Fast Sampling of Diffusion Models with Exponential Integrator

Qinsheng Zhang
Georgia Institute of Technology
qzhang419@gatech.edu

Yongxin Chen
Georgia Institute of Technology
yongchen@gatech.edu

Abstract

The past few years have witnessed the great success of Diffusion models (DMs) in generating high-fidelity samples in generative modeling tasks. A major limitation of the DM is its notoriously slow sampling procedure which normally requires hundreds to thousands of time discretization steps of the learned diffusion process to reach the desired accuracy. Our goal is to develop a fast sampling method for DMs with a much less number of steps while retaining high sample quality. To this end, we systematically analyze the sampling procedure in DMs and identify key factors that affect the sample quality, among which the method of discretization is most crucial. By carefully examining the learned diffusion process, we propose Diffusion Exponential Integrator Sampler (DEIS). It is based on the Exponential Integrator designed for discretizing ordinary differential equations (ODEs) and leverages a semilinear structure of the learned diffusion process to reduce the discretization error. The proposed method can be applied to any DMs and can generate high-fidelity samples in as few as 10 steps. In our experiments, it takes about 3 minutes on one A6000 GPU to generate $50k$ images from CIFAR10. Moreover, by directly using pre-trained DMs, we achieve the state-of-art sampling performance when the number of score function evaluation (NFE) is limited, e.g., 4.17 FID with 10 NFEs, 2.86 FID with only 20 NFEs on CIFAR10. [1]

1 Introduction

The Diffusion model (DM) [13] is a generative modeling method developed recently that relies on the basic idea of reversing a given simple diffusion process. A time-dependent score function is learned for this purpose and DMs are thus also known as score-based models [35]. Compared with other generative models such as generative adversarial networks (GANs), in addition to great scalability, the DM has the advantage of stable training; it avoids mode-collapsing and is not hyperparameter sensitive [6, 22]. DMs have recently achieved impressive performances on a variety of tasks, including unconditional image generation [13, 35, 32, 9], text conditioned image generation [29, 31], text generation [15, 2], 3D point cloud generation [27], inverse problem [20, 37], etc.

However, the remarkable performance of DMs comes at the cost of extremely slow sampling; it takes much longer time to produce high-quality samples compared with GANs. For instance, the Denoising Diffusion Probabilistic Model (DDPM) [13] needs 1000 steps to generate one sample and each step requires evaluating the learning neural network once; this is substantially slower than GANs [11, 18]. For this reason, there exist several studies aiming at improve the sampling speed for DMs (More related works are discussed in Appendix A). One category of methods modify/optimize the forward noising process such that backward denoising process can be more efficient [28, 35, 44, 3]. An important and effective instance is the Denoising Diffusion Implicit Model (DDIM) [34] that uses

[1] Code is available at https://github.com/qsh-zh/deis

Preprint. Under review.

Figure 1: Generated images with various DMs. Latent diffusion [32] (Left), 256×256 image with text *A shirt with inscription "World peace"* (15 NFE). VE diffusion [35] (Mid), FFHQ 256×256 (12 NFE). VP diffusion [13] (Right), CIFAR10 (7 NFE) and CELEBA (5 NFE).

a non-Markovian noising process. Another category of methods speed up the numerical solver for stochastic differential equations (SDEs) or ordinary differential equations (ODEs) associated with the DMs [17, 35, 38]. In [35], blackbox ODE solvers are used to solve a marginal equivalent ODE known as the Probability Flow (PF), for fast sampling. In [25], the authors combine DDIM with high order methods to solve this ODE and achieve further acceleration. Note that the deterministic DDIM can also be viewed as a time discretization of the PF as it matches the latter in the continuous limit [34, 25]. However, it is unclear why DDIM works better than generic methods such as Euler.

The objective of this work is to establish a principled discretization scheme for the learned backward diffusion processes in DMs so as to achieve fast sampling. Since the most expensive part in sampling a DM is the evaluation of the neural network that parameterizes the backward diffusion, we seek a discretization method that requires a small number of network function evaluation (NFE). We start with a family of marginal equivalent SDEs/ODEs associated with DMs and investigate numerical error sources, which include fitting error and discretization error. We observe that even with the same trained model, different discretization schemes can have dramatically different performances in terms of discretization error. We then carry out a sequence of experiments to systematically investigate the influences of different factors on the discretization error. We find out that the *Exponential Integrator (EI)* [14] that utilizes the semilinear structure of the backward diffusion has minimum error. To further reduce the discretization error, we propose to either use high order polynomials to approximate the nonlinear term in the ODE or employ Runge Kutta methods on a transformed ODE. The resulting algorithms, termed *Diffusion Exponential Integrator Sampler (DEIS)*, achieves the best sampling quality with limited NFEs.

Our contributions are summarized as follows: 1) We investigate a family of marginal equivalent SDEs/ODEs for fast sampling and conduct a systematic error analysis for their numerical solvers. 2) We propose DEIS, an efficient sampler that can be applied to any DMs to achieve superior sampling quality with a limited number of NFEs. 3) We prove that the deterministic DDIM is a special case of DEIS, justifying the effectiveness of DDIM from a discretization perspective. 4) We conduct comprehensive experiments to validate the efficacy of DEIS. DEIS generates high quality synthetic images with 12 steps, compared with 1000 steps for DDIM (83x speedup).

2 Background on Diffusion Models

A DM consists of a fixed forward diffusion (noising) process that adds noise to the data, and a learned backward diffusion (denoising) process that gradually removes the added noise. The backward diffusion is trained to match the forward one in probability law, and when this happens, one can in principle generate perfect samples from the data distribution by simulating the backward diffusion.

Forward noising diffusion: The forward diffusion of a DM for D-dimensional data is a linear diffusion described by the stochastic differential equation (SDE) [33]

$$d\boldsymbol{x} = \boldsymbol{F}_t \boldsymbol{x} dt + \boldsymbol{G}_t d\boldsymbol{w}, \tag{1}$$

Table 1: Two popular SDEs, variance preserve SDE (VPSDE) and variance exploding SDE (VESDE), used in DMs. The parameter α_t is decreasing with $\alpha_0 \approx 1, \alpha_T \approx 0$, while σ_t is increasing.

SDE	$\boldsymbol{F}_t$	$\boldsymbol{G}_t$	μ_t	Σ_t
VPSDE [13]	$\frac{1}{2}\frac{d\log\alpha_t}{dt}\boldsymbol{I}$	$\sqrt{-\frac{d\log\alpha_t}{dt}}\boldsymbol{I}$	$\sqrt{\alpha_t}\boldsymbol{I}$	$(1-\alpha_t)\boldsymbol{I}$
VESDE [35]	0	$\sqrt{\frac{d[\sigma_t^2]}{dt}}\boldsymbol{I}$	$\boldsymbol{I}$	$\sigma_t^2\boldsymbol{I}$

where $\boldsymbol{F}_t \in \mathbb{R}^{D\times D}$ denotes the linear drift coefficient, $\boldsymbol{G}_t \in \mathbb{R}^{D\times D}$ denotes the diffusion coefficient, and $\boldsymbol{w}$ is a standard Wiener process. The diffusion Eq (1) is initiated at the training data and simulated over a fixed time window $[0, T]$. Denote by $p_t(\boldsymbol{x}_t)$ the marginal distribution of $\boldsymbol{x}_t$ and by $p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ the conditional distribution from $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$, then $p_0(\boldsymbol{x}_0)$ represents the underlying distribution of the training data. The simulated trajectories are represented by $\{\boldsymbol{x}_t\}_{0\le t\le T}$. The parameters $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are chosen such that the conditional marginal distribution $p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is a simple Gaussian distribution, denoted as $\mathcal{N}(\mu_t\boldsymbol{x}_0, \Sigma_t)$, and the distribution $\pi(\boldsymbol{x}_T) := p_T(\boldsymbol{x}_T)$ is easy to sample from. Two popular SDEs used by diffusion models [35] are summarized in Tab 1.

Backward denoising diffusion: Under mild assumptions [1, 35], the forward diffusion Eq (1) is associated with a reverse-time diffusion process

$$d\boldsymbol{x} = [\boldsymbol{F}_t\boldsymbol{x}dt - \boldsymbol{G}_t\boldsymbol{G}_t^T\nabla\log p_t(\boldsymbol{x})]dt + \boldsymbol{G}_t d\boldsymbol{w}, \tag{2}$$

where $\boldsymbol{w}$ denotes a standard Wiener process in the reverse-time direction. The distribution of the trajectories of Eq (2) with terminal distribution $\boldsymbol{x}_T \sim \pi$ coincides with that of Eq (1) with initial distribution $\boldsymbol{x}_0 \sim p_0$, that is, Eq (2) matches Eq (1) in probability law. Thus, in principle, we can generate new samples from the data distribution p_0 by simulating the backward diffusion Eq (2). However, to solve Eq (2), we need to evaluate the score function $\nabla\log p_t(\boldsymbol{x})$, which is not accessible.

Training: The basic idea of DMs is to use a time-dependent network $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$, known as a score network, to approximate the score $\nabla\log p_t(\boldsymbol{x})$. This is achieved by score matching techniques [16, 41] where the score network $\boldsymbol{s}_\theta$ is trained by minimizing the denoising score matching loss

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathrm{Unif}[0,T]}\mathbb{E}_{p(\boldsymbol{x}_0)p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)}[\|\nabla\log p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) - \boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\|^2_{\Lambda_t}]. \tag{3}$$

Here $\nabla\log p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ has a closed form expression as $p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is a simple Gaussian distribution, and Λ_t denotes a time-dependent weight. This loss can be evaluated using empirical samples by Monte Carlo methods and thus standard stochastic optimization algorithms can be used for training. We refer the reader to [13, 35] for more details on choices of Λ_t and training techniques.

3 Fast Sampling with learned score models

Once the score network $\boldsymbol{s}_\theta(\boldsymbol{x}, t) \approx \nabla\log p_t(\boldsymbol{x})$ is trained, it can be used to generate new samples by solving the backward SDE Eq (2) with $\nabla\log p_t(\boldsymbol{x})$ replaced by $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$. It turns out there are infinitely many diffusion processes one can use. In this work, we consider a family of SDEs

$$d\hat{\boldsymbol{x}} = [\boldsymbol{F}_t\hat{\boldsymbol{x}} - \frac{1+\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}, t)]dt + \lambda\boldsymbol{G}_t d\boldsymbol{w}, \tag{4}$$

parameterized by $\lambda \ge 0$. Here we use $\hat{\boldsymbol{x}}$ to distinguish the solution to the SDE associated with the learned score from the ground truth $\boldsymbol{x}$ in Eq (1) and (2). When $\lambda = 0$, Eq (4) reduces to an ODE known as the *probability flow ODE* [35]. The reverse-time diffusion Eq (2) with an approximated score is a special case of Eq (4) with $\lambda = 1$. Denote the trajectories generated by Eq (4) as $\{\hat{\boldsymbol{x}}_t^*\}_{0\le t\le T}$ and the marginal distributions as $\hat{p}_t^*$. The following Proposition [46] (Proof in Appendix B) holds.

Proposition 1. *When $\boldsymbol{s}_\theta(\boldsymbol{x}, t) = \nabla\log p_t(\boldsymbol{x})$ for all $\boldsymbol{x}, t$, and $\hat{p}_T^* = \pi$, the marginal distribution $\hat{p}_t^*$ of Eq (4) matches p_t of the forward diffusion Eq (1) for all $0 \le t \le T$.*

The above result justifies the usage of Eq (4) for generating samples. To generate a new sample, one can sample $\hat{\boldsymbol{x}}_T^*$ from π and solve Eq (4) to obtain a sample $\hat{\boldsymbol{x}}_0^*$. However, in practice, exact solutions to Eq (4) are not attainable and one needs to discretize Eq (4) over time to get an approximated solution. Denote the approximated solution by $\hat{\boldsymbol{x}}_t$ and its marginal distribution by $\hat{p}_t$, then the error of the generative model, that is, the difference between $p_0(\boldsymbol{x})$ and $\hat{p}_0(\boldsymbol{x})$, is caused by two error sources, *fitting error* and *discretization error*. The fitting error is due to the mismatch between the learned

score network $\boldsymbol{s}_\theta$ and the ground truth score $\nabla \log p_t(\boldsymbol{x})$. The discretization error includes all extra errors introduced by the discretization in numerically solving Eq (4). To reduce discretization error, one needs to use smaller stepsize and thus larger number of steps, making the sampling less efficient.

The objective of this work is to investigate these two error sources and develop a more efficient sampling scheme from Eq (4) with less errors. In this section, we focus on the ODE approach with $\lambda = 0$. All experiments in this section are conducted based on VPSDE over the CIFAR10 dataset unless stated otherwise. The discussions on SDE approach with $\lambda > 0$ are deferred to Section 5.

3.1 Can we learn globally accurate score?

Since DMs demonstrate impressive empirical results in generating high-fidelity samples, it is tempting to believe that the learned score network is able to fit the score of data distribution very well, that is, $\boldsymbol{s}_\theta(\boldsymbol{x}, t) \approx \nabla \log p_t(\boldsymbol{x})$ for almost all $\boldsymbol{x} \in \mathbb{R}^D$ and $t \in [0, T]$. This is, however, not true; the fitting error can be arbitrarily large on some $\boldsymbol{x}, t$ as illustrated in a simple example below. In fact, the learned score model are not accurate for most $\boldsymbol{x}, t$.

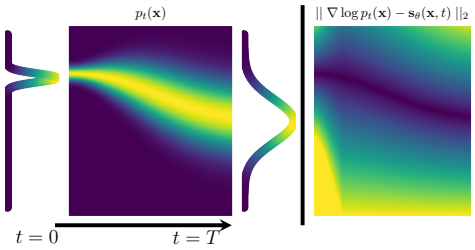


Figure 2: Fitting error on a toy demo. Lighter areas represent higher probability region (left) and larger fitting error (right).

Consider a generative modeling task over 1-dimensional space, i.e., $D = 1$. The data distribution is a Gaussian concentrated at 2 with very small variance. We plot the fitting error[2] between a score model trained by minimizing Eq (3) and the ground truth score in Fig 2. As can be seen from the figure, the score model works well in the region where $p_t(\boldsymbol{x})$ is large but suffers from large error in the region where $p_t(\boldsymbol{x})$ is small. This observation can be explained by examining the training loss Eq (3). In particular, the training data of Eq (3) are sampled from $p_t(\boldsymbol{x})$. In regions with low $p_t(\boldsymbol{x})$ value, the learned score network is not expected to work well due to the lack of training data. This phenomenon becomes even clearer in realistic settings with high dimensional data. The region with high $p_t(\boldsymbol{x})$ value is extremely small since realistic data is often sparsely distributed in $\mathbb{R}^D$; it is believed real data such as images concentrates on an intrinsic low dimensional manifold [8, 30, 25].

As a consequence, to ensure $\hat{\boldsymbol{x}}_0$ is close to $\boldsymbol{x}_0$, we need to make sure $\hat{\boldsymbol{x}}_t$ stays in the high $p_t(\boldsymbol{x})$ region for all t. This makes fast sampling from Eq (4) a challenging task as it prevents us from taking aggressive step size that is likely to take the solution to the region where the fitting error of learned score network is large. A good discretization scheme for Eq (4) should be able to help reduce the impact of the fitting error of the score network during sampling.

3.2 Discretization error

We next investigate the discretization error of solving the probability flow ODE ($\lambda = 0$)

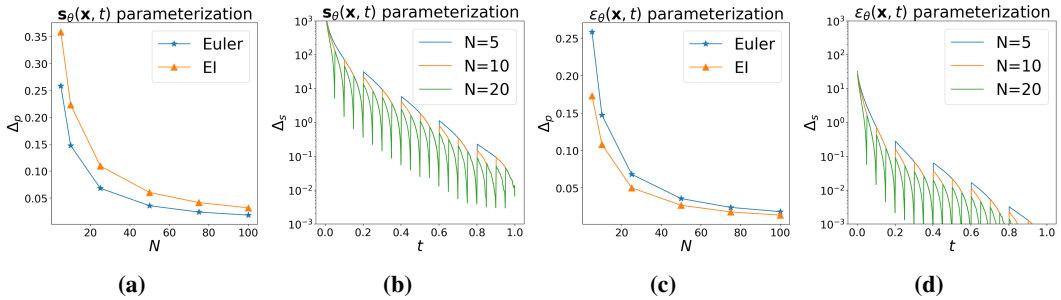
$$\frac{d\hat{\boldsymbol{x}}}{dt} = \boldsymbol{F}_t \hat{\boldsymbol{x}} - \frac{1}{2} \boldsymbol{G}_t \boldsymbol{G}_t^T \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}, t). \tag{5}$$

The exact solution to this ODE is

$$\hat{\boldsymbol{x}}_t = \Psi(t, s)\hat{\boldsymbol{x}}_s + \int_s^t \Psi(t, \tau)[-\frac{1}{2} \boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_\tau, \tau)] d\tau, \tag{6}$$

where $\Psi(t, s)$ satisfying $\frac{\partial}{\partial t}\Psi(t, s) = \boldsymbol{F}_t \Psi(t, s), \Psi(s, s) = \boldsymbol{I}$ is known as the transition matrix from time s to t associated with $\boldsymbol{F}_\tau$. Eq (5) is a semilinear stiff ODE [14] that consists of a linear term $\boldsymbol{F}_t \hat{\boldsymbol{x}}$ and a nonlinear term $\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}, t)$. There exist many different numerical solvers for Eq (5) associated with different discretization schemes to approximate Eq (6) [12]. As the discretization step size goes to zero, the solutions obtained from all these methods converge to that of Eq (5). However, the performances of these methods can be dramatically different when the step size is large. On the other hand, to achieve fast sampling with Eq (5), we need to approximately solve it with a small number of discretization steps, and thus large step size. This motivates us to develop an efficient discretizaiton scheme that fits with Eq (5) best. In the rest of this section, we systematically study the discretization error in solving Eq (5), both theoretically and empirically with carefully designed experiments. Based on these results, we develop an efficient algorithm for Eq (5) that requires a small number of NFEs.

[2]Because fitting error explodes when $t \to 0$, we have scaled the fitting error for better visualization.



(a) **(b)** **(c)** **(d)**

Figure 3: Fig 3a shows average pixel difference Δ_p between ground truth $\hat{\boldsymbol{x}}_0^*$ and numerical solution $\hat{\boldsymbol{x}}_0$ from Euler method and EI method. Fig 3b depicts approximation error Δ_p along ground truth solutions. Fig 3d shows Δ_s can be dramatically reduced if the parameterization $\epsilon_\theta(\boldsymbol{x}, t)$ instead of $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ is used. This parameterization help the EI method outperform the Euler method in Fig 3c.

Ingredient 1: Exponential Integrator over Euler method. The Euler method is the most elementary explicit numerical method for ODEs and is widely used in numerical softwares [42]. When applied to Eq (5), the Euler method reads

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \hat{\boldsymbol{x}}_t - [\boldsymbol{F}_t\hat{\boldsymbol{x}}_t - \frac{1}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t)]\Delta t. \tag{7}$$

This is used in many existing works in DMs [35, 10]. This approach however has low accuracy and is sometimes unstable when the stepsize is not sufficiently small. To improve the accuracy, we propose to use the *Exponential Integrator (EI)*, a method that leverages the semilinear structure of Eq (5). When applied to Eq (5), the EI reads

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \Psi(t-\Delta t, t)\hat{\boldsymbol{x}}_t + [\int_t^{t-\Delta t} -\frac{1}{2}\Psi(t-\Delta t, \tau)\boldsymbol{G}_\tau\boldsymbol{G}_\tau^T d\tau]\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t). \tag{8}$$

It is effective if the nonlinear term $\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t)$ does not change much along the solution. In fact, for any given Δt, Eq (8) solves Eq (5) exactly if $\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t)$ is constant over the time interval $[t-\Delta t, t]$.

To compare the EI Eq (8) and the Euler method Eq (7), we plot in Fig 3a the *average pixel difference* Δ_p between the ground truth $\hat{\boldsymbol{x}}_0^*$ and the numerical solution $\hat{\boldsymbol{x}}_0$ obtained by these two methods for various number N of steps. Surprisingly, the EI method performs worse than the Euler method.

This observation suggests that there are other major factors that contribute to the error Δ_p. In particular, the condition that the nonlinear term $\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t)$ does not change much along the solution assumed for the EI method does not hold. To see this, we plot the *score approximation error* $\Delta_{\boldsymbol{s}}(\tau) = ||\boldsymbol{s}_\theta(\boldsymbol{x}_\tau, \tau) - \boldsymbol{s}_\theta(\boldsymbol{x}_t, t)||_2, \tau \in [t-\Delta t, t]$ along the exact solution $\{\hat{\boldsymbol{x}}_t^*\}$ to Eq (5) in Fig 3b[3]. It can be seen that the approximation error grows rapidly as t approaches 0. This is not strange; the score of realistic data distribution $\nabla \log p_t(\boldsymbol{x})$ should change rapidly as $t \to 0$ [10].

Ingredient 2: $\epsilon_\theta(\boldsymbol{x}, t)$ over $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$. The issues caused by rapidly changing score $\nabla \log p_t(\boldsymbol{x})$ do not only exist in sampling, but also appear in the training of DMs. To address these issues, a different parameterization of the score network is used. In particular, it is found that the parameterization [13] $\nabla \log p_t(\boldsymbol{x}) \approx -\boldsymbol{L}_t^{-T}\epsilon_\theta(\boldsymbol{x}, t)$, where $\boldsymbol{L}_t$ can be any matrix satisfying $\boldsymbol{L}_t\boldsymbol{L}_t^T = \Sigma_t$, leads to significant improvements of accuracy. The rationale of this parameterization is based on a reformulation of the training loss Eq (3) as [13]

$$\bar{\mathcal{L}}(\theta) = \mathbb{E}_{t\sim\mathrm{Unif}[0,T]}\mathbb{E}_{p(\boldsymbol{x}_0),\epsilon\sim\mathcal{N}(0,\boldsymbol{I})}[\|\epsilon - \epsilon_\theta(\mu_t\boldsymbol{x}_0 + \boldsymbol{L}_t\epsilon, t)\|_{\bar{\Lambda}_t}^2] \tag{9}$$

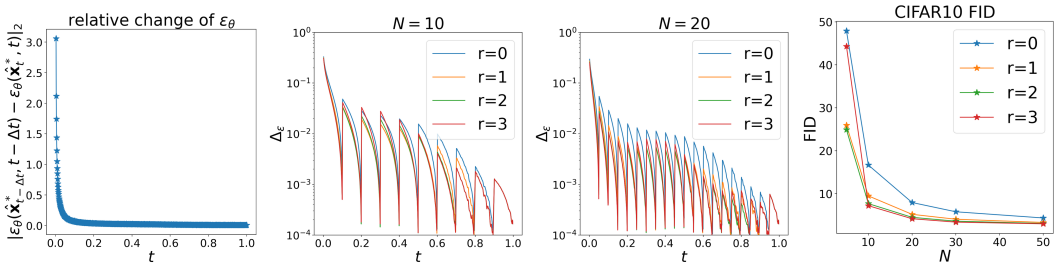
with $\bar{\Lambda}_t = \boldsymbol{L}_t^{-1}\Lambda_t\boldsymbol{L}_t^{-T}$. The network ϵ_θ tries to follow ϵ which is sampled from a standard Gaussian and thus has a small magnitude. In comparison, the parameterization $\boldsymbol{s}_\theta = -\boldsymbol{L}_t^{-T}\epsilon_\theta$ can take large value as $\boldsymbol{L}_t \to 0$ as t approaches 0. It is thus better to approximate ϵ_θ than $\boldsymbol{s}_\theta$ with a neural network.

We adopt this parameterization and rewrite Eq (5) as

$$\frac{d\hat{\boldsymbol{x}}}{dt} = \boldsymbol{F}_t\hat{\boldsymbol{x}} + \frac{1}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\boldsymbol{L}_t^{-T}\epsilon_\theta(\hat{\boldsymbol{x}}, t). \tag{10}$$

Applying the EI to Eq (10) yields

[3]For better visualization, we have removed the time discretization points in Fig 3b and Fig 3d, since $\Delta_s = 0$ at these points and becomes negative infinity in log scale.



(a) **(b)** **(c)** **(d)**

Figure 5: Fig 5a shows relative changes of $\epsilon_\theta(\hat{\boldsymbol{x}}_t^*, t)$ with respect to t are relatively small, especially when $t > 0.15$. Fig 5b and Fig 5c depict the extrapolation error with $N = 10, 20$ respectively. High order polynomial can reduce approximation error effectively. Fig 5d displays the sampling quality measured by FID of DEIS on CIFAR10 with a uniform time discretization. Note that the order 3 polynomial does not work well when $N = 5$; this issue is resolved by optimizing step size.

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \Psi(t-\Delta t, t)\hat{\boldsymbol{x}}_t + [\int_t^{t-\Delta t} \frac{1}{2}\Psi(t-\Delta t, \tau)\boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{L}_\tau^{-T} d\tau]\epsilon_\theta(\hat{\boldsymbol{x}}_t, t). \tag{11}$$

Compared with Eq (8), Eq (11) employs $-\boldsymbol{L}_\tau^{-T}\epsilon_\theta(\boldsymbol{x}_t, t)$ instead of $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) = -\boldsymbol{L}_t^{-T}\epsilon_\theta(\boldsymbol{x}_t, t)$ to approximate the score $\boldsymbol{s}_\theta(\boldsymbol{x}_\tau, \tau)$ over the time interval $\tau \in [t-\Delta t, t]$. This modification from $\boldsymbol{L}_t^{-T}$ to $\boldsymbol{L}_\tau^{-T}$ turns out to be crucial; the coefficient $\boldsymbol{L}_\tau^{-T}$ changes rapidly over time. This is verified by Fig 3d where we plot the score approximation error Δ_s when the parameterization ϵ_θ is used, from which we see the error Δ_s is greatly reduced compared with Fig 3b. With this modification, the EI method significantly outperforms the Euler method as shown in Fig 3c. Next we develop several fast sampling algorithms, all coined as the *Diffusion Exponential Integrator Sampler (DEIS)*, based on Eq (11), for DMs.

Interestingly, the discretization Eq (11) based on EI coincides with the popular deterministic DDIM when the forward diffusion Eq (1) is VPSDE [34] as summarized below (Proof in Appendix C).

Proposition 2. *When the forward diffusion Eq (1) is set to be VPSDE ($\boldsymbol{F}_t, \boldsymbol{G}_t$ are specified in Tab 1), the EI discretization Eq (11) becomes*

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\hat{\boldsymbol{x}}_t + [\sqrt{1-\alpha_{t-\Delta t}} - \sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\sqrt{1-\alpha_t}]\epsilon_\theta(\hat{\boldsymbol{x}}_t, t), \tag{12}$$

which coincides with the deterministic DDIM sampling algorithm.

Our result provides an alternative justification for the efficacy of DDIM for VPSDE from a numerical discretization point of view. Unlike DDIM, our method Eq (11) can be applied to any diffusion SDEs to improve the efficiency and accuracy of discretizations.

In the discretization Eq (11), we use $\epsilon_\theta(\hat{\boldsymbol{x}}_t, t)$ to approximate $\epsilon_\theta(\hat{\boldsymbol{x}}_\tau, \tau)$ for all $\tau \in [t-\Delta t, t]$, which is a zero order approximation. Comparing Eq (11) and Eq (6) we see that this approximation error largely determines he accuracy of discretization. One natural question to ask is whether it is possible to use a better approximation of $\epsilon_\theta(\hat{\boldsymbol{x}}_\tau, \tau)$ to further improve the accuracy? We answer this question affirmatively below with an improved algorithm.

Ingredient 3: Polynomial extrapolation of ϵ_θ.

Before presenting our algorithm, we investigate how $\epsilon_\theta(\boldsymbol{x}_t, t)$ evolves along a ground truth solution $\{\hat{\boldsymbol{x}}_t\}$ from $t = T$ to $t = 0$. We plot the relative change in 2-norm of $\epsilon_\theta(\boldsymbol{x}_t, t)$ in Fig 5a. It reveals that for most time instances the relative change is small. This motivates us to use previous (backward) evaluations of ϵ_θ up to t to extrapolate $\epsilon_\theta(\boldsymbol{x}_\tau, \tau)$ for $\tau \in [t-\Delta t, t]$.

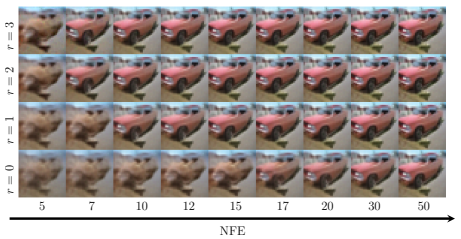


Figure 4: Effects of extrapolation. When N is small, higher order polynomial approximation leads to better samples.

Inspired by the high order polynomial extrapolation in linear multistep methods, we propose to use high order polynomial extrapolation of ϵ_θ in our EI method. To this end, consider time discretization $\{t_i\}_{i=0}^N$ where $t_0 = 0, t_N = T$. For each i, we fit a polynomial $\boldsymbol{P}_r(t)$ of degree r with respect

to the interpolation points $(t_{i+j}, \epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i+j}}, t_{i+j})), 0 \le j \le r$. This polynomial $\boldsymbol{P}_r(t)$ has explicit expression

$$\boldsymbol{P}_r(t) = \sum_{j=0}^{r} [\prod_{k \neq j} \frac{t - t_{i+k}}{t_{i+j} - t_{i+k}}] \epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i+j}}, t_{i+j}). \tag{13}$$

We then use $\boldsymbol{P}_r(t)$ to approximate $\epsilon_\theta(\boldsymbol{x}_\tau, \tau)$ over the interval $[t_{i-1}, t_i]$. For $i > N - r$, we need to use polynomials of lower order to approximate ϵ_θ. To see the advantages of this approximation, we plot the approximate error of $\epsilon_\theta(\boldsymbol{x}_t, t)$ by $\boldsymbol{P}_r(t)$ along ground truth trajectories $\{\hat{\boldsymbol{x}}_t^*\}$ in fig 5b and 5c. It can be seen that higher order polynomials can reduce approximation error compared with the case $r = 0$ which uses zero order approximation as in Eq (11).

As in the EI method Eq (11) that uses a zero order approximation of the score in Eq (6), the update step of order r is obtained by plugging the polynomial approximation Eq (13) into Eq (6). It can be written explicitly as

$$\hat{\boldsymbol{x}}_{t_{i-1}} = \Psi(t_{i-1}, t_i)\hat{\boldsymbol{x}}_{t_i} + \sum_{j=0}^{r} [\mathrm{C}_{ij} \epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i+j}}, t_{i+j})] \tag{14}$$

$$\mathrm{C}_{ij} = \int_{t_i}^{t_{i-1}} \frac{1}{2} \Psi(t_{i-1}, \tau) \boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{L}_\tau^{-T} \prod_{k \neq j} [\frac{\tau - t_{i+k}}{t_{i+j} - t_{i+k}}] d\tau. \tag{15}$$

We remark that the update in Eq (14) is a linear combination of $\hat{\boldsymbol{x}}_{t_i}$ and $\epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i+j}}, t_{i+j})$, where the weights $\Psi(t_{i-1}, t_i)$ and C_{ij} are calculated once for a given forward diffusion Eq (1) and time discretization, and can be reused across batches. For some diffusion Eq (1), $\Psi(t_{i-1}, t_i), \mathrm{C}_{ij}$ have closed form expression. Even if analytic formulas are not available, one can use high accuracy solver to obtain these coefficients. In DMs (e.g., VPSDE and VESDE), Eq (15) are normally 1-dimensional or 2-dimensional integrations and are thus easy to evaluate numerically. This approach resembles the classical Adams–Bashforth [14] method, thus we term it tAB-DEIS. Here we use t to differentiate it from other DEIS algorithms we present later in Section 4 based on a time-scaled ODE.

The tAB-DEIS algorithm is summarized in Alg 1. Note that the deterministic DDIM is a special case of tAB-DEIS for VPSDE with $r = 0$. The polynomial approximation used in DEIS improves the sampling quality significantly when the number of sampling steps N is small, as shown in Fig 5d and Fig 4.

Ingredient 4: Optimizing time discretization. From Fig 5 we observe that the approximation error is not uniformly distributed for all $0 \le t \le T$ when uniform discretization over time is used; the error increases as t approaches 0. This observation implies that, instead of uniform step size (linear timesteps), smaller step size should be used for t close to 0 to improve accuracy. One such option is the quadratic timestep suggested in [34] that follows $\mathrm{linspace}(0, \sqrt{T}, N+1)^2$.

To better understand the effects of time discretization, we investigate the difference between the ground truth $\hat{\boldsymbol{x}}_t^*$ and the numerical solution $\hat{\boldsymbol{x}}_t$ with the same boundary value $\hat{\boldsymbol{x}}_T^*$

$$\hat{\boldsymbol{x}}_t^* - \hat{\boldsymbol{x}}_t = \int_T^t \frac{1}{2} \Psi(t, \tau) \boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{L}_\tau^{-T} \Delta\epsilon_\theta(\tau) d\tau, \quad \Delta\epsilon_\theta(\tau) = \epsilon_\theta(\hat{\boldsymbol{x}}_\tau^*, \tau) - \boldsymbol{P}_r(\tau). \tag{16}$$

Eq (16) shows that the difference between the solutions $\hat{\boldsymbol{x}}_t^*$ and $\hat{\boldsymbol{x}}_t$ is a weighted sum of $\Delta\epsilon_\theta(\tau)$. We emphasis that Eq (16) does not only contain the approximation error of $\boldsymbol{P}_r(\tau)$ which we discussed before, but also accumulation error. Indeed, since $\boldsymbol{P}_r(\tau)$ is fitted on the solution $\{\hat{\boldsymbol{x}}_\tau\}$ instead of ground truth trajectory $\{\hat{\boldsymbol{x}}_\tau^*\}$, there exists accumulation error caused by past errors. A good choice of time discretization should balance the approximation error and the accumulation error.

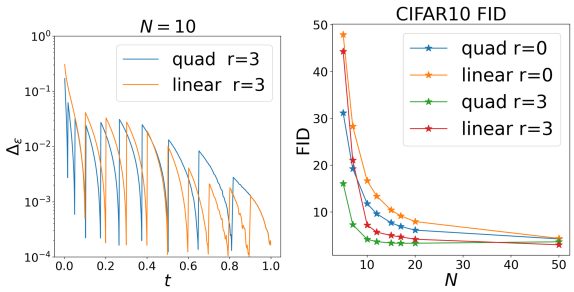
We illustrate this by comparing linear timesteps and quadratic timesteps. We plot the approximation error along $\{\hat{\boldsymbol{x}}_t^*\}$ in Fig 6a. Compared with linear timesteps, the quadratic scheduling has much lower approximation errors when $t \to 0$ but slightly larger error when $t \to T$. In our experiments, we find FIDs associated with quadratic timesteps are lower for small N and similar to linear scheduling for large N, see Fig 6b. This implies that the approximation error plays a dominant role in Eq (16) when N is small; a good choice of discretization should keep the approximation error small for all t.

4 Exponential Integrator: simplify probability Flow ODE

Next we present a different perspective to DEIS based on ODE transformations. The probability ODE Eq (10) can be transformed into a simple non-stiff ODE, and then off-the-shelf ODE solvers

Algorithm 1 tAB-DEIS

Input: $\{t_i\}_{i=0}^N, r$
Instantiate: $\hat{\boldsymbol{x}}_{t_N}$, Empty ϵ-buffer
Calculate weights $\Psi, \boldsymbol{C}$ based on Eq (15)
for i in $N, N-1, \cdots, 1$ **do**
  ϵ-buffer.append($\epsilon_\theta(\hat{\boldsymbol{x}}_{t_i}, t_i)$)
  $\hat{\boldsymbol{x}}_{t_{i-1}} \leftarrow$ run Eq (14) with $\Psi, \boldsymbol{C}, \epsilon$-buffer
end for



(a) **(b)**

Figure 6: Comparison between quadratic timesteps and linear timesteps. Fig 6a shows quadratic timesteps lead to low approximation error when $t \to 0$ but slightly worse approximation when $t \to T$. Fig 6b demonstrates that quadratic scheduling gives better sampling quality for small N.

can be applied to solve the ODE effectively. To this end, we introduce variable $\hat{\boldsymbol{y}}_t := \Psi(t,0)\hat{\boldsymbol{x}}_t$ and rewrite Eq (10) into

$$\frac{d\hat{\boldsymbol{y}}}{dt} = \frac{1}{2}\Psi(t,0)\boldsymbol{G}_t\boldsymbol{G}_t^T\boldsymbol{L}_t^{-T}\epsilon_\theta(\Psi(0,t)\hat{\boldsymbol{y}}, t). \tag{17}$$

Note that, departing from Eq (10), Eq (17) does not possess semi-linear structure. Thus, we can apply off-the-shelf ODE solvers to Eq (17) without worrying about the semi-linear structure. This transformation Eq (17) can be further improved by taking into account the analytical form of $\Psi, \boldsymbol{G}_t, \boldsymbol{L}_t$. Here we present a treatment for VPSDE; the results can be extended to other (scalar) DMs such as VESDE.

Proposition 3. *For the VPSDE, with* $\hat{\boldsymbol{y}}_t = \sqrt{\frac{\alpha_0}{\alpha_t}}\hat{\boldsymbol{x}}_t$ *and the time-scaling* $\beta(t) = \sqrt{\alpha_0}(\sqrt{\frac{1-\alpha_t}{\alpha_t}} - \sqrt{\frac{1-\alpha_0}{\alpha_0}})$, *Eq* (10) *can be transformed into*

$$\frac{d\hat{\boldsymbol{y}}}{d\rho} = \epsilon_\theta(\sqrt{\frac{\alpha_{\beta^{-1}(\rho)}}{\alpha_0}}\hat{\boldsymbol{y}}, \beta^{-1}(\rho)), \quad \rho \in [\beta(0), \beta(T)]. \tag{18}$$

After transformation, the ODE becomes a black-box ODE that can only be solved by generic ODE solvers. This is the core idea of the variants of DEIS we present next. Absorbing useful structures such as semi-linear structure through a transformation so that the transformed ODE does not posses any useful structure that one can utilize to improve generic ODE methods. This ensures that the structure does not induce any unnecessary discretization error.

Based on the transformed ODE Eq (18) and the above discussions, we propose two variants of the DEIS algorithm: ρRK-DEIS when applying classical RK methods, and ρAB-DEIS when applying Adams-Bashforth methods. We remark that the difference between tAB-DEIS and ρAB-DEIS lies in the fact that tAB-DEIS fits polynomials in t which may not be polynomials in ρ.

5 Discretization error of SDE sampling

Next we consider the problem of solving the SDE Eq (4) with $\lambda > 0$. As shown in Proposition 1, this would also lead to a sampling scheme from DMs. The exact solution to Eq (4) satisfies

$$\hat{\boldsymbol{x}}_t = \underbrace{\Psi(t,s)\hat{\boldsymbol{x}}_s}_{\text{Linear term}} + \int_s^t \overbrace{\underbrace{\Psi(t,\tau)\frac{1+\lambda^2}{2}\boldsymbol{G}_\tau\boldsymbol{G}_\tau^T\boldsymbol{L}_\tau^{-T}\epsilon_\theta(\hat{\boldsymbol{x}}_\tau,\tau)d\tau}_{\text{Nonlinear term}}}^{\text{Weight}} + \int_s^t \lambda \underbrace{\Psi(t,\tau)\boldsymbol{G}_\tau d\boldsymbol{w}}_{\text{Noise term}}, \tag{19}$$

where Ψ is as before. The goal is to approximate Eq (19) through discretization. Interestingly, the stochastic DDIM [34] turns out to be a numerical solver for Eq (19) as follows (Proof in Appendix D).

Proposition 4. *For the VPSDE, the stochastic DDIM is a discretization scheme of Eq* (19).

How do we discretize Eq (19) for a general SDE Eq (4)? One strategy is to follow what we did for the ODE ($\lambda = 0$) in Section 3.2 and approximate $\epsilon_\theta(\hat{\boldsymbol{x}}_\tau, \tau)$ by a polynomial. However, we found this

NFE	FID for various DEIS									
	DDIM	ρ2Heun	ρ3Kutta	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	26.91	108^{+1}	185^{+1}	193^{+3}	22.28	21.53	21.43	19.72	16.31	**15.37**
10	11.14	14.72	13.19^{+2}	28.65^{+2}	7.56	6.72	6.50	6.09	4.57	**4.17**
15	7.06	4.89^{+1}	5.88	6.88^{+1}	4.69	4.16	3.99	4.29	3.57	**3.37**
20	5.47	3.50	2.97^{+1}	3.92	3.70	3.32	3.17	3.54	3.05	**2.86**
50	3.27	2.60	$\mathbf{2.55}^{+1}$	2.57^{+2}	2.70	2.62	2.59	2.67	2.59	2.57

Table 2: More results of DEIS for VPSDE on CIFAR10 with limited NFE. For ρRK-DEIS, the upper right number indicates extra NFEs used. Bold numbers denote the best performance achieved with similar NFE budgets.

strategy does not work well in practice. We believe it is due to several possible reasons as follows. We do not pursue the discretization of the SDE Eq (4) further in this paper and leave it for future.

Nonlinear weight and discretization error. In Eq (19), the linear and noise terms can be calculated exactly without discretizaiton error. Thus, only the nonlinear term ϵ_θ can induce error in the EI method. Compared with Eq (11), Eq (19) has a larger weight for the nonlinearity term as $\lambda > 0$ and is therefore more likely to cause larger errors. From this perspective, the ODE with $\lambda = 0$ is the best option since it minimizes the weight of nonlinear term. In [34], the authors also observed that the deterministic DDIM outperforms stochastic DDIM. Such observation is consistent with our analysis. Besides, we notice that the nonlinear weight in VPSDE is significantly smaller than that in VESDE, which implies VPSDE has smaller discretization error. Indeed, empirically, VPSDE has much better sampling performance when N is small. **Additional noise**. Compared with Eq (11) for ODEs, Eq (19) injects additional noise to the state when it is simulated backward. Thus, to generate new samples by denoising, the score model needs not only to remove noise in $\hat{\boldsymbol{x}}_{t_N}$, but also to remove this injected noise. For this reason, a better approximation of ϵ_θ may be needed.

6 More experiments

We compare our DEIS algorithm with several fast diffusion model sampling algorithms on image generation tasks. We show quantitative comparisons on CIFAR10 [24], CELEBA [26], ImageNet 32×32 [39] datasets. Since we focus on fast sampling with small NFE (less than 50 or even 20), we select competitive training-free algorithms, DDIM [34], A-DDIM [3] and PNDM [25] as baselines. We further propose Improved PNDM (iPNDM) that uses lower order multistep methods for the first few steps in sampling to avoid expensive warming start. This modification leads to better performance when we only have a limited NFE budget, such as 10. More implementation details, performance of various DM, and many more qualitative experiments are included in Appendix E.

We include performance evaluations of various DEIS with VPSDE on CIFAR10 in Tab 2, including DDIM, ρRK-DEIS, ρAB-DEIS and tAB-DEIS. For ρRK-DEIS, we chose second order Heun's method denoted as ρ2Heun, third order Heun's method denoted as ρ3Heun, and classic fourth-order RK denoted as ρ4RK. For Adam-Bashforth methods, we consider fitting $1, 2, 3$ order polynomial in t, ρ, denoted as tAB and ρAB respectively. We observe that almost all DEIS algorithms can generate high-fidelity images with a very small NFE budget. Also note that DEIS with high order polynomial approximation can significantly outperform DDIM; the latter coincides with the zero order polynomial approximation. We also find the performance of high order ρRK-DEIS is not satisfying when NFE is small but competitive as NFE increases. It is within expectation as high order methods enjoy smaller local truncation error and total accumulated error when small step size is used and the advantage is vanishing as we reduce number of steps (increase step size).

7 Conclusion and Broader Impact

In this work, we consider fast sampling problems for DMs. We present the diffusion exponential integrator sampler (DEIS), a fast sampling algorithm for DMs based on a novel discretization scheme of the backward diffusion process. In additional to its theoretical elegance, DEIS also works amazingly in practice; it is able to generate high-fidelity samples with less than 10 NFEs. There are several potential ways to further improve DEIS: 1) Use better approximation for $\epsilon_\theta(\boldsymbol{x}, t)$ than high

order polynomial methods; 2) DEIS with adaptive step size; 3) Develop better discretization schemes for $\epsilon_\theta(\boldsymbol{x}, t)$ for sampling with SDEs. How to effectively remove injected noise from the backward denoising diffusion is an open problem.

DMs have been proved to be effective on various generative tasks such as creation in art and photography. Our algorithm can substantially reduce the computational cost for these applications. Like many other algorithms for generative modeling, our method suffers from malicious usage, such as generating fake data and biased information.

References

[1] Anderson, B. D. Reverse-time diffusion equation models. Stochastic Process. Appl., 12 (3):313–326, May 1982. ISSN 0304-4149. doi: 10.1016/0304-4149(82)90051-5. URL https://doi.org/10.1016/0304-4149(82)90051-5.

[2] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. Advances in Neural Information Processing Systems, 34:17981–17993, 2021.

[3] Bao, F., Li, C., Zhu, J., and Zhang, B. Analytic-DPM: An Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models. 2022. URL http://arxiv.org/abs/2201.06503.

[4] Chen, T., Liu, G.-H., and Theodorou, E. A. Likelihood training of schrödinger bridge using forward-backward sdes theory. arXiv preprint arXiv:2110.11291, 2021.

[5] Chen, Y., Georgiou, T. T., and Pavon, M. Stochastic control liaisons: Richard sinkhorn meets gaspard monge on a schrodinger bridge. SIAM Review, 63(2):249–313, 2021.

[6] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. Generative adversarial networks: An overview. IEEE Signal Processing Magazine, 35(1):53–65, 2018.

[7] De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. Diffusion schrödinger bridge with applications to score-based generative modeling. Advances in Neural Information Processing Systems, 34, 2021.

[8] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee, 2009.

[9] Dhariwal, P. and Nichol, A. Diffusion Models Beat GANs on Image Synthesis. 2021. URL http://arxiv.org/abs/2105.05233.

[10] Dockhorn, T., Vahdat, A., and Kreis, K. Score-Based Generative Modeling with Critically-Damped Langevin Diffusion. pp. 1–13, 2021. URL http://arxiv.org/abs/2112.07068.

[11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

[12] Griffiths, D. F. and Higham, D. J. Numerical methods for ordinary differential equations: initial value problems, volume 5. Springer, 2010.

[13] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Advances in Neural Information Processing Systems, volume 2020-Decem, 2020. ISBN 2006.11239v2. URL https://github.com/hojonathanho/diffusion.

[14] Hochbruck, M. and Ostermann, A. Exponential integrators. Acta Numerica, 19:209–286, 2010.

[15] Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. Advances in Neural Information Processing Systems, 34, 2021.

[16] Hyvärinen, A. Estimation of non-normalized statistical models by score matching. J. Mach. Learn. Res., 6:695–708, 2005. ISSN 15337928.

[17] Jolicoeur-Martineau, A., Li, K., Piché-Taillefer, R., Kachman, T., and Mitliagkas, I. Gotta go fast when generating data with score-based models. arXiv preprint arXiv:2105.14080, 2021.

[18] Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4401–4410, 2019.

[19] Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. arXiv preprint arXiv:2206.00364, 2022.

[20] Kawar, B., Vaksman, G., and Elad, M. Snips: Solving noisy inverse problems stochastically. Advances in Neural Information Processing Systems, 34, 2021.

[21] Kim, D., Shin, S., Song, K., Kang, W., and Moon, I.-C. Score matching model for unbounded data score. arXiv preprint arXiv:2106.05527, 2021.

[22] Kingma, D. P. and Welling, M. An introduction to variational autoencoders. arXiv preprint arXiv:1906.02691, 2019.

[23] Kong, Z. and Ping, W. On fast sampling of diffusion probabilistic models. arXiv preprint arXiv:2106.00132, 2021.

[24] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

[25] Liu, L., Ren, Y., Lin, Z., and Zhao, Z. Pseudo Numerical Methods for Diffusion Models on Manifolds. (2021):1–23, 2022. URL http://arxiv.org/abs/2202.09778.

[26] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[27] Lyu, Z., Kong, Z., Xu, X., Pan, L., and Lin, D. A conditional point diffusion-refinement paradigm for 3d point cloud completion. arXiv preprint arXiv:2112.03530, 2021.

[28] Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models. ArXiv, abs/2102.09672, 2021.

[29] Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. 2021. URL http://arxiv.org/abs/2112.10741.

[30] Pless, R. and Souvenir, R. A survey of manifold learning for images. IPSJ Transactions on Computer Vision and Applications, 1:83–94, 2009.

[31] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

[32] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-Resolution Image Synthesis with Latent Diffusion Models. 2021. URL http://arxiv.org/abs/2112.10752.

[33] Särkkä, S. and Solin, A. Applied stochastic differential equations, volume 10. Cambridge University Press, 2019.

[34] Song, J., Meng, C., and Ermon, S. Denoising Diffusion Implicit Models. 2020. URL http://arxiv.org/abs/2010.02502.

[35] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-Based Generative Modeling through Stochastic Differential Equations. ArXiv preprint, abs/2011.13456, 2020. URL https://arxiv.org/abs/2011.13456.

[36] Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. Advances in Neural Information Processing Systems, 34, 2021.

[37] Song, Y., Shen, L., Xing, L., and Ermon, S. Solving inverse problems in medical imaging with score-based generative models. arXiv preprint arXiv:2111.08005, 2021.

[38] Tachibana, H., Go, M., Inahara, M., Katayama, Y., and Watanabe, Y. Itô-taylor sampling scheme for denoising diffusion probabilistic models using ideal derivatives. arXiv preprint arXiv:2112.13339, 2021.

[39] Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In International conference on machine learning, pp. 1747–1756. PMLR, 2016.

[40] Vargas, F., Thodoroff, P., Lamacraft, A., and Lawrence, N. Solving Schrödinger bridges via maximum likelihood. Entropy, 23(9):1134, 2021.

[41] Vincent, P. A connection between score matching and denoising autoencoders. Neural Comput., 23(7):1661–1674, July 2011. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco_a_00142. URL https://doi.org/10.1162/neco_a_00142.

[42] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[43] Wang, G., Jiao, Y., Xu, Q., Wang, Y., and Yang, C. Deep generative learning via schrödinger bridge. In International Conference on Machine Learning, pp. 10794–10804. PMLR, 2021.

[44] Watson, D., Ho, J., Norouzi, M., and Chan, W. Learning to efficiently sample from diffusion probabilistic models. arXiv preprint arXiv:2106.03802, 2021.

[45] Whalen, P., Brio, M., and Moloney, J. Exponential time-differencing with embedded Runge–Kutta adaptive step control. J. Comput. Phys., 280:579–601, January 2015. ISSN 0021-9991. doi: 10.1016/j.jcp.2014.09.038. URL https://doi.org/10.1016/j.jcp.2014.09.038.

[46] Zhang, Q. and Chen, Y. Diffusion normalizing flow. Advances in Neural Information Processing Systems, 34, 2021.

A More related works

A lot of research has been conducted to speed up the sampling of DMs. In [23, 44] the authors optimize denosing process by modifying the underlying stochastic process. However, such acceleration can not generate high quality samples with a small number of discretization steps. In [34] the authors use a non-Markovian forward noising. The resulted algorihtm, DDIM, achieves significant acceleration than DDPMs. More recently, the authors of [3] optimize the backward Markovian process to approximate the non-Markovian forward process and get an analytic expression of optimal variance in denoising process. Another strategy to make the forward diffusion nonlinear and trainable [46, 40, 7, 43, 4] in the spirit of Schrödinger bridge [5]. This however comes with a heavy training overhead.

More closely related to our method is [25], which interprets update step in stochastic DDIM as a combination of gradient estimation step and transfer step. It modifies high order ODE methods to provide an estimation of the gradient and uses DDIM for transfer step. However, the decomposition of DDIM into two separate components is not theoretically justified. Based on our analysis on Exponential Integrator, Liu et al. [25] uses Exponential Integral but with a Euler discretization-based approximation of the nonlinear term. This approximation is inaccurate and may suffer large discretization error if the step size is large as we show in Section 6.

The semilinear structure presented in probability flow ODE has been widely investigated in physics and numerical simulation [14, 45], from which we get inspirations. The stiff property of the ODEs requires more efficient ODE solvers instead of black-box solvers that are designed for general ODE problems. In this work, we investigate sovlers for differential equations in diffusion model and take advantage of the semilinear structure.

B Proof of Proposition 1

The proof is inspired by [46]. We show that the marginal distribution induced by Eq (4) does not depend on the choice of λ and equals the marginal distribution induced by Eq (2) when the score model is perfect.

Consider the distribution q induced by the SDE

$$d\boldsymbol{x} = [\boldsymbol{F}_t\boldsymbol{x} - \frac{1+\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla \log q_t(\boldsymbol{x})]dt + \lambda \boldsymbol{G}_t d\boldsymbol{w}. \tag{20}$$

Eq (20) is simulated from $t = T$ to $t = 0$. According to the Fokker-Planck-Kolmogorov (FPK) Equation, q solves the partial differential equation

$$\begin{aligned}\frac{\partial q_t(\boldsymbol{x})}{\partial t} =& -\nabla\cdot\{[\boldsymbol{F}_t\boldsymbol{x} - \frac{1+\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla\log q_t(\boldsymbol{x})]q_t(\boldsymbol{x})\} - \frac{\lambda^2}{2}\langle \boldsymbol{G}_t\boldsymbol{G}_t^T, \frac{\partial^2}{\partial x_i\partial x_j}q_t(\boldsymbol{x})\rangle \\ =& -\nabla\cdot\{[\boldsymbol{F}_t\boldsymbol{x} - \frac{1}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla\log q_t(\boldsymbol{x})]q_t(\boldsymbol{x})\} + \nabla\cdot\{[\frac{\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla\log q_t(\boldsymbol{x})]q_t(\boldsymbol{x})\} - \\ & \frac{\lambda^2}{2}\langle \boldsymbol{G}_t\boldsymbol{G}_t^T, \frac{\partial^2}{\partial x_i\partial x_j}q_t(\boldsymbol{x})\rangle,\end{aligned}$$

where $\nabla\cdot$ denotes the divergence operator. Since

$$\nabla\cdot\{[\frac{\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla\log q_t(\boldsymbol{x})]q_t(\boldsymbol{x})\} = \nabla\cdot[\frac{\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla q_t(\boldsymbol{x})] = \langle\frac{\lambda^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T, \frac{\partial^2}{\partial x_i\partial x_j}q_t(\boldsymbol{x})\rangle, \tag{21}$$

we obtain

$$\frac{\partial q_t(\boldsymbol{x})}{\partial t} = -\nabla\cdot\{[\boldsymbol{F}_t\boldsymbol{x} - \frac{1}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\nabla\log q_t(\boldsymbol{x})]q_t(\boldsymbol{x})\}. \tag{22}$$

Eq (22) shows that the above partial differential equation does not depend on λ. Thus, the marginal distribution of Eq (20) is independent of the value of λ.

C Proof of Proposition 2

Thanks to , A straightforward calculation based on Eq (6) gives that $\Psi(t,s)$ for the VPSDE is

$$\Psi(t,s) = \sqrt{\frac{\alpha_t}{\alpha_s}}.$$

It follows that

$$\begin{aligned}\int_s^t \Psi(t,\tau)\frac{1}{2}\boldsymbol{G}_\tau\boldsymbol{G}_\tau^T\boldsymbol{L}_\tau^{-1}d\tau =& \int_s^t -\frac{1}{2}\sqrt{\frac{\alpha_t}{\alpha_\tau}}\frac{d\log\alpha_\tau}{d\tau}\frac{1}{\sqrt{1-\alpha_\tau}}d\tau\\ =&\sqrt{\alpha_t}\int_s^t -\frac{1}{2}\frac{d\alpha_\tau}{\alpha_\tau^{1.5}(1-\alpha_\tau)^{0.5}}\\ =&\sqrt{\alpha_t}\left.\sqrt{\frac{1-\tau}{\tau}}\right|_{\alpha_s}^{\alpha_t}\\ =&\sqrt{1-\alpha_t}-\sqrt{\frac{\alpha_t}{\alpha_s}}\sqrt{1-\alpha_s}.\end{aligned}$$

Setting $t \leftarrow t-\Delta t, s \leftarrow t$, we write Eq (11) as

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\hat{\boldsymbol{x}}_t + \left[\sqrt{1-\alpha_{t-\Delta t}} - \sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\sqrt{1-\alpha_t}\right]\epsilon_\theta(\hat{\boldsymbol{x}}_t, t).$$

D Proof of Proposition 4

Our derivation uses the notations in [34]. The DDIM employs the update step

$$\begin{aligned}\boldsymbol{x}_{t-\Delta t} = \sqrt{\alpha_{t-\Delta t}}\left(\frac{\boldsymbol{x}_t-\sqrt{1-\alpha_t}\epsilon_\theta(\boldsymbol{x}_t,t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-\Delta t}-\eta^2\frac{1-\alpha_{t-\Delta t}}{1-\alpha_t}\left(1-\frac{\alpha_t}{\alpha_{t-\Delta t}}\right)}\epsilon_\theta(\boldsymbol{x}_t,t)\\ + \eta\sqrt{\frac{1-\alpha_{t-\Delta t}}{1-\alpha_t}\left(1-\frac{\alpha_t}{\alpha_{t-\Delta t}}\right)}\epsilon_t,\end{aligned} \tag{23}$$

where η is a hyperparameter and $\eta \in [0,1]$. When $\eta = 0$, Eq (23) becomes determinstic and reduces to Eq (12). We show that Eq (23) is equivalent to Eq (4) when $\eta = \lambda$ and $\Delta t \to 0$.

By Eq (23), $\boldsymbol{x}_{t-\Delta t} \sim \mathcal{N}(\mu_\eta, \sigma_\eta^2\boldsymbol{I})$, where

$$\begin{aligned}\mu_\eta &= \sqrt{\alpha_{t-\Delta t}}\left(\frac{\boldsymbol{x}_t-\sqrt{1-\alpha_t}\epsilon_\theta(\boldsymbol{x}_t,t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-\Delta t}-\eta^2\frac{1-\alpha_{t-\Delta t}}{1-\alpha_t}\left(1-\frac{\alpha_t}{\alpha_{t-\Delta t}}\right)}\epsilon_\theta(\boldsymbol{x}_t,t)\\ \sigma_\eta^2 &= \eta^2\frac{1-\alpha_{t-\Delta t}}{1-\alpha_t}\left(1-\frac{\alpha_t}{\alpha_{t-\Delta t}}\right).\end{aligned}$$

It follows that

$$\begin{aligned}\lim_{\Delta t\to 0}\frac{\boldsymbol{x}_t-\mu_\eta}{t-(t-\Delta t)} =& \frac{\left(1-\sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\right)}{\Delta t}\boldsymbol{x}_t\\ &+\frac{\sqrt{\alpha_{t-\Delta t}}\sqrt{\frac{1-\alpha_t}{\alpha_t}} - \sqrt{1-\alpha_{t-\Delta t}-\eta^2\frac{1-\alpha_{t-\Delta t}}{1-\alpha_t}\left(1-\frac{\alpha_t}{\alpha_{t-\Delta t}}\right)}}{\Delta t}\epsilon_\theta(\boldsymbol{x}_t,t)\\ =&\frac{1}{2}\frac{d\log\alpha_t}{dt}\boldsymbol{x} + \frac{1+\eta^2}{2}\frac{d\log\alpha_t}{dt}\frac{1}{\sqrt{1-\alpha_t}}\epsilon_\theta(\boldsymbol{x}_t,t)\\ =&\boldsymbol{F}_t\boldsymbol{x} + \frac{1+\eta^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\boldsymbol{L}_t^{-T}\epsilon_\theta(\boldsymbol{x}_t,t),\end{aligned}$$

and

$$\lim_{\Delta t\to 0}\frac{\eta^2\frac{1-\alpha_{t-\Delta t}}{1-\alpha_t}\left(1-\frac{\alpha_t}{\alpha_{t-\Delta t}}\right)}{dt} = -\eta^2\frac{d\log\alpha_t}{dt} = \eta^2\boldsymbol{G}_t\boldsymbol{G}_t^T.$$

Consequently, the continuous limit of Eq (23) is

$$d\boldsymbol{x} = \left[\boldsymbol{F}_t\boldsymbol{x} + \frac{1+\eta^2}{2}\boldsymbol{G}_t\boldsymbol{G}_t^T\boldsymbol{L}_t^{-T}\epsilon_\theta(\boldsymbol{x},t)\right]dt + \eta\boldsymbol{G}_t d\boldsymbol{w}, \tag{24}$$

which is exactly Eq (4) if $\eta = \lambda$.

E More experiment details

E.1 Important technical details and modifications

- It is found that correcting steps and an extra denoising step can improve image quality at additional NFE costs [35, 17]. For a fair comparison, we disable the correcting steps, extra denoising step or other heuristic clipping tricks for all methods and experiments in this work unless stated otherwise.
- Due to numerical issues, we set ending time t_0 in DMs during sampling a non-zero number. Song et al. [35] suggests $t_0 = 10^{-3}$ for VPSDE and $t_0 = 10^{-5}$ for VESDE. In practice, we find the value of t_0 and time scheduling have huge impacts on FIDs. This finding is not new and has been pointed out by existing works [17, 21, 34]. Interestingly, we found different algorithms have different preference for t_0 and time scheduling. We report best FIDs for each methods among different choices of t_0 and time scheduling in Tab 2. We use t_0 suggested by original paper and codebase for different checkpoints and quadratic time scheduling suggested by Song et al. [34] unless stated otherwise. We include a comprehensive study about t_0 and time scheduling in Appendix E.3
- Because PNDM needs 12 NFE for the first 3 steps, we compare PNDM only when NFE is great than 12. However, our proposed iPNDM can work when NFE is less than 12.
- We include the comparison against A-DDIM [3] with its official checkpoints and implementation in Appendix E.4.
- We only provide qualitative results for text-to-image experiment with pre-trained model [31].
- We include proposed r-th order iPNDM in Appendix E.2. We use $r = 3$ by default unless stated otherwise.

E.2 Improved PNDM

By Eq (11), PNDM can be viewed as a combination of Exponential Integrator and linear multistep method based on Euler method. More specifically, it uses a linear combination of multiple score evaluations instead of using only the latest score evaluation. PNDM follows the steps

$$\hat{\epsilon}_t^{(3)} = \frac{1}{24}(55\epsilon_t - 59\epsilon_{t+\Delta t} + 37\epsilon_{t+2\Delta t} - 9\epsilon_{t+3\Delta t}), \tag{25}$$

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\hat{\boldsymbol{x}}_t + [\sqrt{1-\alpha_{t-\Delta t}} - \sqrt{\frac{\alpha_{t-\Delta t}}{\alpha_t}}\sqrt{1-\alpha_t}]\hat{\epsilon}_t^{(4)}, \tag{26}$$

where $\epsilon_t = \epsilon_\theta(\hat{\boldsymbol{x}}_t, t), \epsilon_{t+\Delta t} = \epsilon_\theta(\hat{\boldsymbol{x}}_{t+\Delta t}, t+\Delta t)$. The coefficients in Eq (25) are derived based on black-box ODE Euler discretization with fixed step size. Similarly, there exist lower order approximations

$$\hat{\epsilon}_t^{(0)} = \epsilon_t \tag{27}$$

$$\hat{\epsilon}_t^{(1)} = \frac{3}{2}\epsilon_t - \frac{1}{2}\epsilon_{t+\Delta t} \tag{28}$$

$$\hat{\epsilon}_t^{(2)} = \frac{1}{12}(23\epsilon_t - 16\epsilon_{t+\Delta t} + 5\epsilon_{t+2\Delta t}). \tag{29}$$

Originally, PNDM uses Runge-Kutta for warming start and costs 4 score network evaluation for each of the first 3 steps. To reduce the NFE in sampling, the improved PNDM (iPNDM) uses lower order multistep for warming start. We summarize iPNDM in Alg 2.

E.3 Impact of t_0 and time scheduling on FIDs

We present a study about sampling with difference t_0 and time scheduling based VPSDE. We consider two choices of t_0 $(10^{-3}, 10^{-4})$ and three choices for time scheduling. The first time scheduling follows power function in t

$$t_i = (\frac{N-i}{N}t_0^{\frac{1}{\kappa}} + \frac{i}{N}t_N^{\frac{1}{\kappa}})^\kappa, \tag{30}$$

Algorithm 2 Improved PNDM (iPNDM)

Input: $\{t_i\}_{i=0}^N, t_i = i\Delta t$, order r
Instantiate: $\boldsymbol{x}_{t_N}$, Empty ϵ-buffer
for i in $N, N-1, \cdots, 1$ **do**
  $j = \min(N-i+1, r)$
  ϵ-buffer.append($\epsilon_\theta(\hat{\boldsymbol{x}}_{t_i}, t_i)$)
  Simulate $\hat{\epsilon}_{t_i}^{(j)}$ based on j and ϵ-buffer
  $\hat{\boldsymbol{x}}_{t_{i-1}} \leftarrow$ Simulate Eq (26) with $\hat{\boldsymbol{x}}_{t_i}$ and $\hat{\epsilon}_{t_i}^{(j)}$
end for

the second time scheduling follows power function in ρ

$$\rho_i = (\frac{N-i}{N}\rho_0^{\frac{1}{\kappa}} + \frac{i}{N}\rho_N^{\frac{1}{\kappa}})^\kappa, \tag{31}$$

and the last time scheduling follows uniform step in $\log \rho$ space

$$\log \rho_i = \frac{N-i}{N}\log \rho_0 + \frac{i}{N}\log \rho_N. \tag{32}$$

We include the comparison between different t_0 and time scheduling in Tab 3 to 5. We notice t_0 has a huge influence on image FIDs, which is also noticed and investigated across different studies [21, 10]. Among various scheduling, we observe tAB-DEIS has obvious advantages when NFE is small and ρRK-DEIS is competitive when we NFE is relatively large.

E.4 Comparison with Analytic-DDIM (A-DDIM) [3]

We also compare our algorithm with Analytic-DDIM (A-DDIM) in terms of fast sampling performance. We failed to reproduce the significant improvements claimed in [3] in our default CIFAR10 checkpoint. There could be two factors that contribute to this. First, we use a score network trained with continuous time loss objective and different weights [35]. However, Analytic-DDIM is proposed for DDPM with discrete times and finite timestamps. Second, some tricks have huge impacts on the sampling quality in A-DDIM. For instance, A-DDIM heavily depends on clipping value in the last few steps [3]. A-DDIM does not provide high-quality samples without proper clipping when NFE is low.

To compare with A-DDIM, we conduct another experiment with checkpoints provided by [3] and integrate iPNDM and DEIS into the provided codebase; the results are shown in Tab 6. We use piecewise linear function to fit discrete SDE coefficients in [3] for DEIS. Without any ad-hoc tricks, the plugin-and-play iPNDM is comparable or even slightly better than A-DDIM when the NFE budget is small, and DEIS is better than both of them.

E.5 Sampling quality on ImageNet 32×32

We conduct experiments on ImageNet 32×32 with pre-trained VPSDE model provided in [36]. Again, we observe significant improvement over DDIM and iPNDM methods when the NFE budget is low. Even with 50 NFE, DEIS is able to outperform blackbox ODE solver in terms of sampling quality.

E.6 More results on VPSDE

We report the performance of the DOPRI5 ODE solver for VPSDE on CIFAR10 in Tab 8 [4]. As a popular and well-developed ODE solver, DOPRI5 has decent sampling performance when NFE ≥ 50. However, the sampling quality with limited NFE is not satisfying. Such results are within expectation as DOPRI5 does not take advantage of the structure information of diffusion models. The overall performance of DOPRI5 solver is worse than iPNDM and DEIS when NFE is small.

We include mean and standard deviation for CELEBA in Tab 9.

[4] We use $scipy.integrate.solve\_ivp$ and tune tolerance to get different performance on different NFE.

NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
	FID for various DEIS with $\kappa = 1$ in Eq (30)									
5	47.59	207^{+1}	238^{+1}	212^{+3}	35.14	32.51	32.02	25.99	**25.06**	44.29
10	16.60	84.55	66.81^{+2}	78.57^{+2}	10.47	8.85	8.18	9.51	7.71	**7.18**
15	10.39	46.36^{+1}	47.45	41.27^{+1}	6.69	5.70	5.24	6.47	5.51	**5.01**
20	7.93	34.87	28.35^{+1}	27.21	5.27	4.56	4.24	5.20	4.50	**4.14**
50	4.36	11.58	7.00^{+1}	7.48^{+2}	3.32	3.08	**2.99**	3.32	3.09	**2.99**
	FID for various DEIS with $\kappa = 2$ in Eq (30)									
NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	30.64	256^{+1}	357^{+1}	342^{+3}	24.58	23.60	23.48	20.01	16.52	**16.10**
10	11.71	56.62	56.51^{+2}	103^{+2}	7.56	6.72	6.50	6.09	4.57	**4.17**
15	7.67	10.62^{+1}	14.96	36.15^{+1}	4.93	4.40	4.26	4.29	3.57	**3.37**
20	6.11	6.33	4.74^{+1}	12.81	4.16	3.84	3.77	3.81	3.41	**3.33**
50	4.24	3.88	3.75^{+1}	3.78^{+2}	3.70	3.68	3.69	3.62	**3.61**	3.36
	FID for various DEIS with $\kappa = 3$ in Eq (30)									
NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	34.07	356^{+1}	388^{+1}	377^{+3}	29.80	29.35	29.38	24.87	22.57	**22.06**
10	14.59	115	171^{+2}	267^{+2}	10.73	10.16	10.11	8.11	6.36	**5.97**
15	9.22	32.94^{+1}	77.44	103^{+1}	6.45	6.03	5.98	5.21	4.26	**4.05**
20	7.27	13.06	11.55^{+1}	50.56	5.17	4.83	4.78	4.45	3.88	**3.75**
50	4.64	3.76	$\mathbf{3.68}^{+1}$	3.74^{+2}	3.92	3.82	3.79	3.81	3.72	3.71
	FID for various DEIS with Eq (32)									
NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	54.58	216^{+1}	335^{+1}	313^{+3}	49.25	48.56	48.47	37.99	28.45	**26.11**
10	20.03	14.72	13.19^{+2}	28.65^{+2}	14.05	12.63	12.18	10.36	7.03	**5.71**
15	11.99	5.03^{+1}	5.88	6.88^{+1}	7.72	6.67	6.29	6.22	4.69	**4.13**
20	8.92	4.12	3.97^{+1}	4.14	5.79	5.05	4.78	4.97	4.10	**3.80**
50	5.05	**3.67**	3.75^{+1}	3.73^{+2}	4.01	3.84	3.79	3.89	3.74	3.72

Table 3: DEIS for VPSDE on CIFAR10 with $t_0 = 10^{-3}$.

E.7 More reuslts on VESDE

Though VESDE does not achieve the same accelerations as VPSDE, our method can significantly accelerate VESDE sampling compared with previous method for VESDE. We show the accelerated FID for VESDE on CIFAR10 in Tab 10 and sampled images in Fig 7.

E.8 Checkpoint used and code licenses

Our code will be released in the future. We implemented our approach in Jax and PyTorch. We have also used code from a number of sources in Tab 11.

We list the used checkpoints and the corresponding experiments in Tab 12.

F More results for image generation

NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
	FID for various DEIS with $\kappa = 1$ in Eq (30)									
5	42.38	239^{+1}	232^{+1}	199^{+3}	33.09	31.06	30.67	26.01	**20.57**	42.35
10	17.23	143	95.13^{+2}	130^{+2}	12.44	11.04	10.41	12.01	10.57	**8.04**
15	12.06	99.76^{+1}	77.37	88.56^{+1}	9.12	8.25	7.79	9.08	8.20	**7.50**
20	9.71	82.89	57.54^{+1}	66.61	7.57	6.89	6.50	7.60	6.90	**6.45**
50	5.76	31.56	13.10^{+1}	15.73^{+2}	4.64	4.25	**4.06**	4.67	4.28	4.10
	FID for various DEIS with $\kappa = 2$ in Eq (30)									
NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	26.91	271^{+1}	362^{+1}	348^{+3}	22.28	21.53	21.43	19.72	16.31	**15.37**
10	11.14	66.25	63.53^{+2}	111^{+2}	7.65	6.89	6.67	6.74	5.49	**5.02**
15	7.06	13.48^{+1}	17.15	44.83^{+1}	4.69	4.16	3.99	4.38	3.78	**3.50**
20	5.47	6.62	4.15^{+1}	15.14	3.70	3.32	3.17	3.57	3.19	**3.03**
50	3.27	2.65	$\mathbf{2.55}^{+1}$	2.57^{+2}	2.70	2.62	2.59	2.70	2.61	2.59
	FID for various DEIS with $\kappa = 3$ in Eq (30)									
NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	32.11	364^{+1}	393^{+1}	383^{+3}	28.87	28.58	28.62	25.78	23.66	**23.38**
10	13.18	135	199^{+2}	298^{+2}	9.89	9.38	9.33	7.74	6.20	**5.77**
15	7.92	42.04^{+1}	99.64	122^{+1}	5.41	4.99	4.91	4.48	3.65	**3.37**
20	5.92	17.05	16.66^{+1}	64.40	4.04	3.69	3.60	3.54	3.05	**2.86**
50	3.36	2.77	2.57^{+1}	2.71^{+2}	2.73	2.63	2.60	2.67	2.59	**2.57**
	FID for various DEIS with Eq (32)									
NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
5	54.85	230^{+1}	382^{+1}	370^{+3}	51.94	51.62	51.58	43.84	39.91	**38.76**
10	19.80	23.35	25.08^{+2}	82.17^{+2}	14.63	13.43	13.07	11.14	7.78	**6.02**
15	11.29	5.63^{+1}	7.46	8.90^{+1}	7.31	6.28	5.90	5.89	4.35	**3.71**
20	7.91	3.84	3.05^{+1}	4.14	4.91	4.19	3.91	4.23	3.35	**3.00**
50	3.82	2.60	$\mathbf{2.56}^{+1}$	2.59^{+2}	2.86	2.70	2.64	2.79	2.63	2.58

Table 4: DEIS for VPSDE on CIFAR10 with $t_0 = 10^{-4}$ in Eq (30)

NFE	DDIM	ρ2Heun	ρ3Heun	ρ4RK	ρAB1	ρAB2	ρAB3	tAB1	tAB2	tAB3
	FID for various DEIS with $\kappa = 7$ in Eq (31)									
5	53.20	108^{+1}	185^{+1}	193^{+3}	47.56	46.36	46.13	36.98	28.76	**25.76**
10	18.99	18.75	20.27^{+2}	54.92^{+2}	13.38	11.84	11.21	10.92	8.26	**6.87**
15	10.91	4.89^{+1}	6.31	9.79^{+1}	6.90	5.86	5.42	6.12	4.86	**4.33**
20	7.81	3.50	$\mathbf{2.97}^{+1}$	3.92	4.84	4.10	3.80	4.48	3.69	3.38
50	3.84	2.60	$\mathbf{2.58}^{+1}$	2.59^{+2}	2.86	2.69	2.64	2.82	2.66	2.61

Table 5: DEIS for VPSDE on CIFAR10 with t_0 and time scheduling suggested by Karras et al. [19]

FID \ NFE / Method	5	10	20	50
A-DDIM	51.47	14.06	6.74	4.04
1-iPNDM	30.13	13.01	8.25	5.65
2-iPNDM	84.00	10.45	6.79	4.73
3-iPNDM	105.38	14.03	5.79	4.24
tAB1-DEIS	20.45	8.11	4.91	3.88
tAB2-DEIS	18.87	7.47	4.66	3.79
tAB3-DEIS	**18.43**	**7.12**	**4.53**	**3.78**

Table 6: Comparison with A-DDIM on the checkpoint and time scheduling provided by [3] on CIFAR10

FID \ NFE / Method	5	10	20	50
iPNDM	54.62	15.32	9.26	8.26
DDIM	49.08	23.52	13.69	9.44
tAB1-DEIS	34.69	13.94	9.55	8.41
tAB2-DEIS	29.50	11.36	8.79	8.29
tAB3-DEIS	**28.09**	**10.55**	**8.58**	**8.25**

Table 7: Sampling quality on VPSDE ImageNet32 $\times$ 32 with the checkpoint provided by Song et al. [36]. Blackbox ODE solver reports FID 8.34 with ODE tolerance 1×10^{-5} (NFE around 130).

NFE	14	26	45	56	62	91
FID	21.11	17.18	9.35	5.12	3.58	3.59

Table 8: More results of VPSDE on CIFAR10 with DOPRI5 ODE solver.

Dataset	FID \ NFE / Method	5	10	20	50
CELEBA	PNDM	-	-	7.60±0.12	3.51±0.03
	iPNDM	59.87±1.01	7.78±0.18	5.58±0.11	3.34±0.04
	DDIM	30.42±0.87	13.53±0.48	6.89±0.11	4.17±0.04
	tAB1-DEIS	26.65±0.63	8.81±0.23	4.33±0.07	3.19±0.03
	tAB2-DEIS	25.13±0.56	7.20±0.21	3.61±0.05	3.04±0.02
	tAB3-DEIS	**25.07±0.49**	**6.95±0.09**	**3.41±0.04**	**2.95±0.03**

Table 9: Mean and standard deviation of multiple runs with 4 different random seeds on the checkpoint and time scheduling provided by Liu et al. [25] on CELEBA.

SDE	FID \ NFE / Method	5	10	20	50
VESDE	tAB0-DEIS	103.52±2.09	46.90±0.38	27.64±0.05	19.86±0.03
	tAB1-DEIS	**56.33±0.87**	26.16±0.12	18.52±0.03	16.64±0.01
	tAB2-DEIS	58.65±0.25	**20.89±0.09**	16.94±0.03	16.33±0.02
	tAB3-DEIS	96.70±0.90	25.01±0.03	**16.59±0.03**	**16.31±0.02**

Table 10: FID results of DEIS on VESDE CIFAR10. We note the Predictor-Corrector algorithm proposed in [35] have ≥ 100 FID if sampling with limited NFE budget (≤ 50).

Figure 7: Generated images with tABr-DEIS on VESDE CIFAR10.

URL	Citation	License
https://github.com/yang-song/score_sde	[35]	Apache License 2.0
https://github.com/luping-liu/PNDM	[25]	Apache License 2.0
https://github.com/CompVis/latent-diffusion	[32]	MIT
https://github.com/baofff/Analytic-DPM	[3]	Unknown

Table 11: Code Used and License

Experiment	Citation	License
CIFAR10 Tab 2, 3 to 5 and 8 to 10, FFHQ Fig 1	[35]	Apache License 2.0
CIFAR10 Tab 6	[3]	Unknown
CELEBA Tab 9	[25]	Apache License 2.0
ImageNet 32×32 Tab 7	[36]	Unknown
Text-to-image	[32]	MIT

Table 12: Checkpoints for experiments

Figure 8: Generated images with text "A artistic painting of snow trees by Pablo Picaso, oil on canvas" (15 NFE)

Figure 9: Generated images with text "A street sign that reads Diffusion" (15 NFE)

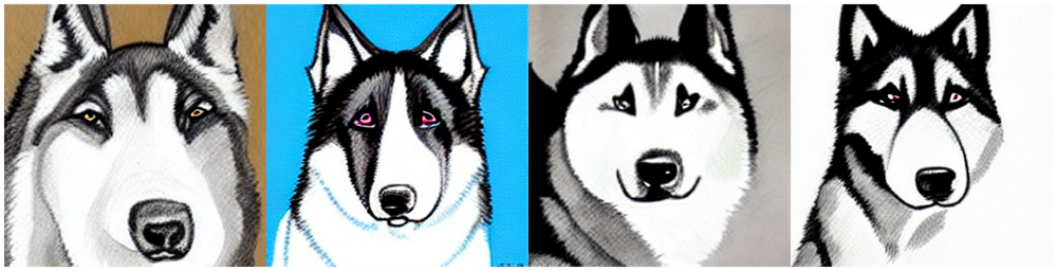


Figure 10: Generated images with text "The drawing of a funny husky" (15 NFE)

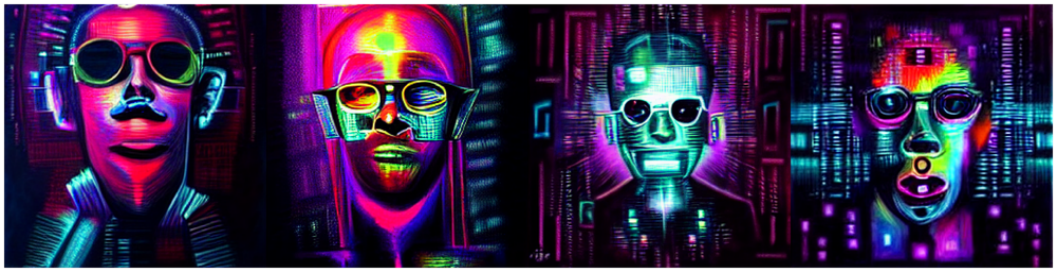


Figure 11: Generated images with text "Cyber punk oil painting" (15 NFE)

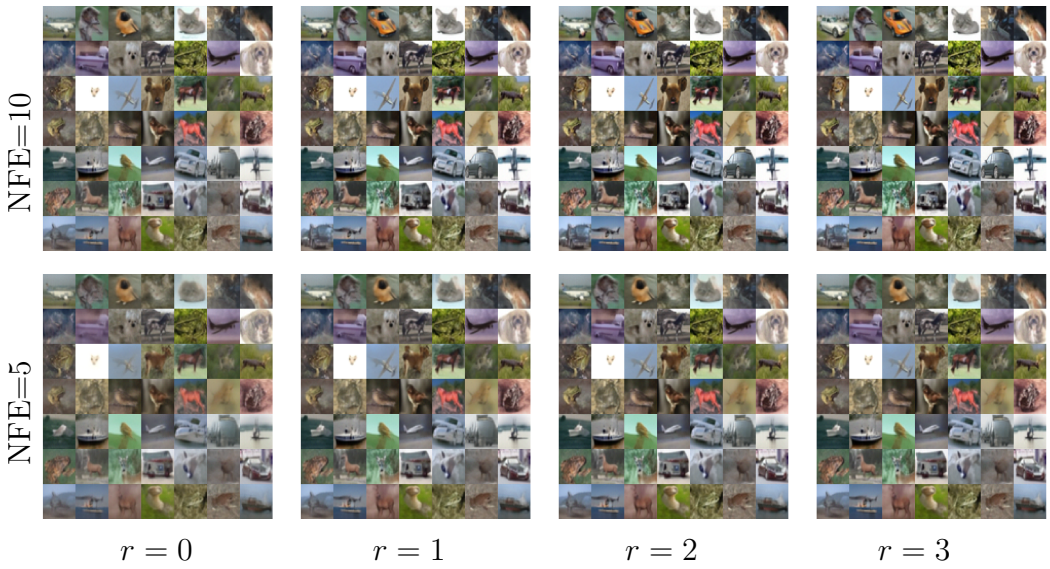


Figure 12: Generated images with DEIS on VPSDE CIFAR10.

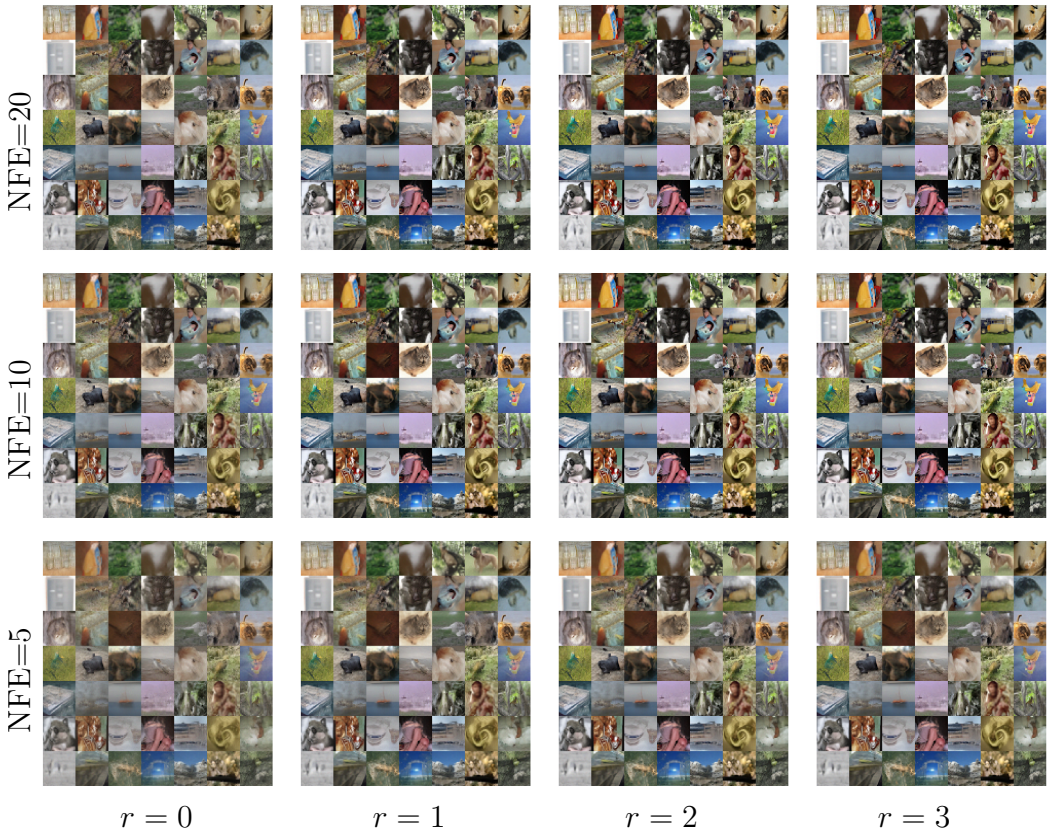


Figure 13: Generated images with DEIS on VPSDE ImageNet 32×32.

Figure 14: Generated images with DEIS on VPSDE CelebA (NFE 5).

Figure 15: Generated images with DEIS on VPSDE CelebA (NFE 10).

Figure 16: Generated images with DEIS on VPSDE CelebA (NFE 20).