}} - \eta_{t_i}\mathbf{g}_{t_i}, \tag{H.7}$$

where η_{t_i} is a learning rate defined per timestep. Importantly, FreeDoM operates on diffusion SDEs. They have an addition algorithm in which the add noise back to the image, in essence going back one timestep, and applying the guidance step again.

Similar to FreeDoM, greedy guided generation [48] looks to alter the generative trajectory by injecting the gradient defined on the approximated clean image; however, this technique does so w.r.t. the prediction noise, *i.e.*,

$$\boldsymbol{\epsilon}'_{t_i} = \text{stopgrad}(\boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, \mathbf{z}, t_i)) \tag{H.8}$$

$$\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_{t_i} - \sigma_{t_i}\boldsymbol{\epsilon}'_{t_i}}{\alpha_{t_i}}, \tag{H.9}$$

$$\mathbf{g}_{t_i} = \frac{\partial \mathcal{L}(\hat{\mathbf{x}}_0)}{\partial \boldsymbol{\epsilon}_{t_i}}, \tag{H.10}$$

$$\boldsymbol{\epsilon}'_{t_i} = \boldsymbol{\epsilon}'_{t_i} - \eta_{t_i}\mathbf{g}_{t_i}. \tag{H.11}$$

The technique would work for either diffusion ODEs or SDEs.

DOODL [53] looks at gradient calculation based on the invertibility of EDICT [56]. This method can find the gradient w.r.t. $\mathbf{x}_T$; however, it cannot for the other quantities. DOODL additionally has further overhead due to the dual diffusion process of EDICT. Further analysis of DOODL compared to continuous adjoint equations for diffusion models can be found in [54].

More closely related to our work is the recent AdjointDPM [54] who also explores the use of adjoint sensitivity methods for backpropagation through the *probability flow* ODE. While they also propose to use the continuous adjoint equations to find gradients for diffusion models, our work differs in several ways which we enumerate in Table 5. For clarity, we use orange to denote their notation. They reparameterize Equation (2.3) as

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\rho} = \tilde{\boldsymbol{\epsilon}}_\theta(e^{\int_0^{\gamma^{-1}(\rho)} f(\tau)\,\mathrm{d}\tau}\mathbf{y}, \gamma^{-1}(\rho), c) \tag{H.12}$$

Table 5: Comparison of adjoint sensitivity algorithms for diffusion models.

	AdjointDPM [54]	**AdjointDEIS**
Discretization domain	ϵ_θ over ρ	ϵ_θ over λ
Solver type	Black box ODE solver	Custom solver
Closed form SDE coefficients	✗	✓
Interoperability with existing samplers	✗	✓
Decoupled ODE schedule	✗	✓
Supports SDEs	✗	✓

where c denotes the conditional information, $\rho = \gamma(t)$, and $\frac{\mathrm{d}\gamma}{\mathrm{d}t} = e^{-\int_0^t f(\tau)\,\mathrm{d}\tau} \frac{g^2(t)}{2\sigma_t}$. which gives them the following ODE for calculating the adjoint.

$$\frac{\mathrm{d}}{\mathrm{d}\rho}\left[\frac{\partial \mathcal{L}}{\partial \mathbf{y}_\rho}\right] = -\frac{\partial \mathcal{L}}{\partial \mathbf{y}_\rho}^\top \frac{\partial \boldsymbol{\epsilon}_\theta(e^{\int_0^{\gamma^{-1}(\rho)} f(\tau)\,\mathrm{d}\tau}\mathbf{y}_\rho, \mathbf{z}, \gamma^{-1}(\rho))}{\partial \mathbf{y}_\rho} \tag{H.13}$$

In our approach we integrate over λ_t whereas they integrate over ρ. Moreover, we provide custom solvers designed specifically for diffusion ODEs instead of using a black box ODE solver. Our approach is also interoperable with other forward ODE solvers meaning our AdjointDEIS solver is agnostic to the ODE solver used to generate the output; however, the AdjointDPM model is tightly coupled to its forward solver. Lastly and most importantly our method is more general and supports diffusion SDEs, not just ODEs.

Although the original paper omitted a closed form expression for $\gamma^{-1}(\rho)$, we provide to give a comparison between both methods, and to ensure AdjointDPM can be fully implemented. In the VP SDE scheme with a linear noise schedule $\log \alpha_t$ is found to be

$$\log \alpha_t = -\frac{\beta_1 - \beta_0}{4}t^2 - \frac{\beta_0}{2}t \tag{H.14}$$

on $t \in [0, 1]$ with $\beta_0 = 0.1, \beta_1 = 20$, following Song et al. [15]. Then $\gamma^{-1}(\rho)$ is found to be

$$\gamma^{-1}(\rho) = \frac{\beta_0 - \sqrt{\beta_0^2 + 4\log\frac{1}{\sqrt{\frac{1}{\alpha_0^2}(\rho+\sigma_0)^2+1}}(\beta_0 - \beta_1)}}{\beta_0 - \beta_1} \tag{H.15}$$

Concurrent work to ours by Marion et al. [55] has also explored the method of adjoint sensitivity for guidance of diffusion models. They, however, focus on an efficient scheme to parallelize the solution to the adjoint ODE from the perspective of bi-level optimization rather than the adjoint technique itself. So while we focused on the details of the continuous adjoint equations, they focused on an efficient implementation of the optimization problem from the perspective of bi-level optimization.

I Analytic Formulations of Drift and Diffusion Coefficients

For completeness, we show how to analytically compute the drift and diffusion coefficients for a linear noise schedule Ho et al. [1] in the VP scenario Song et al. [15]. With a linear noise schedule $\log \alpha_t$ is found to be

$$\log \alpha_t = -\frac{\beta_1 - \beta_0}{4}t^2 - \frac{\beta_0}{2}t \tag{I.1}$$

on $t \in [0, 1]$ with $\beta_0 = 0.1, \beta_1 = 20$, following Song et al. [15]. The drift coefficient becomes

$$f(t) = -\frac{\beta_1 - \beta_0}{2}t - \frac{\beta_0}{2} \tag{I.2}$$

and as $\sigma_t = \sqrt{1 - \alpha_t^2}$ we find

$$\begin{aligned}\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left[1 - \exp\left(-\frac{\beta_1 - \beta_0}{4}t^2 - \frac{\beta_0}{2}t\right)^2\right] \\ &= ((\beta_1 - \beta_0)t + \beta_0)\exp\left(-\frac{\beta_1 - \beta_0}{2}t^2 - 2\beta_0 t\right)\end{aligned} \tag{I.3}$$

Therefore, the diffusion coefficient $g^2(t)$ is

$$g^2(t) = \underbrace{((\beta_1 - \beta_0)t + \beta_0)\exp\left(-\frac{\beta_1 - \beta_0}{2}t^2 - 2\beta_0 t\right)}_{\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}} + \underbrace{\big((\beta_1 - \beta_0)t + \beta_0\big)\left[1 - \exp\left(-\frac{\beta_1 - \beta_0}{4}t^2 - \frac{\beta_0}{2}t\right)^2\right]}_{-2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2} \tag{I.4}$$

Importantly, $\frac{\mathrm{d}\sigma_t}{\mathrm{d}t}$ does not exist at time $t = 0$, as σ_t is discontinuous at that point, and so an approximation is needed when starting from this initial step. In practice, adding a small $\epsilon \ll 1$ to $t = 0$ should suffice.

The inverse of λ_t is found via

$$\Lambda^{-1}(\lambda) = \frac{-\beta_0 + \sqrt{\beta_0^2 - 4(\beta_1 - \beta_0)\log\sqrt{\frac{e^{2\lambda}}{1+e^{2\lambda}}}}}{\beta_1 - \beta_0} \tag{I.5}$$

Simplifies to

$$\Lambda^{-1}(\lambda) = \frac{-\beta_0 + \sqrt{\beta_0 - 2\big(2\lambda - \log(1 + e^{2\lambda})\big)(\beta_1 - \beta_0)}}{\beta_1 - \beta_0} \tag{I.6}$$

$$\alpha_\lambda = \exp\left(\frac{\beta_0^2 - 4\beta_0\lambda + 2\beta_0\log\left(e^{2\lambda} + 1\right) - \beta_0 + 4\beta_1\lambda - 2\beta_1\log\left(e^{2\lambda} + 1\right)}{4\left(\beta_1 - \beta_0\right)}\right) = \frac{e^\lambda}{\sqrt{e^{2\lambda} + 1}} \tag{I.7}$$