QUASI-TAYLOR SAMPLERS FOR DIFFUSION GENERATIVE MODELS BASED ON IDEAL DERIVATIVES

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion generative models have emerged as a new challenger to popular deep neural generative models such as GANs, but have the drawback that they often require a huge number of neural function evaluations (NFEs) during synthesis unless some sophisticated sampling strategies are employed. This paper proposes new efficient samplers based on the numerical schemes derived by the familiar Taylor expansion, which directly solves the ODE/SDE of interest. In general, it is not easy to compute the derivatives that are required in higher-order Taylor schemes, but in the case of diffusion models, this difficulty is alleviated by the trick that the authors call "ideal derivative substitution," in which the higher-order derivatives are replaced by tractable ones. To derive ideal derivatives, the authors argue the "single point approximation," in which the true score function is approximated by a conditional one, holds in many cases, and considered the derivatives of this approximation. Applying thus obtained new quasi-Taylor samplers to image generation tasks, the authors experimentally confirmed that the proposed samplers could synthesize plausible images in small number of NFEs, and that the performance was better or at the same level as DDIM and Runge-Kutta methods. The paper also argues the relevance of the proposed samplers to the existing ones mentioned above.

1 INTRODUCTION

Generative modeling based on deep neural networks is an important research subject for both fundamental and applied purposes, and has been a major trend in machine learning studies for several years. To date, various types of neural generative models have been studied including GANs (Goodfellow et al., 2014), VAEs (Kingma et al., 2021; Kingma & Welling, 2019), normalizing flows (Rezende & Mohamed, 2015), and autoregressive models (van den Oord et al., 2016b;a). In addition to these popular models, a class of novel generative models based on the idea of iteratively refinement using the diffusion process has been rapidly gaining attention recently as a challenger that rivals the classics above (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Song et al., 2020b; Song & Ermon, 2020; Ho et al., 2020; Dhariwal & Nichol, 2021). The diffusion-based generative models have recently been showing impressive results in many fields including image (Ho et al., 2020; Vahdat et al., 2021; Saharia et al., 2021; Ho et al., 2021; Sasaki et al., 2021), video (Ho et al., 2022), text-to-image (Nichol et al., 2021; Ramesh et al., 2022), speech (Chen et al., 2020; 2021; Kong et al., 2021; Popov et al., 2021; Kameoka et al., 2020), symbolic music (Mittal et al., 2021), natural language (Hoogeboom et al., 2021; Austin et al., 2021), chemoinformatics (Xu et al., 2022), etc.

However, while the diffusion models have good synthesis quality, it has been said that they have a fatal drawback that they often require a very large number of iterations (refinement steps) during synthesis, ranging from hundreds to a thousand. In particular, the increase in refinement steps critically reduces the synthesis speed, as each step involves at least one neural function evaluation (NFE). Therefore, it has been a common research question how to establish a systematic method to stably generate good data from diffusion models in a relatively small number of refinement steps, or NFEs in particular. From this motivation, there have already been some studies aiming at reducing the NFEs (See § 2). Among these, Probability Flow ODE (PF-ODE) (Song et al., 2020b) enable efficient and deterministic sampling, and is gaining attention. This framework has the merit of deriving a simple ODE by a straightforward conceptual manipulation of diffusion process. However, the ODE is eventually solved by using a black-box Runge-Kutta solver in the original paper, which requires several NFEs per step and is clearly costly. Another PF-ODE solver includes DDIM (Song et al., 2020a), and is also

commonly used. It is certainly efficient and can generate plausible images. However, it was not originally formulated as a PF-ODE solver, and the relationship between DDIM and PF-ODE is not straightforward.

From these motivations, we provide another sampler to solve the same ODE, which performs better than or on par with DDIM. The derivation outline is simple and intuitive: (1) consider the Taylor expansion of the given system, and (2) replace the derivatives in the Taylor series with appropriate functions; that's all.

The contribution of this paper would be as follows: (1) We propose novel samplers for diffusion models based on Taylor expansion of PF-ODE. They outperformed, or were on par with DDIM, while its derivation would be more elementary. (2) We also show that the same technique is applicable to derive a stochastic solver for reverse-time SDE argued in (Song et al., 2020b), which we call R-SDE in this paper. (3) To derive these algorithms, we show that the derivatives of score function can be approximated by simple functions. We call this technique the *ideal derivative substitution*. (4) This idea is also used to derive a Lipschitz-continuity-aware noise schedule. (5) The fact that proposed methods work well could be an evidence that the assumptions behind our methods are largely valid. This will promote further understanding of the fundamentals of diffusion generative models.

2 Background and Related Work

Diffusion Process to draw a new data from a target density: Let us first briefly summarize the framework of the diffusion-based generative models. Following Song et al. (2020b), we describe the mechanisms using the language of continuous-time diffusion process for later convenience. Let us consider "particles" $\{\mathbf{x}_t\}$ moving in a d-dim space obeying the following Itô diffusion,

$$\text{SDE:} \quad d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(\mathbf{x}_t, t)d\mathbf{B}_t, \tag{1}$$

where $\mathbf{B}_t$ is the d-dim Brownian motion whose temporal increments obeys the standard Gaussian. The drift $\mathbf{f}(\cdot, \cdot)$ is d-dim vector, and the diffusion coefficient $g(\cdot, \cdot)$ is scalar. The SDE describes the microscopic dynamics of each particle. On the other hand, the "population" of the particles obeying the above SDE, i.e. density function $p(\mathbf{x}_t, t \mid \mathbf{x}_s, s), (t > s)$, follows the following PDEs, which are known as Kolmogorov's forward and backward equations; the former is also known as the Fokker-Planck equation (see § D.2),

$$\text{FPE:} \quad \partial_t p(\mathbf{x}_t, t \mid \mathbf{x}_s, s) = -\nabla_{\mathbf{x}_t} \cdot \mathbf{f}(\mathbf{x}_t, t)p(\mathbf{x}_t, t \mid \mathbf{x}_s, s) + \Delta_{\mathbf{x}_t} \frac{g(\mathbf{x}_t, t)^2}{2} p(\mathbf{x}_t, t \mid \mathbf{x}_s, s), \tag{2}$$

$$\text{KBE:} \quad -\partial_s p(\mathbf{x}_t, t \mid \mathbf{x}_s, s) = \mathbf{f}(\mathbf{x}_t, t) \cdot \nabla_{\mathbf{x}_s} p(\mathbf{x}_t, t \mid \mathbf{x}_s, s) + \frac{g(\mathbf{x}_t, t)^2}{2} \Delta_{\mathbf{x}_s} p(\mathbf{x}_t, t \mid \mathbf{x}_s, s), \tag{3}$$

where $\Delta_{\mathbf{x}} := \nabla_{\mathbf{x}}^2$ is Laplacian. (FPE also holds for $p(\mathbf{x}_t, t)$; consider the expectation $\mathbb{E}_{p(\mathbf{x}_s, s)}[\cdot]$.) These PDEs enables us to understand the macroscopic behavior of the particle ensemble. For example, if $\mathbf{f}(\mathbf{x}, t) = -\nabla U(\mathbf{x}), g(\mathbf{x}, t) = \sqrt{2D}$, where $U(\mathbf{x})$ a certain potential and D a constant, then we may verify that the stationary solution of FPE is $p(\mathbf{x}) \propto e^{-U(\mathbf{x})/D}$. It means that we may draw a sample $\mathbf{x}$ that follows the stationary density by evolving the SDE over time. This technique is often referred to as the Langevin Monte Carlo method (Rossky et al., 1978; Roberts & Tweedie, 1996). Some of the diffusion generative models are based on this framework, e.g. (Song & Ermon, 2019; 2020), in which the potential gradient $\nabla U(\mathbf{x})$ is approximated by a neural network.

Another systematic approach is considering the reverse-time dynamics (Song et al., 2020b) based on KBE. Roughly speaking, FPE gives information about the future from the initial density, while KBE gives information about what the past states were likely to be from the terminal density. Here, instead of using KBE directly, it is useful to consider a variant of it which is transformed into the form of FPE, because it has an associated SDE that enables the particle-wise backward sampling (Stratonovich, 1965; Anderson, 1982).

$$\text{R-FPE:} \quad -\partial_s p(\mathbf{x}_s, s \mid \mathbf{x}_t, t) = \nabla_{\mathbf{x}_s} \cdot \bar{\mathbf{f}}(\mathbf{x}_s, s)p(\mathbf{x}_s, s \mid \mathbf{x}_t, t) + \Delta_{\mathbf{x}_s} \frac{\bar{g}(\mathbf{x}_s, s)^2}{2} p(\mathbf{x}_s, s \mid \mathbf{x}_t, t) \tag{4}$$

$$\text{R-SDE:} \quad d\mathbf{x}_s = -\bar{\mathbf{f}}(\mathbf{x}_s, s)(-ds) + \bar{g}(\mathbf{x}_s, s)d\bar{\mathbf{B}}_s. \tag{5}$$

Hereafter, let $g(x_t, t) = g(t)$ for simplicity. Then the specific forms of drift and diffusion coefficients are written as follows,

$$\bar{\mathbf{f}}(\mathbf{x}_t, t) = \bar{\mathbf{f}}_\sharp(\mathbf{x}_t, t) := \mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t), \quad \bar{g}(t) = g(t). \tag{6}$$

Starting from a certain random variable $\mathbf{x}_T$, then by evolving the R-SDE reverse in time, we may obtain a $\hat{\mathbf{x}}_0$ which follows $p(\mathbf{x}_0, 0 \mid \mathbf{x}_T, T)$ (i.e. the solution of R-FPE eq. (4)). Therefore, if the initial density $p(\mathbf{x}_0, 0)$ of the forward dynamics eq. (2) is the true density, then we may utilize this mechanism as a generative model to draw a new sample $\hat{\mathbf{x}}_0$ from it.

Exactly the same initial density is achieved by evolving a deterministic system where the diffusion coefficient is exactly zero, i.e. $\bar{g}(t) = 0$ in eq. (5). This is called the Probability Flow ODE (PF-ODE) (Song et al., 2020b), which is an example of neural ODEs (Chen et al., 2018). The PF-ODE is derived by formally eliminating the diffusion term of FPE eq. (2). The coefficients of PF-ODE are written as follows. See also § D.3.

$$\bar{\mathbf{f}}(\mathbf{x}_t, t) = \bar{\mathbf{f}}_\flat(\mathbf{x}_t, t) \coloneqq \mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t). \quad \bar{g}(t) = 0. \tag{7}$$

Some extensions of this framework include as follows. Dockhorn et al. (2021) introduced the velocity variable considering the Hamiltonian dynamics. Another extension is the introduction of a conditioning parameter, and guidance techniques using it (Dhariwal & Nichol, 2021; Ho & Salimans, 2021; Choi et al., 2021) to promote the dynamics to go to a specific class of images, which has achieved remarkable results in text-to-image tasks (Nichol et al., 2021; Ramesh et al., 2022).

Variance-Preserving Model (VP-SDE Model): The solution of unconditioned FPE is written as the convolution with the initial density $p(\mathbf{x}_0, 0)$ and the *fundamental solution*, or the *heat kernel*, $p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$, which is the solution of the conditional FPE under the assumption that the initial density was delta function, $p(\mathbf{x}_0, 0) = \delta(\mathbf{x}_0 - \mathbf{x}_0^*)$. Although it is still intractable to solve this problem in general, a well-known exception is the (time-dependent) Ornstein-Uhlenbeck process where $\mathbf{f}(\mathbf{x}_t, t) = -\frac{1}{2}\beta_t \mathbf{x}_t$ and $g(\mathbf{x}_t, t) = \sqrt{\beta_t}$; $\beta_t = \beta(t)$ is a non-negative continuous function. In this case, the heat kernel is simply written as follows,

$$p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{1-\nu_t}\mathbf{x}_0, \nu_t \mathbf{I}), \quad \text{where } \nu_t = 1 - e^{-\int_0^t \beta_{t'} dt'}. \tag{8}$$

This model is referred to as the variance-preserving (VP) model by Song et al. (2020b). This model has good properties such as the scale of data $\|\mathbf{x}_t\|_2$ is almost homogeneous, which is advantageous in neural models. Although Song et al. (2020b) also considered variance-exploding (VE) models, we only consider VP-type models in this paper.

Training Objective: In diffusion-based generative models, one estimates the *score function* $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log \mathbb{E}_{p(\mathbf{x}_0, 0)}[p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)]$ by a neural network $\mathbf{S}_\theta(\mathbf{x}_t, t)$. This sort of learning has been referred to as the score matching (Hyvärinen & Dayan, 2005; Vincent, 2011). However, the exact evaluation of this training target is clearly intractable because of the expectation $\mathbb{E}_{p(\mathbf{x}_0, 0)}[\cdot]$, so it has been common to consider a Variational Bayesian surrogate loss; Ho & Salimans (2021) showed that the following loss function approximates the negative ELBO,

$$\mathcal{L} \coloneqq \mathbb{E}[\|-\sqrt{\nu_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) - \mathbf{S}_\theta(\mathbf{x}_t, t)\|_2^2] = \mathbb{E}[\|\tfrac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\sqrt{\nu_t}} - \mathbf{S}_\theta(\mathbf{x}_t, t)\|_2^2] \tag{9}$$

$$= \mathbb{E}[\|\mathbf{w} - \mathbf{S}_\theta(\sqrt{1-\nu_t}\mathbf{x}_0 + \sqrt{\nu_t}\mathbf{w}, t)\|_2^2], \tag{10}$$

where the expectation in eq. (10) is taken w.r.t. $\mathbf{x}_0 \sim \mathcal{D}$, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $t \sim \mathrm{Uniform}([0, T])$. Some variants of the score matching objectives are also studied. For example, Chen et al. (2020) reported that the L_1 loss gave better results than the L_2 loss in speech synthesis. Also, Kingma et al. (2021) argued that the weighted loss with SNR-based weights improves the performance.

It should be noted that the above loss function will actually be very close to the ideal score matching loss function in practice, where the probability is not conditioned on $\mathbf{x}_0$, i.e.,

$$\mathcal{L}_{\text{ideal}} = \mathbb{E}[\|-\sqrt{\nu_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) - \mathbf{S}_\theta(\mathbf{x}_t, t)\|_2^2]. \tag{11}$$

This is because there almost always exists a point $\mathbf{x}_0$ on the data manifold such that $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$ holds with very high accuracy in very high-dim cases, because of the well-known "log-sum-exp $\approx$ max" law. For more details, see § 3.3 and § A.

Sampling Schemes for R-SDE and PF-ODE: Thus obtained $\mathbf{S}_\theta(\mathbf{x}_t, t)$ is expected to finely approximate $-\sqrt{\nu_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t)$, and we may use it in eq. (5). One of the simplest numerical

schemes for solving SDEs is the Euler-Maruyama method (Maruyama, 1955, Theorem. 1) as follows, and many diffusion generative models are actually using it.

$$\text{Euler-Maruyama:} \quad \mathbf{x}_{t-h} \leftarrow \mathbf{x}_t - h\bar{\mathbf{f}}_\sharp(\mathbf{x}_t, t) + \sqrt{h}g(t)\mathbf{w}, \quad \text{where } \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{12}$$

where $h > 0$ is the step size. The error of the Euler-Maruyama method is the order of $O(\sqrt{h})$ in general, though it is actually $O(h)$ in our case; this is because $\nabla_{\mathbf{x}_t} g(t) = 0$. As a better solver for the R-SDE, the Predictor-Corrector (PC)-based sampler was proposed in (Song et al., 2020b). The PC sampler outperformed the Predictor-only strategy, but it requires many NFEs in the correction process, so we will exclude it in our discussion. Another R-SDE solver is the one proposed by Jolicoeur-Martineau et al. (2021), whose NFE per refinement step is 2.

On the other hand, there are also deterministic samplers for PF-ODE eqs. (5), (7) as follows,

$$\text{Euler:} \quad \mathbf{x}_{t-h} \leftarrow \mathbf{x}_t - h\bar{\mathbf{f}}_\flat(\mathbf{x}_t, t) \tag{13}$$

$$\text{Runge-Kutta:} \quad \mathbf{x}_{t-h} \leftarrow \mathbf{x}_t - h\sum_{i=1}^{m} b_i \mathbf{k}_i, \quad \text{where } \mathbf{k}_i = \bar{\mathbf{f}}_\flat(\mathbf{x}_t - h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j, t - hc_i) \tag{14}$$

where $\{a_{ij}\}, \{b_i\}, \{c_i\}$ are coefficients of the Runge-Kutta (RK) method (see § D.5). The error of the Euler method is $o(h)$, and that of the RK method is $o(h^p), p \leq m$ in general (Press et al., 2007, § 16). Another deterministic sampler is DDIM (Song et al., 2020a, Eq. (13)), and is also understood as a PF-ODE solver (Salimans & Ho, 2022). Its NFE per step is only 1, and is capable of efficiently generate samples.

$$\text{DDIM:} \quad \mathbf{x}_{t-h} \leftarrow \sqrt{\frac{1-\nu_{t-h}}{1-\nu_t}}\mathbf{x}_t + \frac{\sqrt{(1-\nu_t)\nu_{t-h}} - \sqrt{(1-\nu_{t-h})\nu_t}}{\sqrt{1-\nu_t}}\mathbf{S}_\theta(\mathbf{x}_t, t). \tag{15}$$

Other techniques that aimed to make sampling faster include as follows. Song & Ermon (2020) proposed some techniques to accelerate the sampling. Watson et al. (2021) proposed a DP-based optimization method to tune noise schedules for faster sampling. Salimans & Ho (2022) proposed distilling the pretrained teacher model to a student model that can predict teacher's several steps in a single step, which is efficient during the sampling but the distillation-based training is costly.

3 Proposed Method: Quasi-Taylor Samplers

3.1 Motivation: Higher-order Straightforward Solvers for R-SDE and PF-ODE

As mentioned above, DDIM already exists as an efficient sampler of diffusion generative models, but it is not necessarily derived directly from PF-ODEs, and its relationship to PF-ODEs was revealed through a little argumentation (Song et al., 2020a; Salimans & Ho, 2022). Since PF-ODE gives a basis for the diffusion generative models, developing a sampler that directly solves it through intuitive and straightforward argumentation would be beneficial in various contexts in the future development of this field. Of course, the RK method mentioned directly solves the PF-ODE, but it requires several NFEs in a single refinement step, and it seems to be behind in competitiveness despite its high convergence rate, since the state-of-the-art samplers e.g. (Salimans & Ho, 2022) are already generating data in single digit NFEs.

From these motivations, we propose a simple sampler based on the Taylor expansion, a very basic technique that is familiar to many researchers and practitioners. In general, Taylor methods are not very popular as numerical schemes because they require higher-order derivatives, which are not always tractable. Nevertheless, in diffusion models, the derivatives are expected to have good structure, and are effectively evaluated. This section describes the details of the idea, and derives solvers for both PF-ODE and R-SDE. We summarize the entire procedures as pseudocodes in § E.

3.2 Taylor Scheme for ODE and Itô-Taylor Scheme for SDE

For simplicity, we consider the 1-dim case in this section. It can be easily generalized to multidimensional cases simply by parallelizing each dimension, since the Jacobian matrix is diagonal assuming that each dimension is independent of each other.

Taylor Scheme for Deterministic Systems Given a ODE $\dot{x}_t = a(x_t, t)$, we can consider the Taylor expansion of it. Using a differential operator $L_\flat \coloneqq \left(\partial_t + a(t, x_t)\partial_{x_t}\right)$, we can write the Taylor expansion of the path x_t as follows. If we ignore $o(h^p)$ terms of the series, we may obtain a numerical scheme of order p. See also § D.1.1.

$$x_{t+h} = x_t + ha(x_t, t) + \frac{h^2}{2!} L_\flat a(x_t, t) + \frac{h^3}{3!} L_\flat^2 a(x_t, t) + \cdots . \tag{16}$$

Itô-Taylor Scheme for Stochastic Systems In stochastic systems, the Taylor expansion requires modifications. If x_t obeys a stochastic system $dx_t = a(x_t, t)dt + b(x_t, t)dB_t$, then the path is written in a stochastic version of Taylor-like series, which is often called the Itô-Taylor expansion, a.k.a. Wagner-Platen expansion (Platen & Wagner, 1982);(Kloeden et al., 1994, § 2.3.B);(Särkkä & Solin, 2019, § 8.2). The Itô-Taylor expansion is based on the following differential operators $L_\sharp, G_\sharp$, which are based on Itô's formula (Itô, 1944).

$$L_\sharp \coloneqq \partial_t + a(x, t)\partial_x + \frac{1}{2} b(x, t)^2 \partial_x^2, \quad G_\sharp \coloneqq b(x, t)\partial_x \tag{17}$$

In (Kloeden & Platen, 1992), a number of higher order numerical schemes for SDEs based on the Itô-Taylor expansion are presented. One of the simplest of them is as follows. See also § D.1.2.

Theorem 1 (Kloeden & Platen (1992, § 14.2): An Itô-Taylor scheme of weak order $\beta = 2$)**.** *Let x_t obeys the above SDE, and let the differential operators $L_\sharp, G_\sharp$ be given by eq. (17). Then, the following numerical scheme weakly converges with the order of $\beta = 2$.*

$$x_{t+h} \leftarrow x_t + ha + \tilde{w}_t b + \frac{\tilde{w}_t^2 - h}{2} G_\sharp b + \frac{h^2}{2} L_\sharp a + (\tilde{w}_t h - \tilde{z}_t) L_\sharp b + \tilde{z}_t G_\sharp a \tag{18}$$

where $\tilde{w}_t = \sqrt{h} w_t, \tilde{z}_t = h\sqrt{h} z_t$ are correlated Gaussian random variables, and w_t, z_t are given by $w_t = u_1$ and $z_t = \frac{1}{2} u_1 + \frac{1}{2\sqrt{3}} u_2$, where $u_1, u_2 \sim \mathcal{N}(0, 1)$ (i.i.d.). The notations $a, L_\sharp a$, etc. are the abbreviations for $a(x_t, t), (L_\sharp a)(x_t, t)$, etc.

This scheme has the order $\beta = 2$ of weak convergence (see § D.4). In addition, in a special case where $G_\sharp^2 b \equiv 0$, the strong $\gamma = 1.5$ convergence is also guaranteed (Kloeden & Platen, 1992, § 10.4).

3.3 Single Point Approximation of the Score Function

Before proceeding, let us introduce the single point approximation of score function that $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t)$ almost certainly has a some point $\mathbf{x}_0$ on the data manifold such that the following approximation holds,

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log \int p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) p(\mathbf{x}_0, 0) d\mathbf{x}_0 \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0). \tag{19}$$

To date, this approximation has often been understood as "a tractable surrogate." However, the error between the integral and the single point approximation is actually very small in practical scenarios. More specifically, the following facts can be shown under some assumptions.

1. The relative L_2 distance between $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t)$ and $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$ is bounded above by $\sqrt{(1 - \nu_t)/\nu_t}$ for any point $\mathbf{x}_0$ on the "data manifold" in practical scenarios.
2. When the noise level is low $\nu_t \approx 0$, and the data space is sufficiently high-dimensional, the distant points far from $\mathbf{x}_t$ do not contribute to the integral. If the data manifold, which is sufficiently flat locally, is locally a k-dim subspace of the entire d-dim data space where $1 \ll k \ll d$, then the relative L_2 distance is bounded above by $2\sqrt{k/d}$.

Of course, the single point approximation is not always valid. In fact, the approximation tends to break down when the noise level ν_t is around 0.9 (SNR $= (1 - \nu_t)/\nu_t$ is around 0.1). In this region, the single point approximation can deviates from the true gradient by about 20% in some cases. Conversely, however, it would be also said that the error is as small as this level even in the worst empirical cases. For more details on this approximation, see § A.

3.4 TAYLOR AND ITÔ-TAYLOR SCHEMES WITH IDEAL DERIVATIVES

Now let us adopt the above Taylor schemes to our problem setting where the base SDE is eq. (5), and $\bar{\mathbf{f}}_\sharp$ and $\bar{\mathbf{f}}_\flat$ are given by eqs. (6), (7). Under these conditions, the operators are written as follows; note that the time evolves backward in time in our case, the temporal derivative should be $-\partial_t$,

$$L_\flat = -\partial_t - \left(\bar{\mathbf{f}}_\flat(\mathbf{x}_t, t) \cdot \nabla_{\mathbf{x}_t}\right), \quad L_\sharp = -\partial_t - \left(\bar{\mathbf{f}}_\sharp(\mathbf{x}_t, t) \cdot \nabla_{\mathbf{x}_t}\right) + \frac{\beta_t}{2}\Delta_{\mathbf{x}_t}, \quad G_\sharp = \sqrt{\beta_t}\left(\mathbf{1} \cdot \nabla_{\mathbf{x}_t}\right),$$
$$\text{where } \bar{\mathbf{f}}_\flat(\mathbf{x}_t, t) = -\frac{\beta_t}{2}\mathbf{x}_t + \frac{\beta_t}{2\sqrt{\nu_t}}\mathbf{S}_\theta(\mathbf{x}_t, t), \quad \bar{\mathbf{f}}_\sharp(\mathbf{x}_t, t) = -\frac{\beta_t}{2}\mathbf{x}_t + \frac{\beta_t}{\sqrt{\nu_t}}\mathbf{S}_\theta(\mathbf{x}_t, t). \tag{20}$$

It is not easy in general to evaluate expressions involving such many derivatives. Indeed, for example, $L_\flat(-\bar{\mathbf{f}}_\flat)$ has the derivatives of the learned score function, viz. $\partial_t \mathbf{S}_\theta(\mathbf{x}_t, t)$ and $(\bullet \cdot \nabla_{\mathbf{x}_t})\mathbf{S}_\theta(\mathbf{x}_t, t)$, which are intractable to evaluate exactly. Fortunately, however, by using the trick which the authors call the "*ideal derivative substitution*", we may write all of the derivatives above explicitly. Since the score function has a single point approximation eq. (19) in many cases, we may assume that the derivatives should ideally hold following equalities. For derivation, see § B.

Proposition 1 (Ideal Derivatives)**.** *Let* $\mathbf{a}$ *be an arbitrary vector. Following approximations hold in many cases. We call them the "ideal derivatives".*

$$\left(\mathbf{a} \cdot \nabla_{\mathbf{x}_t}\right)\mathbf{S}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\nu_t}}\mathbf{a}, \quad -\partial_t \mathbf{S}_\theta(\mathbf{x}_t, t) = -\frac{\beta_t}{2\sqrt{\nu_t}}\left(\mathbf{x}_t - \frac{\mathbf{S}_\theta(\mathbf{x}_t, t)}{\sqrt{\nu_t}}\right). \tag{21}$$

Using the above ideal derivatives as well as the relation $\dot{\nu}(t) = (1 - \nu_t)\beta_t$, we can compute symbolic expressions for $L_\flat(-\bar{\mathbf{f}}_\flat)$, $L_\sharp(-\bar{\mathbf{f}}_\sharp)$, $L_\sharp(g)$, $G_\sharp(-\bar{\mathbf{f}}_\sharp)$ and $G_\sharp(g)$ by routine calculations, which can be easily automated by computer algebra systems such as SymPy (Meurer et al., 2017) and Mathematica. These expressions consist of only $\mathbf{x}_t, \mathbf{S}_\theta(\mathbf{x}_t, t), \nu_t, \beta_t$ and derivatives of β_t. Using them, we can derive Taylor schemes for both deterministic and stochastic systems as follows.

Algorithm 1 (Quasi-Taylor Sampler with Ideal Derivatives for PF-ODE)**.** *Starting from a Gaussian noise* $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$*, iterate the following refinement steps until* $\mathbf{x}_0$ *is obtained.*

$$\mathbf{x}_{t-h} = \rho^\flat_{t,h}\mathbf{x}_t + \mu^\flat_{t,h}\mathbf{S}_\theta(\mathbf{x}_t, t)/\sqrt{\nu_t}, \textit{where} \tag{22}$$

$$\rho^\flat_{t,h} = 1 + \frac{\beta_t h}{2} + \frac{h^2}{4}\left(\frac{\beta_t^2}{2} - \dot{\beta}_t\right) + \frac{h^3}{4}\left(\frac{\beta_t^3}{12} - \frac{\beta_t\dot{\beta}_t}{2} + \frac{\ddot{\beta}_t}{3}\right) + \cdots, \tag{23}$$

$$\mu^\flat_{t,h} = -\frac{\beta_t h}{2} + \frac{h^2}{4}\left(\dot{\beta}(t) - \frac{\beta^2(t)}{2\nu_t}\right) + \frac{h^3}{4}\left(\frac{\beta_t^3(-\nu_t^2 + 3\nu_t - 3)}{12\nu_t^2} + \frac{\beta_t\dot{\beta}_t}{2\nu_t} - \frac{\ddot{\beta}_t}{3}\right) + \cdots. \tag{24}$$

Using terms up to $O(h^2)$, the sampler will have 2nd-order convergence (henceforth referred to as *Taylor 2nd*), and using terms up to $O(h^3)$, the sampler will 3rd-order convergent (similarly, *Taylor 3rd*). If we use up to the $O(h)$ terms, the algorithm is same as the Euler method.

Algorithm 2 (Quasi-Itô-Taylor Sampler with Ideal Derivatives for R-SDE)**.** *Starting from a Gaussian noise* $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$*, iterate the following refinement steps until* $\mathbf{x}_0$ *is obtained.*

$$\mathbf{x}_{t-h} = \rho^\sharp_{t,h}\mathbf{x}_t + \mu^\sharp_{t,h}\mathbf{S}_\theta(\mathbf{x}_t, t)/\sqrt{\nu_t} + \mathbf{n}^\sharp_{t,h}, \textit{where} \tag{25}$$

$$\rho^\sharp_{t,h} = 1 + \frac{\beta_t}{2}h + \frac{h^2}{4}\left(\frac{\beta_t^2}{2} - \dot{\beta}(t)\right), \qquad \mu^\sharp_{t,h} = -\beta_t h + \frac{\dot{\beta}(t)h^2}{2}, \tag{26}$$

$$\mathbf{n}^\sharp_{t,h} = \sqrt{\beta_t}\sqrt{h}\mathbf{w}_t + h^{3/2}\left(-\frac{\dot{\beta}_t}{2\sqrt{\beta_t}}(\mathbf{w}_t - \mathbf{z}_t) + \frac{\beta_t^{3/2}(\nu_t - 2)}{2\nu_t}\mathbf{z}_\tau\right). \tag{27}$$

The Gaussian variables $\mathbf{w}_t$ *and* $\mathbf{z}_t$ *have dimension-wise correlations, and each dimension is sampled similarly to Theorem 1.*

At first glance, these algorithms may appear to be very complex. However, the computational complexity hardly increases compared to the Euler or Euler-Maruyama methods, because almost all of the computational cost is accounted for by the neural network $\mathbf{S}_\theta(\mathbf{x}_t, t)$, and the costs for scalar values $\rho^\bullet_{t,h}, \mu^\bullet_{t,h}$ and noise generation $\mathbf{n}^\sharp_{t,h}$ are almost negligible. It should also be noted that these scalar values can be pre-computed and stored in the memory before synthesis. Thus the computational complexity of these methods are practically equal to Euler, Euler-Maruyama, and DDIM methods.

3.5 SPECIFIC SCHEDULES OF β_t, ν_t AND h

Lipschitz-continuity-aware Noise Schedule Functions: Let us specify the noise schedule. A simplest option would be the linear one, i.e., $\beta_t = \beta_0 + 2\beta_1 t, \nu_t = 1 - e^{-\beta_0 t - \beta_1 t^2}$, and another would be the cosine schedule proposed in (Nichol & Dhariwal, 2021), $\nu_t = \sin^2(\pi t/2), \beta_t = \min(\textit{threshold}, \pi \tan(\pi t/2))$. However, we propose the following parametric noise schedule,

$$\beta_t = \dot{\lambda}(t) \tanh(\lambda(t)/2), \quad \nu_t = \tanh^2(\lambda(t)/2). \tag{28}$$

The parameter function $\lambda(t)$ has some options, but we chose the softplus, $\lambda(t) = \log(1 + Ae^{kt})$ where $0 < A \ll 1$ and $0 < k$, because under the ideal-derivative approximation, it satisfies the Lipschitz continuity condition required by the Picard-Lindelöf (Cauchy-Lipschitz) theorem, which guarantees the existence of the unique solution of the ODE; though it does not rigorously satisfy the requirement that $\nu_0 = 0$ (see eq. (8)).

These scheduling functions have some advantages. Firstly, all the coefficients do not diverge. In particular, the term $\beta_t/\sqrt{\nu_t}$ that appears in $\mu_\bullet(t,h)/\sqrt{\nu_t}$ equals to $\dot{\lambda}(t)$ which does not explode near $t \sim 0$, contrary to the case of linear schedule where the term diverges, i.e. $\lim_{t\to 0} \beta_t/\sqrt{\nu_t} = \infty$. Moreover, unlike the cosine schedule where β_t is modified in an ad hoc manner, our β_t has an advantage of being naturally bounded and differentiable (the differentiability of β_t is particularly essential for Taylor schemes) while our ν_t has a similar curve to that of the empirically validated cosine schedule. Another merit is that its parameters A, k are easy to tune, since they are explicitly written in a closed form as follows, if initial and terminal noise levels $\nu_0 \approx 0, \nu_T \approx 1$ are specified,

$$A = \frac{2\sqrt{\nu_0}}{1 - \sqrt{\nu_0}}, \quad k = \frac{1}{T}\left(\log \frac{2\sqrt{\nu_T}}{1 - \sqrt{\nu_T}} - \log A\right). \tag{29}$$

Schedule of step-size h: The simplest step-size schedule is the equally spaced one, i.e., $h = T/N$, where T is the number of refinement steps. Alternatively, another simple choice would be the use of exponentially decreasing h_i-s, i.e., $h_i = r^{i-1}h_1, (1 \le i \le N)$, where h_i is the step-size of i-th refinement step. In this paper, h_1 and r were determined so that $\sum_{i=1}^N h_i = T$ and $r^N = 0.1$. In our pilot study, the exponentially decreasing schedule was found to give faster convergence.

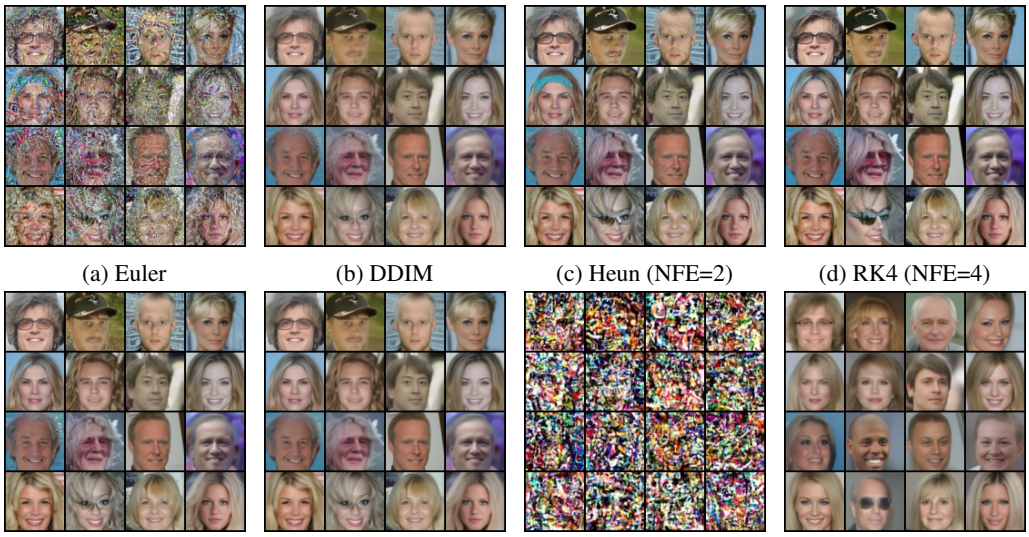
4 IMAGE SYNTHESIS EXPERIMENT

General Configuration: In this section, we conduct experiments to verify the effectiveness of the methods developed in this paper. Specifically, we compare the performance of the Euler scheme eq. (13), *Taylor 2nd* (proposed; Alg. 1), *Taylor 3rd* (proposed; Alg. 1), DDIM (Song et al., 2020a), and the Runge Kutta methods (Heun and Classical RK4 § D.5; these are less efficient than others because of NFEs per step) for PF-ODE, as well as the Euler-Maruyama scheme eq. (12) and *Itô-Taylor* (proposed; Alg. 2) for R-SDE.

■ *Data*: In our experiment, we used the CIFAR-10 (32×32) (Krizhevsky, 2009) and CelebA (64×64) (Liu et al., 2015) datasets. Separate networks were trained for each dataset.

■ *Network and Loss*: The network structure of $\mathbf{S}_\theta(\mathbf{x}_t, t)$ and the experiment code were based on the implementation provided by Song et al. (2020b). The network structure was "NCSN++" implemented in their official PyTorch code. The network consisted of 4 levels of resolution, with the feature dimension of each level being $128 \to 128 \to 256 \to 256 \to 256$. Each level consisted of BigGAN-type ResBlocks, and the number of ResBlocks in each level was 8 (CIFAR-10) and 4 (CelebA). When feeding the time parameter t to the network, we used the noise embedding $\sqrt{1 - \nu_t}$ instead of the default t embedding, following e.g. (Chen et al., 2020). This can mitigate the noise schedule mismatches between training and synthesis, and will be useful to search synthesis noise schedules. The loss function we used was the SNR-weighted L_2 loss Kingma et al. (2021).

■ *Training*: The optimizer was Adam (Kingma & Ba, 2014). The machine used for training was an in-house Linux server dedicated to medium-scale machine learning training with four GPUs (NVIDIA Tesla V100). The batch size was 256. The number of training steps was 100k, and the training took about a day for each dataset. The integration duration was $T = 1$, and the noising schedule was determined by the initial and terminal values $(\nu_0, \nu_T) = (5 \times 10^{-4}, 0.995)$.



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

Figure 1: Comparison of the synthesis results of CelebA (64×64) data. The noise schedule for synthesis was the condition (ii). The number of refinement steps is $N = 12$.

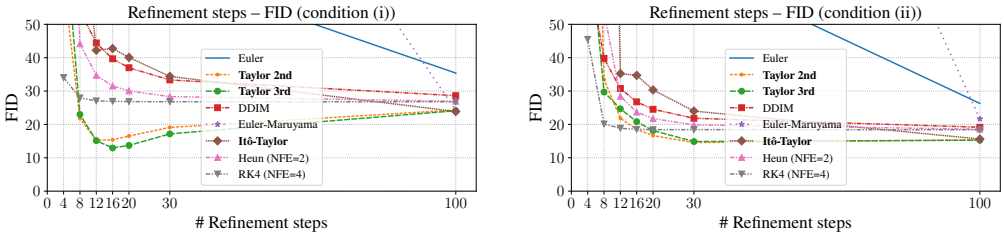


Figure 2: FID scores (↓) for generated CelebA images (64×64).

■ *Synthesis and Evaluation*: During the synthesis, the noise schedule was determined by the following two initial and terminal values; condition (i): $(\nu_0, \nu_T) = (5 \times 10^{-4}, 0.995)$, and condition (ii): $(\nu_0, \nu_T) = (1 \times 10^{-4}, 0.99)$. As a quality assessment metric, we used the Fréchet Inception Distance (FID) (Heusel et al., 2017; Seitzer, 2020). 50,000 images were randomly generated for each condition to compute the FID scores.

Result and Discussion: Figure 1 and Figure 3 show random samples for each sampler. More examples are available in § F. Figure 2 and Figure 4 reports the FID scores. The deterministic samplers considered in this paper generated plausible images much faster than the vanilla Euler-Maruyama sampler, and the proposed stochastic sampler, i.e. the Itô-Taylor, also showed sufficiently competitive results, in terms of both FID scores and visual impression. Based on these results, we may conclude that the use of the Euler-Maruyama sampler is not recommended, but other higher-order methods should be used in situations where the synthesis efficiency matters. In particular, when the synthesis noise schedule was (i), the proposed Quasi-Taylor methods tend to give good results around $N = 16, 20$, and the FID deteriorates from there. The cause is yet unclear at this point, and should be investigated in the future. Nevertheless, regardless of the cause, this fact may be useful in applications.

It can be observed that all of the PF-ODE solvers produce images with generally similar trends, although there is some variation in quality. This would be the consequence of the fact that all the methods are solving the same ODE although the ideas and the accuracy are different. The R-SDE solvers produced images that tend to be slightly different from these deterministic samplers because they are solving different problems. However, some similarities can be seen in the overall direction (see Figure 20).

(a) DDIM (b) RK4 (NFE=4) (c) **Taylor (3rd)** (d) **Itô-Taylor**

Figure 3: Comparison of the synthesis results of CIFAR-10 (32×32) data. The noise schedule for synthesis was the condition (ii). The number of refinement steps is $N = 30$.

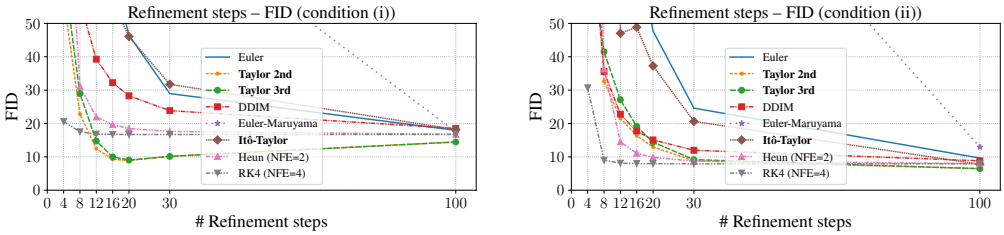


Figure 4: FID scores (↓) for generated CIFAR-10 images (32×32).

What would be particularly important here would be the fact that our Quasi-Taylor methods show qualitatively similar results as derivative-free Runge-Kutta methods (Heun and Classical RK4). This implicitly suggests that the approximated *ideal derivatives* of score function we are using are actually very close to the *true derivatives*, since the p-th order Runge-Kutta method actually approximates up to p-th order terms of the *true* Taylor series based on the true derivatives of the learned score function; see § D.5. This implicitly supports the validity of our hypotheses of single point approximation, ideal derivative approximation, and assumptions to support them.

Another important issue is the relation to the DDIM. As we will discuss in § B.5, DDIM is identical to our Quasi-Taylor methods at least up to 3rd order terms, which suggests that the ideas behind DDIM and our ideal derivative are logically closely related. However, despite this fact, DDIM is considerably less effective than the Quasi-Taylor methods in some cases. It could be hypothesized that this difference may be due to the less-reliable higher order derivatives that are automatically incorporated into DDIM, which may have a negative effect. Further studies will be needed on this point.

5 Concluding Remarks

This paper proposed a Taylor-expansion approach for diffusion generative models, particularly the Probability Flow ODE (PF-ODE) and the reverse-time SDE (R-SDE) solvers. The assumptions to derive our sampler were minimalistic, and the derivation process was straightforward. We just substituted the derivatives in the Taylor series by ideal ones which are approximately specified by the model structure.

The obtained Quasi-Taylor and Quasi-Itô-Taylor samplers performed better than or on par with DDIM and Runge-Kutta methods. This fact implicitly supports the validity of our approximations. Conversely, if we could find some examples where the Quasi-Taylor methods, DDIM and RK methods gave decisively different results, we might be able to gain a deeper understanding of the structure of data manifold and the fundamentals of diffusion models by investigating the causes of discrepancy.

Reproducibility Statement Pseudocodes of the proposed methods are available in § E, and the derivation of the proposed method is described in § B, § C. The experiment is based on open source code with minimal modifications to match the proposed method, and all the data used in this paper are publicly available. Experimental conditions are elaborated in § 4.

Ethics Statement As a final note, negative aspects of generative models are generally pointed out, such as the risk of reproducing bias and discrimination in training data and the risk of being misused for deep fakes. Since this method only provides a solution to existing generative models, it does not take special measures against these problems. Maximum ethical care should be taken in the practical application of this method.

REFERENCES

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Jacob Austin, Daniel Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34, 2021.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations* (*ICLR*), 2020.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, Najim Dehak, and William Chan. WaveGrad 2: Iterative refinement for text-to-speech synthesis. In *Proceedings of INTERSPEECH*, pp. 3765–3769. ISCA, 2021.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021.

David Roxbee Cox and Hilton David Miller. *The theory of stochastic processes*. Routledge, 1965.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped Langevin diffusion. *arXiv preprint arXiv:2112.07068*, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*, 2021.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *Advances in Neural Information Processing Systems*, 2021.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Kiyosi Itô. Stochastic integral. *Proceedings of the Imperial Academy*, 20(8):519–524, 1944. doi: 10.3792/pia/1195572786.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, Nobukatsu Hojo, and Shogo Seki. VoiceGrad: Non-parallel any-to-many voice conversion with annealed Langevin dynamics. *arXiv preprint arXiv:2010.02977*, 2020.

Samuel Karlin and Howard E Taylor. *A second course in stochastic processes*. Academic Press, 1981.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *arXiv preprint arXiv:2107.00630*, 2021.

Peter E Kloeden and Eckhard Platen. Numerical solution of stochastic differential equations. Stochastic Modelling and Applied Probability, Applications of Mathematics. Springer, 1992. doi: 10.1007/978-3-662-12616-5. Corrected 3rd Printing.

Peter E Kloeden, Eckhard Platen, and Henri Schurz. *Numerical solution of SDE through computer experiments*. Universitext. Springer, 1994. doi: 10.1007/978-3-642-57913-4. Corrected 3rd Printing.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations* (*ICLR*), 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. Dataset license: MIT.

Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pp. 1302–1338, 2000.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database, 2010. `http://yann.lecun.com/exdb/mnist`, Dataset license: CC-SA 3.0.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision* (*ICCV*), pp. 3730–3738, 2015. Dataset license: available for non-commercial research purpose only.

Gisirō Maruyama. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4(1):48–90, 1955.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL `https://doi.org/10.7717/peerj-cs.103`.

Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference* (*ISMIR*), 2021.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning* (*ICML*). PMLR, 2021.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Bernt Øksendal. *Stochastic differential equations: an introduction with applications*. Springer, 2013.

Eckhard Platen and Wolfgang Wagner. On a Taylor formula for a class of Ito processes. *Probability and Mathematical Statistics*, 3:37–51, 1982.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-TTS: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning* (*ICML*), pp. 8599–8608. PMLR, 2021.

William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Gareth O Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

Peter J Rossky, Jimmie D Doll, and Harold L Friedman. Brownian dynamics as smart Monte Carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 1978.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10 of *Institute of Mathematical Statistics Textbooks*. Cambridge University Press, 2019.

Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. UNIT-DDPM: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021.

Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid` (Apache-2.0 license), August 2020. Version 0.1.1.

Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer, 2004.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning* (*ICML*), pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations* (*ICLR*), 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in Neural Information Processing Systems*, 2020.

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* (*ICLR*), 2020b. `https://github.com/yang-song/score_sde_pytorch` (Apache-2.0 license).

Ruslan Leont'evich Stratonovich. Conditional Markov processes. In *Non-linear transformations of stochastic processes*, pp. 427–453. Elsevier, 1965.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 2021.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016b.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

A ON THE APPROXIMATION $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$

Although $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t)$ and $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$ are clearly distinct concepts, they are nearly equivalent in practical situations that the data space is high-dimensional and the data are distributed in a small subset (low-dimensional manifold) of the space. That is, in such a case, the integrated gradient $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t)$, given by

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) &= \nabla_{\mathbf{x}_t} \log \int p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) p(\mathbf{x}_0, 0) d\mathbf{x}_0 \\
&= \frac{\nabla_{\mathbf{x}_t} \int p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) p(\mathbf{x}_0, 0) d\mathbf{x}_0}{\int p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) p(\mathbf{x}_0, 0) d\mathbf{x}_0} \\
&= \frac{\mathbb{E}_{p(\mathbf{x}_0)} \left[\nabla_{\mathbf{x}_t} p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) \right]}{\mathbb{E}_{p(\mathbf{x}_0)} \left[p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) \right]} \\
&= \frac{\mathbb{E}_{p(\mathbf{x}_0)} \left[-\frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\nu_t} p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) \right]}{\mathbb{E}_{p(\mathbf{x}_0)} \left[p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) \right]} \\
&= \mathbb{E}_{p(\mathbf{x}_0)} \left[\underbrace{-\frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\nu_t}}_{=\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)} \underbrace{\frac{e^{-\|\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0\|^2/2\nu_t}}{\mathbb{E}_{p(\mathbf{x}_0)}[e^{-\|\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0\|^2/2\nu_t}]}}_{=:q(\mathbf{x}_0 \mid \mathbf{x}_t)} \right],
\end{aligned} \tag{30}$$

almost always has a single point approximation which is written as $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)$, where $\mathbf{x}_0^{(i)}$ is a certain point on the data manifold. There are two phases depending on the noise scale $\bar{\alpha}_t = 1 - \nu_t$, and both phases have different reasons for the validity of the approximation.

A.1 PHASE (1): ANYONE CAN BE A REPRESENTATIVE ($\bar{\alpha}_t = 1 - \nu_t \sim 0$)

If $\mathbf{x}_t$ is far from any of $\{\sqrt{1-\nu_t}\mathbf{x}_0^{(i)} \mid \mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)\}$, the gradients from each scaled data points are almost the same. Therefore, the integrated one viz. $\nabla \log p(\mathbf{x}_t, t)$ can be approximated by any of $\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)$ for any $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$. Figure 5 intuitively illustrates the reason.

More quantitatively, noting that the weight function $q(\cdot)$ satisfies $\mathbb{E}_{p(\mathbf{x}_0)}[q(\mathbf{x}_0 \mid \mathbf{x}_t)] = 1$, the L_2 distance between $\nabla \log p(\mathbf{x}_t, t)$ and $\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)$ is bounded above as follows,

$$\begin{aligned}
&\|\nabla \log p(\mathbf{x}_t, t) - \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)\|_2^2 \\
&= \left\| \mathbb{E}_{p(\mathbf{x}_0)} \left[\frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\nu_t} q(\mathbf{x}_0 \mid \mathbf{x}_t) \right] - \frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0^{(i)}}{\nu_t} \right\|_2^2 \\
&= \left\| \mathbb{E}_{p(\mathbf{x}_0)} \left[\left(\frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\nu_t} - \frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0^{(i)}}{\nu_t} \right) q(\mathbf{x}_0 \mid \mathbf{x}_t) \right] \right\|_2^2 \\
&= \frac{1-\nu_t}{\nu_t^2} \left\| \mathbb{E}_{p(\mathbf{x}_0)} \left[(\mathbf{x}_0 - \mathbf{x}_0^{(i)}) q(\mathbf{x}_0 \mid \mathbf{x}_t) \right] \right\|_2^2 \\
&\leq \frac{1-\nu_t}{\nu_t^2} \mathbb{E}_{p(\mathbf{x}_0)} \left[\|\mathbf{x}_0 - \mathbf{x}_0^{(i)}\|_2^2 q(\mathbf{x}_0 \mid \mathbf{x}_t) \right] \quad (\because \text{Jensen's inequality}) \\
&\leq \frac{1-\nu_t}{\nu_t^2} \left(\max_{\mathbf{x}_0} \|\mathbf{x}_0 - \mathbf{x}_0^{(i)}\|_2^2 \mathbb{E}_{p(\mathbf{x}_0)} \left[q(\mathbf{x}_0 \mid \mathbf{x}_t) \right] \right) \\
&= \frac{1-\nu_t}{\nu_t^2} \left(\max_{\mathbf{x}_0} \|\mathbf{x}_0 - \mathbf{x}_0^{(i)}\|_2^2 \right) \\
&\leq \frac{4(1-\nu_t) R_{\mathcal{M}}^2}{\nu_t^2},
\end{aligned} \tag{31}$$

where $R_{\mathcal{M}}$ is the radius of the smallest ball that covers the data manifold $\mathcal{M}$. The radius $R_{\mathcal{M}}$ will be a finite constant in most practical scenarios we are interested in. For example, if the data space is d-dim box, i.e., $\mathcal{M} \subset [0, 1]^d$, then $R_{\mathcal{M}}$ is bounded above by $R_{\mathcal{M}} \leq \sqrt{d}/2$.

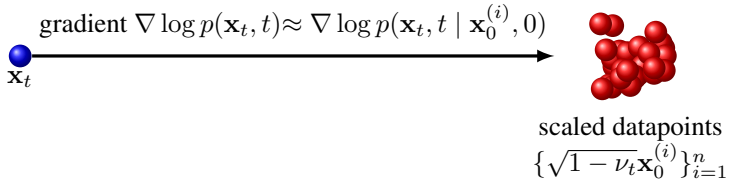


Figure 5: Intuitive reasons why a single point approximation is valid. The case where $1 - \nu_t \approx 0$

Noting that $\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)\|_2 \approx \sqrt{d/\nu_t}$ (this is actually a random variable that follows the χ-distribution[1]), the relative L_2 error is evaluated as follows,

$$(\text{relative } L_2 \text{ error}) = \frac{\|\nabla \log p(\mathbf{x}_t, t) - \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)\|_2}{\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)\|_2} \lessapprox \sqrt{\frac{1-\nu_t}{\nu_t}}. \tag{32}$$

Similarly, the cosine similarity is evaluated as

$$\text{cossim}(\nabla \log p(\mathbf{x}_t, t), \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)) \gtrapprox 1 - \frac{1}{2}\left(1 + \sqrt{\frac{1-\nu_t}{\nu_t}}\right)\frac{1-\nu_t}{\nu_t}. \tag{33}$$

These bounds suggest that the relative error between the true gradient and the single point approximation approaches to 0, and the cosine similarity goes to 1, as $\nu_t \to 1$ ($\bar{\alpha}_t \to 0$). That is, the single point approximation is largely valid when the noise level ν_t is high.

Proofs for some facts used in this section

■ *Jensen's inequality*: A sketchy proof of Jensen's inequality for L_2-norm is as follows:

$$\left\|\sum_n w_n \mathbf{x}^{(n)}\right\|_2^2 = \sum_i \left[\sum_n w_n \mathbf{x}^{(n)}\right]_i^2 = \sum_i \left(\sum_n w_n x_i^{(n)}\right)^2$$
$$\leq \sum_i \sum_n w_n (x_i^{(n)})^2 = \sum_n w_n \sum_i (x_i^{(n)})^2 = \sum_n w_n \|\mathbf{x}^{(n)}\|_2^2. \tag{34}$$

■ *Cosine similarity*: Given the relative distance between $\mathbf{x}$ and $\mathbf{y}$ as $\|\mathbf{x} - \mathbf{y}\|_2 / \|\mathbf{x}\|_2 = r$ (for simplicity, let $r \ll 1$) then the cosine similarity between the vectors are evaluated as follows,

$$\text{cossim}(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \frac{\|\mathbf{x}\|_2}{\|\mathbf{y}\|_2} \cdot \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\|_2^2} = \frac{1}{2}\frac{\|\mathbf{x}\|_2}{\|\mathbf{y}\|_2} \cdot \frac{\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 - \|\mathbf{x} - \mathbf{y}\|_2^2}{\|\mathbf{x}\|_2^2}$$
$$= \frac{1}{2}\frac{\|\mathbf{x}\|_2}{\|\mathbf{y}\|_2} \cdot \left(1 + \frac{\|\mathbf{y}\|_2^2}{\|\mathbf{x}\|_2^2} - r^2\right) = \frac{1}{2}\left(\frac{\|\mathbf{x}\|_2}{\|\mathbf{y}\|_2} + \frac{\|\mathbf{y}\|_2}{\|\mathbf{x}\|_2} - \frac{\|\mathbf{x}\|_2}{\|\mathbf{y}\|_2} r^2\right). \tag{35}$$

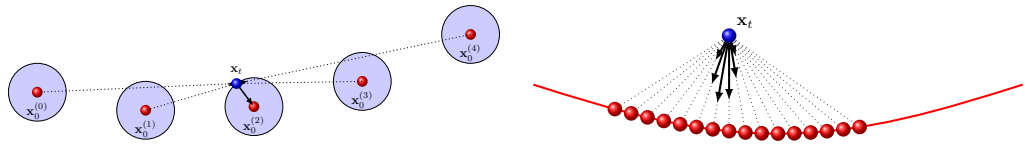
Noting that the triangle inequality implies that the L_2 norm of $\mathbf{y}$ is bounded as follows,

$$(1 - r)\|\mathbf{x}\|_2 \leq \|\mathbf{y}\|_2 \leq (1 + r)\|\mathbf{x}\|_2, \tag{36}$$

we can put $\|\mathbf{y}\|_2 / \|\mathbf{x}\|_2 = 1 + \delta$, $(0 < |\delta| < r \ll 1)$, and can evaluate the cosine similarity as follows,

$$\text{cossim}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\left(\frac{1}{1+\delta} + (1+\delta) - \frac{1}{1+\delta} r^2\right)$$
$$\approx \frac{1}{2}\left((1-\delta) + (1+\delta) - (1-\delta) r^2\right) > 1 - \frac{1}{2}(1+r) r^2. \tag{37}$$

[1]Given Gaussian random variables $X_i \sim \mathcal{N}(0, 1)$, then the squared sum of them $\sum_{i=1}^d X_i^2$ follows the χ^2-distribution with d degrees of freedom. It is well known that χ^2 distribution converges to $\mathcal{N}(d, 2d)$ as d increases. Therefore, If d is sufficiently large, $\|\mathbf{x}\|_2 \coloneqq (\sum_{i=1}^d X_i^2)^{1/2} \approx \sqrt{d \pm \sqrt{2d}} \approx \sqrt{d} \pm \sqrt{1/2}$. Thus the L_2 norm of d-dim Gaussian variable is approximated by $\sqrt{d}$, and the error is as small as the scale of $\sqrt{1/2}$. It implies that Gaussian distribution looks like a sphere in high-dimensional space, contrary to our low-dimensional intuition. Indeed, the following inequality holds (Laurent & Massart, 2000, Lem. 1): $p((1 - 2\sqrt{y/d})^{1/2} \leq \|\mathbf{x}\|_2 / \sqrt{d} \leq (1 + 2\sqrt{y/d} + 2y/d)^{1/2}) \geq 1 - 2e^{-y}$.



(a) Discrete case (b) Continuous case

Figure 6: Intuitive reasons why a single point approximation is valid. The case where $1 - \nu_t \approx 1$

A.2 PHASE (2): WINNER TAKES ALL ($\bar{\alpha}_t = 1 - \nu_t \gg 0$)

The above bounds suggest that the single point approximation is valid only when the noise level ν_t is high ($\nu_t \approx 1$). However, we can also show that the approximation is also valid because of another reason when the noise level ν_t is low ($\nu_t \approx 0$).

If $p(\mathbf{x}_0)$ is a discrete distribution, and if $\nu_t \approx 0$, the weight factor $q(\mathbf{x}_0^{(i)} \mid \mathbf{x}_t)$ in eq. (30) is almost certainly a "one-hot vector" such that

$$q(\mathbf{x}_0^{(i)} \mid \mathbf{x}_t) \approx 1, \quad \text{and} \quad q(\mathbf{x}_0^{(j)} \mid \mathbf{x}_t) \approx 0 \ (j \neq i). \tag{38}$$

That is, the entropy of $q(\mathbf{x}_0 \mid \mathbf{x}_t)$ is nearly zero (see Figure 9). It is almost certain (prob ≈ 1), since the converse is quite rare; i.e., there exists another data point $\mathbf{x}_0^{(j)}$ near $\mathbf{x}_t \sim p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(j)}, 0)$ in such a situation[2]. Figure 6a intuitively shows the reason.

If $p(\mathbf{x}_0)$ is a continuous distribution, very small area (almost a single point) of the data manifold contributes to the integration eq. (30). Figure 6b intuitively shows the situation. The weight function will become nearly the delta function, and the integration will be almost the same as the single point approximation,

$$q(\mathbf{x}_0 \mid \mathbf{x}_t) \approx \delta(\mathbf{x}_0 - \mathbf{x}_0^*). \tag{39}$$

where $\mathbf{x}_0^*$ is a certain point close to the perpendicular projection of $\mathbf{x}_t$ on the data manifold. These are related to the well-known formulae $\varepsilon \log \sum \exp(\cdot/\varepsilon)$ and $\varepsilon \log \int \exp(\cdot/\varepsilon)$ goes to $\max(\cdot)$ when $\varepsilon \to 0$. Naturally, this is more likely to be true for high-dimensional space and is expected to break down for low-dimensional toy data.

Now let us elaborate the above discussion. As the scale we are interested in now is very small, the data manifold is approximated as a flat k-dim space. In addition, as $q(\cdot \mid \mathbf{x}_t)$ is a Gaussian function, it decays rapidly with distance from the center (perpendicular projection of $\mathbf{x}_t \in \mathbb{R}^d$ to the k-dim

[2]Let us evaluate the probability more qualitatively using a toy model. Let us denote a k-dim ball of radius r by $B^k(r)$, and let D_i be the L_2 distance between $\mathbf{x}_t$ and $\sqrt{1-\nu_t}\mathbf{x}_0^{(i)}$, i.e., $D_i := \|\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0^{(i)}\|_2 \approx \sqrt{\nu_t d}$. The question is whether there exists another data point $\mathbf{x}_0^{(j)} (j \neq i)$ in the discrete data distribution $\mathcal{D} = \{\mathbf{x}_0^{(i)}\}_i$ such that $D_j < D_i$. In other words, whether there exists another data point in $B^d(\sqrt{\nu_t d})$ centered at $\sqrt{1-\nu_t}\mathbf{x}_0^{(i)}$. If it is the case, $q(\mathbf{x}_0^{(j)} \mid \mathbf{x}_t)$ has significantly large value compared to $q(\mathbf{x}_0^{(i)} \mid \mathbf{x}_t)$, and it will have strong effects on the result of integration eq. (30). Now let us show that it is unlikely in a high-dimensional data space. As it is difficult evaluating the probability for general cases, let us consider a toy model that the all the n points are uniformly distributed in a k-dim ball ($k \ll d$). That is, we assume that the data manifold is $\mathcal{M} = B^k(\sqrt{k})$ which satisfies $\mathcal{D} \subset \mathcal{M} \subset \mathbb{R}^d$. Then, the probability we are interested in is roughly evaluated as

$$p(\exists j \neq i, D_j < D_i) \approx 1 - \left(1 - \frac{\mathrm{vol}[B^k(\sqrt{\nu_t k})]}{\mathrm{vol}[\sqrt{1-\nu_t}\mathcal{M}]}\right)^n = 1 - \left(1 - \frac{\mathrm{vol}[B^k(\sqrt{\nu_t k})]}{\mathrm{vol}[B^k(\sqrt{(1-\nu_t)k})]}\right)^n.$$

Remembering that the volume of k-dim ball is given by $\mathrm{vol}[B^k(r)] = \frac{\pi^{k/2}}{\Gamma(k/2+1)} r^k$, the probability is approximated as follows,

$$\text{r.h.s} = 1 - \left(1 - \left(\frac{\nu_t k}{(1-\nu_t)k}\right)^{k/2}\right)^n \approx n \left(\frac{\nu_t}{1-\nu_t}\right)^{k/2}, \quad \text{if } \nu_t \approx 0, k \gg 1.$$

Thus, when ν_t is sufficiently small and the data manifold is sufficiently high-dimensional, it is almost unlikely that there exists such a point $\mathbf{x}_0^{(j)}$ $(j \neq i)$ unless n is very large.

subspace). For this reason, only a small ball around the perpendicular foot of $\mathbf{x}_t$ contributes to the integration. As the majority of k-dim Gaussian variables of variance $\sigma^2\mathbf{I}$ are distributed on a sphere of radius $\sqrt{k}\sigma$, it would be sufficient to consider a ball of radius $(\sqrt{k}+2)\sqrt{\nu_t}$.

Now the integration eq. (30) is approximated as follows,

$$\text{eq. (30)} \approx \int_{\mathcal{B}} [\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0) q(\mathbf{x}_0 \mid \mathbf{x}_t) p(\mathbf{x}_0)]\, d\mathbf{x}_0,$$
$$\text{where } \mathcal{B} = \{\mathbf{x}_0 \mid \|\sqrt{1-\nu_t}\mathbf{x}_0 - \mathbf{x}_t\|_2 < (\sqrt{k}+2)\sqrt{\nu_t}\}. \quad (40)$$

As $\mathcal{B}$ is small, the gradient vector $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$ is almost constant in this region. Quantitatively, let $\mathbf{x}_0^{(1)}, \mathbf{x}_0^{(2)} \in \mathcal{B}$, then the distance between these points is evidently bounded above by $\|\mathbf{x}_0^{(1)} - \mathbf{x}_0^{(2)}\|_2 \leq 2(\sqrt{k}+2)\sqrt{\nu_t}$, which implies

$$\begin{aligned}
&\left\| \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(1)}, 0) - \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(2)}, 0) \right\|_2 \\
=&\left\| \frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0^{(1)}}{\nu_t} - \frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0^{(2)}}{\nu_t} \right\|_2 \\
=&\left\| \frac{\sqrt{1-\nu}(\mathbf{x}_0^{(1)} - \mathbf{x}_0^{(2)})}{\nu_t} \right\|_2 \leq \frac{\sqrt{1-\nu_t}}{\nu_t} \cdot 2(\sqrt{k}+2)\sqrt{\nu_t} \\
=&2(\sqrt{k}+2)\sqrt{\frac{1-\nu_t}{\nu_t}}. \qquad (41)
\end{aligned}$$

Noting that $\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)\|_2 \approx \sqrt{d/\nu_t}$, the relative L_2 error between two gradients is bounded above by

$$\frac{\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(1)}, 0) - \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(2)}, 0)\|_2}{\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(2)}, 0)\|_2} \lessapprox 2\frac{(\sqrt{k}+2)\sqrt{1-\nu_t}}{\sqrt{d}} \lessapprox 2\sqrt{\frac{k}{d}}. \quad (42)$$

By integrating both sides [3] w.r.t. $\mathbf{x}_0^{(1)}$, i.e. $\int(\cdot)q(\mathbf{x}_0^{(1)} \mid \mathbf{x}_t)p(\mathbf{x}_0^{(1)})d\mathbf{x}_0^{(1)}$, we obtain the following inequality for any point $\mathbf{x}_0^* \in \mathcal{B}$,

$$\frac{\|\nabla \log p(\mathbf{x}_t, t) - \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^*, 0)\|_2}{\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^*, 0)\|_2} \lessapprox 2\sqrt{\frac{k}{d}}. \quad (43)$$

Now, let us recall that a noised data $\mathbf{x}_t \sim p(\mathbf{x}_t, t)$ is generated in the following procedure.

1. Draw a data $\mathbf{x}_0^{(i)}$ from $p(\mathbf{x}_0, 0)$.
2. Draw a Gaussian $\mathbf{x}_t \sim p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0) = \mathcal{N}(\sqrt{1-\nu_t}\mathbf{x}_0^{(i)}, \nu_t \mathbf{I}_d)$.

Here, the data point $\mathbf{x}_0^{(i)} \in \mathcal{M}$ is in the ball $\mathcal{B}$, because the distance between $\sqrt{1-\nu_t}\mathbf{x}_0^{(i)}$ and the perpendicular foot of $\mathbf{x}_t$ is about $\sqrt{\nu_t k}$, which is clearly smaller than $(\sqrt{k}+2)\sqrt{\nu_t}$. Thus we can write the following inequality.

$$\frac{\|\nabla \log p(\mathbf{x}_t, t) - \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)\|_2}{\|\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0)\|_2} \lessapprox 2\sqrt{\frac{k}{d}}, \quad (44)$$

and we may conclude that the following approximation is largely valid if $\nu_t \approx 0, k \ll d$,

$$\nabla \log p(\mathbf{x}_t, t) \approx \nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0). \quad (45)$$

[3] We used the following relation: $\|\mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] - \mathbf{y}\|_2 = \|\mathbb{E}_{\mathbf{x}}[f(\mathbf{x}) - \mathbf{y}]\|_2 \leq \mathbb{E}_{\mathbf{x}}[\|f(\mathbf{x}) - \mathbf{y}\|_2]$.

A.3 PLOTS USING REAL DATA

Let us empirically validate that single point approximation using real data, MNIST (LeCun et al., 2010) 60,000 samples and CIFAR-10 (Krizhevsky, 2009) 50,000 samples. In specific, we evaluated following three metrics.

- Relative L_2 error between $\nabla \log p(\mathbf{x}_t, t)$ and $\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$,
- Cosine similarity between $\nabla \log p(\mathbf{x}_t, t)$ and $\nabla \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$,
- Entropy of $q(\mathbf{x}_0 \mid \mathbf{x}_t)$.

Figure 7 shows the relative L_2 distance, for both datasets. Figure 8 similarly show the distribution (random 3000 trials) of the cosine similarity, and Figure 9 shows the entropy. Dashed curves indicate the bounds evaluated in eq. (32) and eq. (33).

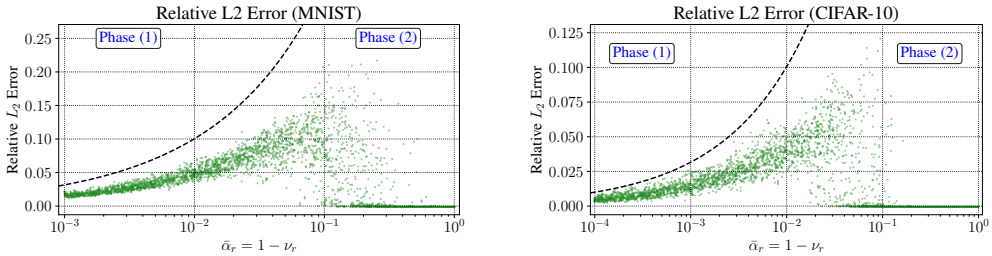
These figures show that the range of intermediate region between Phase (1) and Phase (2) will not have impact in practical situations since we do not evaluate the neural network $\mathbf{S}_\theta(\cdot, \cdot)$ in this range so many times (i.e., $\bar{\alpha}_t \sim 10^{-3}$ to $10^{-1} \Leftrightarrow \nu_t \sim 0.999$ to 0.9). Moreover, the approximation accuracy is still very high even in this region. Furthermore, although MNIST and CIFAR-10 are quite "low-dimensional" for real-world images, approximations are established with such high accuracy. Therefore, it is expected to be established with higher accuracy for more realistic images.

A.4 SUMMARY OF THIS SECTION

Thus, we can assume that the single point approximation almost always holds practically.

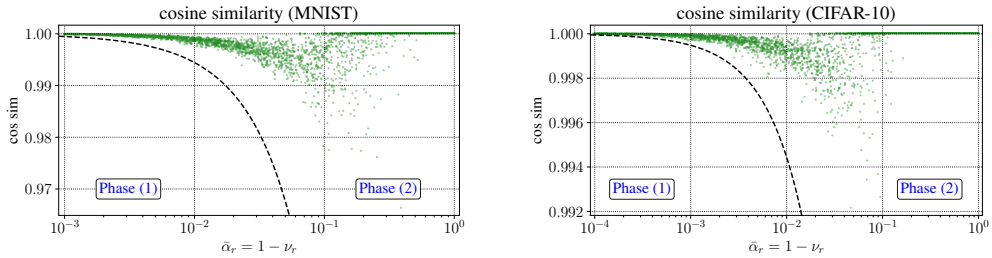
$$-\frac{\mathbf{S}_\theta(\mathbf{x}_t, t)}{\sqrt{\nu_t}} \overset{\text{model}}{\approx} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t) \overset{\text{almost equal}}{\approx} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0^{(i)}, 0) = -\frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0^{(i)}}{\nu_t}.$$

Therefore, we may also expect that the similar approximation will be valid for their derivatives. Of course, such an expectation is mathematically incorrect; for example, let $g(x) = f(x) + \varepsilon \sin \omega x$, then the difference $g(x) - f(x) = \varepsilon \sin \omega x$ goes to zero as $\varepsilon \to 0$, but the difference of derivatives $g'(x) - f'(x) = \varepsilon\omega \cos \omega x$ does not if $\omega \to \infty$ faster than $1/\varepsilon$. In other words, the single point approximation does not necessarily imply the ideal derivative approximation, and if it is to be mathematically rigorous, it must be supported by other nontrivial knowledge on the data manifold. Conversely, if the approximation holds for the derivative, it would provide indirect evidence that such nontrivial logic holds that the derivative of error term is small in the region of interest, and would provide a better understanding of the fundamental properties of diffusion generative models. This nontrivial leap is the most important "conjecture" made in this paper and should be examined more closely in the future.



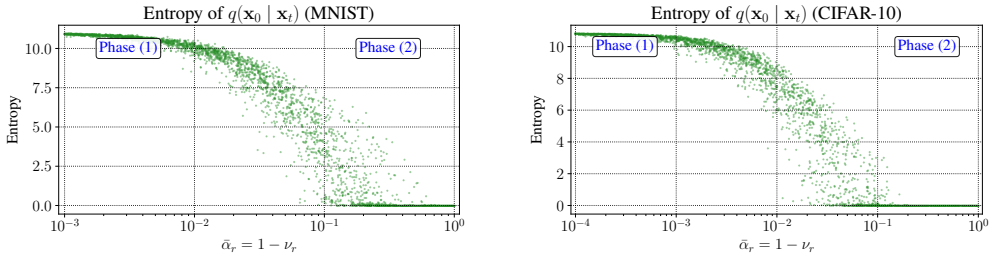
(a) MNIST training data. 784-dim ($= 28 \times 28$) (b) CIFAR10 data. 3072-dim ($= 32 \times 32 \times 3$)

Figure 7: Relative L_2 distance between the true gradient and the single point approximation depending on the noise level ν_t. Dashed curves indicate the upper bounds evaluated in eq. (32).



(a) MNIST training data. 784-dim ($= 28 \times 28$) (b) CIFAR10 data. 3072-dim ($= 32 \times 32 \times 3$)

Figure 8: Cosine similarity between the true gradient and the single point approximation depending on the noise level ν_t. Dashed curves indicate the lower bounds evaluated in eq. (33).



(a) MNIST training data. 784-dim ($= 28 \times 28$) (b) CIFAR10 data. 3072-dim ($= 32 \times 32 \times 3$)

Figure 9: Entropy of $q(\mathbf{x}_0 \mid \mathbf{x}_t)$ depending on the noise level ν_t.

B DERIVATION OF THE "IDEAL DERIVATIVES"

Because of the discussion in § A, the true score function $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t)$ is finely approximated by a single point approximation $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t, t \mid \mathbf{x}_0, 0)$. Now we may also assume that the derivatives of both will also be close.

In this paper, we are interested in the Taylor expansion of the following form (see also § D.1.1),

$$\boldsymbol{\psi}(\mathbf{x}_h, h) = \boldsymbol{\psi}(\mathbf{x}_0, 0) + \sum_{k=1}^{\infty} \frac{h^k}{k!} \left(\partial_t + \mathbf{a}(\mathbf{x}_t, t) \cdot \nabla_{\mathbf{x}_t}\right)^k \boldsymbol{\psi}(\mathbf{x}_t, t)\Big|_{t=0}. \tag{46}$$

If the function $\boldsymbol{\psi}(\mathbf{x}_t, t)$ is separable in each dimension (i.e., $\partial_{x_i} \psi_j = 0$ for $i \neq j$), the following relation holds,

$$\left(\mathbf{a}(\mathbf{x}_t, t) \cdot \nabla_{\mathbf{x}_t}\right) \boldsymbol{\psi}(\mathbf{x}_t, t) = \mathbf{a}(\mathbf{x}_t, t) \odot \nabla_{\mathbf{x}_t} \odot \boldsymbol{\psi}(\mathbf{x}_t, t), \tag{47}$$

where $\odot$ is the element-wise product or operation. If $\mathbf{a}(\mathbf{x}_t, t)$ is also separable in each dimension[4] the Taylor series is formally rewritten as follows,

$$\boldsymbol{\psi}(\mathbf{x}_t, t) = \boldsymbol{\psi}(\mathbf{x}_0, 0) + \sum_{k=1}^{\infty} \frac{t^k}{k!} \left(\mathbf{1}\partial_t + \mathbf{a}(\mathbf{x}_t, t) \odot \partial_{\mathbf{x}_t}\right)^k \boldsymbol{\psi}(\mathbf{x}_t, t)\Big|_{t=0} \tag{48}$$

where $\partial_{\mathbf{x}_t} \coloneqq \nabla_{\mathbf{x}_t} \odot$ is the element-wise derivative operator. This is formally the same as the 1-dim Taylor series. Therefore, it is sufficient to consider the 1-dim Taylor series first, and parallelize each dimension later. Thus the derivatives we actually need are the following two.

$$\partial_{\mathbf{x}_t} \mathbf{S}_\theta(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \odot \mathbf{S}_\theta(\mathbf{x}_t, t), \quad \partial_t \mathbf{S}_\theta(\mathbf{x}_t, t) = (\mathbf{1}\partial_t) \odot \mathbf{S}_\theta(\mathbf{x}_t, t). \tag{49}$$

B.1 SPATIAL DERIVATIVE $\partial_{\mathbf{x}_t} \mathbf{S}_\theta(\mathbf{x}_t, t) \coloneqq \nabla_{\mathbf{x}_t} \odot \mathbf{S}_\theta(\mathbf{x}_t, t)$

Let us first compute the spatial derivative $(\mathbf{a} \cdot \nabla_{\mathbf{x}_t}) \mathbf{S}_\theta(\mathbf{x}_t, t)$.

$$\begin{aligned}
(\mathbf{a} \cdot \nabla_{\mathbf{x}_t}) \mathbf{S}_\theta(\mathbf{x}_t, t) &= \left(\sum_i a_i \partial_{{x_t}^i}\right) \mathbf{S}_\theta(\mathbf{x}_t, t) \approx \left(\sum_i a_i \partial_{{x_t}^i}\right) \frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\sqrt{\nu_t}} \\
&= \frac{1}{\sqrt{\nu_t}} \begin{bmatrix} \left(\sum_i a_i \partial_{{x_t}^i}\right) (\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0)^1 \\ \vdots \\ \left(\sum_i a_i \partial_{{x_t}^i}\right) (\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0)^d \end{bmatrix} \\
&= \frac{1}{\sqrt{\nu_t}} \begin{bmatrix} \left(\sum_i a_i \partial_{{x_t}^i}\right) ({x_t}^1 - \sqrt{1-\nu_t}{x_0}^1) \\ \vdots \\ \left(\sum_i a_i \partial_{{x_t}^i}\right) ({x_t}^d - \sqrt{1-\nu_t}{x_0}^d) \end{bmatrix} \\
&= \frac{1}{\sqrt{\nu_t}} \begin{bmatrix} \left(a_1 \partial_{{x_t}^1}\right) ({x_t}^1 - \sqrt{1-\nu_t}{x_0}^1) \\ \vdots \\ \left(a_d \partial_{{x_t}^d}\right) ({x_t}^d - \sqrt{1-\nu_t}{x_0}^d) \end{bmatrix} \\
&= \frac{1}{\sqrt{\nu_t}} \begin{bmatrix} a_1 \\ \vdots \\ a_d \end{bmatrix} = \frac{1}{\sqrt{\nu_t}} \mathbf{a} = \mathbf{a} \odot \frac{1}{\sqrt{\nu_t}} \mathbf{1}.
\end{aligned} \tag{50}$$

[4]In general, $(\mathbf{a} \cdot \nabla)^2 = (\sum_i a_i \partial_i)^2 = (\sum_i a_i \partial_i)(\sum_j a_j \partial_j) = \sum_i a_i \sum_j (\partial_i a_j + a_j \partial_i \partial_j)$. If $\mathbf{a}$ is separable in each dimension, the $\partial_i a_j (i \neq j)$ terms vanish, and $(\mathbf{a} \cdot \nabla)^2 = \sum_i (a_i \partial_i a_i + \sum_j a_i a_j \partial_i \partial_j)$. If the function $\boldsymbol{\psi}(\mathbf{x}_t, t)$ is separable in each dimension, then $(\mathbf{a} \cdot \nabla)^2 \psi_k = \sum_i (a_i \partial_i a_i + \sum_j a_i a_j \partial_i \partial_j) \psi_k = (a_k \partial_k a_k + a_k^2 \partial_k^2) \psi_k$. Thus we can formally write $(\mathbf{a} \cdot \nabla)^2 \boldsymbol{\psi} = (\mathbf{a} \odot \nabla \odot \mathbf{a} + \mathbf{a} \odot \mathbf{a} \odot \nabla \odot \nabla) \odot \boldsymbol{\psi} = \mathbf{a} \odot (\nabla \odot \mathbf{a} + \mathbf{a} \odot \nabla \odot \nabla) \odot \boldsymbol{\psi} = \mathbf{a} \odot \nabla \odot (\mathbf{a} \odot \nabla) \odot \boldsymbol{\psi} = (\mathbf{a} \odot \nabla \odot)^2 \boldsymbol{\psi} = (\mathbf{a} \odot \partial_{\mathbf{x}})^2 \boldsymbol{\psi}$. (Note that the operator $(\mathbf{a} \cdot \nabla)$ is scalar while $(\mathbf{a} \odot \partial_{\mathbf{x}})$ is d-dim vector.) We can similarly show $(\mathbf{a} \cdot \nabla)^k \boldsymbol{\psi} = (\mathbf{a} \odot \partial_{\mathbf{x}})^k \boldsymbol{\psi}$ for $k \geq 3$.

(Here, we used the notation ${x_t}^i$ to denotes the i-th component of a vector $\mathbf{x}_t$.) Thus we have formally obtained the relation

$$\mathbf{a} \odot (\partial_{\mathbf{x}_t} \mathbf{S}_\theta(\mathbf{x}_t, t)) = \mathbf{a} \odot \frac{1}{\sqrt{\nu_t}} \mathbf{1}, \tag{51}$$

and therefore, we may formally write

$$\partial_{\mathbf{x}_t} \mathbf{S}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\nu_t}} \mathbf{1}. \tag{52}$$

B.2 Time Derivative $-\partial_t \mathbf{S}_\theta(\mathbf{x}_t, t)$

Next, let us compute $-\partial_t \mathbf{S}_\theta(\mathbf{x}_t, t)$. During the computation, $\mathbf{x}_0$ is replaced by the relation

$$\mathbf{x}_0 = \frac{1}{\sqrt{1-\nu_t}} \left(\mathbf{x}_t - \sqrt{\nu_t} \mathbf{S}_\theta(\mathbf{x}_t, t) \right). \tag{53}$$

We also use the following relations between ν_t, β_t, which is immediately obtained from the definition of ν_t,

$$\dot{\nu}_t = (1-\nu_t)\beta(t). \tag{54}$$

Using the above information, we may compute $-\partial_t \mathbf{S}_\theta(\mathbf{x}_t, t)$ as follows.

$$\begin{aligned}
-\partial_t \mathbf{S}_\theta(\mathbf{x}_t, t) &\approx -\partial_t \frac{\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0}{\sqrt{\nu_t}} \\
&= -\frac{1}{\sqrt{\nu_t}} \left(\frac{1}{2} \dot{\nu}_t (1-\nu_t)^{-1/2} \mathbf{x}_0 \right) - \left(\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0 \right) \left(-\frac{1}{2} \dot{\nu}_t \nu_t^{-3/2} \right) \\
&= -\frac{\dot{\nu}_t}{2\nu_t^{3/2}} \left(\frac{\nu_t}{\sqrt{1-\nu_t}} \mathbf{x}_0 - \left(\mathbf{x}_t - \sqrt{1-\nu_t}\mathbf{x}_0 \right) \right) \\
&= -\frac{\dot{\nu}_t}{2\nu_t^{3/2}} \left(-\mathbf{x}_t + \frac{1}{\sqrt{1-\nu_t}} \mathbf{x}_0 \right) \\
&= -\frac{\dot{\nu}_t}{2\nu_t^{3/2}} \left(-\mathbf{x}_t + \frac{1}{\sqrt{1-\nu_t}} \frac{1}{\sqrt{1-\nu_t}} \left(\mathbf{x}_t - \sqrt{\nu_t} \mathbf{S}_\theta(\mathbf{x}_t, t) \right) \right) \\
&= -\frac{\dot{\nu}_t}{2\nu_t^{3/2}} \left(\left(-1 + \frac{1}{1-\nu_t} \right) \mathbf{x}_t - \frac{\sqrt{\nu_t}}{1-\nu_t} \mathbf{S}_\theta(\mathbf{x}_t, t) \right) \\
&= -\frac{1}{2\nu_t^{3/2}} \frac{\dot{\nu}_t}{1-\nu_t} \left(\nu_t \mathbf{x}_t - \sqrt{\nu_t} \mathbf{S}_\theta(\mathbf{x}_t, t) \right) \\
&= -\frac{1}{2\nu_t^{3/2}} \beta(t) \left(\nu_t \mathbf{x}_t - \sqrt{\nu_t} \mathbf{S}_\theta(\mathbf{x}_t, t) \right) \\
&= -\frac{\beta(t)}{2\sqrt{\nu_t}} \left(\mathbf{x}_t - \frac{\mathbf{S}_\theta(\mathbf{x}_t, t)}{\sqrt{\nu_t}} \right).
\end{aligned} \tag{55}$$

The “derivatives" have some good points. For example, the partial derivatives commute,

$$\partial_{\mathbf{x}_t} \partial_t \mathbf{S}_\theta(\mathbf{x}_t, t) = \partial_t \partial_{\mathbf{x}_t} \mathbf{S}_\theta(\mathbf{x}_t, t). \tag{56}$$

B.3 THE DERIVATIVES $L_\flat(-\bar{\mathbf{f}}_\flat)$, $L_\sharp(-\bar{\mathbf{f}}_\sharp)$, $L_\sharp(g)$, $G_\sharp(-\bar{\mathbf{f}}_\sharp)$, $G_\sharp(g)$

The computation of the derivative $L_\flat(-\bar{\mathbf{f}}_\flat)$, $L_\sharp(-\bar{\mathbf{f}}_\sharp)$, $L_\sharp(g)$, $G_\sharp(-\bar{\mathbf{f}}_\sharp)$, $G_\sharp(g)$ does not require any particular nontrivial process. All we have to do is rewrite a term every time we encounter a derivative of $\mathbf{S}_\theta(x_t, t)$ or ν_t, and the rest is at the level of elementary exercises in introductory calculus. To execute this symbolic computation, the use of computer algebra systems will be a good option. It should be noted, however, that some implementation tricks to process such custom derivatives are required (in other words, the term-rewriting system should be customized).

The results are shown below. Although these expressions appear complex at first glance, the code generation system can automatically generate code for such expressions.

$$L_\flat(-\bar{\mathbf{f}}_\flat)(x_t, t) = \left(\frac{\beta_t^2}{4} - \frac{\dot{\beta}_t}{2}\right) x_t + \left(\frac{\dot{\beta}_t}{2\sqrt{\nu_t}} - \frac{\beta_t^2}{4\nu_t^{3/2}}\right) \mathbf{S}_\theta(x_t, t) \tag{57}$$

$$L_\sharp(-\bar{\mathbf{f}}_\sharp)(x_t, t) = \left(\frac{\beta_t^2}{4} - \frac{\dot{\beta}_t}{2}\right) x_t + \frac{\dot{\beta}_t}{\sqrt{\nu_t}} \mathbf{S}_\theta(x_t, t) \tag{58}$$

$$G_\sharp(-\bar{\mathbf{f}}_\sharp)(x_t, t) = \left(\frac{1}{2} - \frac{1}{\nu_t}\right) \beta_t^{3/2} \tag{59}$$

$$L_\sharp g(t) = -\frac{\dot{\beta}_t}{2\sqrt{\beta_t}} \tag{60}$$

$$G_\sharp g(t) = 0. \tag{61}$$

We may also compute higher order derivatives. For example,

$$\begin{aligned} L_\flat L_\flat(-\bar{\mathbf{f}}_\flat)(x_t, t) = &\left(\frac{\beta_t^3}{8} - \frac{3\beta_t\dot{\beta}_t}{4} + \frac{\ddot{\beta}_t}{2}\right) x_t \\ &+ \left(\frac{\beta_t^3(-\nu_t^2 + 3\nu_t - 3)}{8\nu_t^{5/2}} + \frac{3\beta_t\dot{\beta}_t}{4\nu_t^{3/2}} - \frac{\ddot{\beta}_t}{2\sqrt{\nu_t}}\right) \mathbf{S}_\theta(x_t, t) \end{aligned} \tag{62}$$

$$L_\sharp L_\sharp(-\bar{\mathbf{f}}_\sharp)(x_t, t) = \left(\frac{\beta_t^3}{8} - \frac{3\beta_t\dot{\beta}_t}{4} + \frac{\ddot{\beta}_t}{2}\right) x_t - \frac{\beta_t^3 + 4\ddot{\beta}_t}{4\sqrt{\nu_t}} \mathbf{S}_\theta(x_t, t) \tag{63}$$

$$L_\sharp G_\sharp(-\bar{\mathbf{f}}_\sharp)(x_t, t) = \frac{\sqrt{\beta_t}}{\nu_t^2}\left(\frac{\nu_t(2\beta_t^2 + 3\dot{\beta}_t)}{2} - \beta_t^2 - \frac{3\nu_t^2\dot{\beta}_t}{4}\right) \tag{64}$$

$$G_\sharp L_\sharp(-\bar{\mathbf{f}}_\sharp)(x_t, t) = \sqrt{\beta_t}\left(\frac{\beta_t^2}{4} - \frac{\dot{\beta}_t}{2} + \frac{\dot{\beta}_t}{\nu_t}\right) \tag{65}$$

$$G_\sharp G_\sharp(-\bar{\mathbf{f}}_\sharp)(x_t, t) = 0 \tag{66}$$

$$L_\sharp L_\sharp g(t) = \frac{2\beta_t\ddot{\beta}_t - \dot{\beta}_t^2}{4\beta_t^{3/2}} \tag{67}$$

$$L_\sharp G_\sharp g(t) = 0 \tag{68}$$

$$G_\sharp L_\sharp g(t) = 0 \tag{69}$$

$$G_\sharp G_\sharp g(t) = 0. \tag{70}$$

As we can see, no factors other than integers, x_t, $\mathbf{S}_\theta(x_t, t)$, ν_t, β_t and derivatives of β_t appear. This is also true for higher order derivatives, which can be easily shown.

B.4 SymPy Code Snippet for Automatic Symbolic Computation of Derivatives

The following code snippet is a minimalistic example of SymPy code to compute the above derivatives using the customized derivative method. We used SymPy 1.11 to test the following code snippet.

```
from sympy import Function, symbols, sqrt, simplify

x, t = symbols('x t') # x,t
B = Function('beta') # β_t

# define customized derivatives of ν_t
class nu(Function):
    def fdiff(self, argindex=1):
        t, = self.args
        return (1 - nu(t)) * B(t) # ν̇_t = (1 − ν_t)β_t

# define customized derivatives of S_θ(x,t)
class S_theta(Function):
    def fdiff(self, argindex=1):
        x, t = self.args
        if argindex == 1: # ∂/∂x
            d =  1 / sqrt(nu(t))
        elif argindex == 2: # ∂/∂t
            d = (x - S_theta(x, t)/sqrt(nu(t))) * B(t) / (2 * sqrt(nu(t)))
        return d

# define f̄_♭
class f_flat(Function):
    @classmethod
    def eval(cls, x, t):
        return - B(t) * x / 2 + S_theta(x, t) * B(t) / (2 * sqrt(nu(t)))

# define differential operator L_♭
class L_flat(Function):
    @classmethod
    def eval(cls, fxt):
        return -fxt.diff(t) - f_flat(x, t) * fxt.diff(x)

# show each derivative
print(f_flat(x, t))
print(simplify(L_flat(f_flat(x,t)))) # L_♭ f̄_♭(x_t,t); see eq. (57)
print(simplify(L_flat(L_flat(f_flat(x,t))))) # L_♭ L_♭ f̄_♭(x_t,t); see eq. (62),

# we can similarly define f̄_♯, L_♯, G_♯ and compute other derivatives.
```

The result will look like

$$\text{[Out 1]} \quad -\frac{x\beta(t)}{2} + \frac{S_\theta(x,t)\beta(t)}{2\sqrt{\nu(t)}}$$

$$\text{[Out 2]} \quad -\frac{x\beta^2(t)}{4} + \frac{x\frac{d}{dt}\beta(t)}{2} + \frac{S_\theta(x,t)\beta^2(t)}{4\nu^{\frac{3}{2}}(t)} - \frac{S_\theta(x,t)\frac{d}{dt}\beta(t)}{2\sqrt{\nu(t)}}$$

$$\text{[Out 3]} \quad -\frac{x\beta^3(t)}{8} + \frac{3x\beta(t)\frac{d}{dt}\beta(t)}{4} - \frac{x\frac{d^2}{dt^2}\beta(t)}{2} + \frac{S_\theta(x,t)\beta^3(t)}{8\sqrt{\nu(t)}}$$
$$- \frac{3S_\theta(x,t)\beta^3(t)}{8\nu^{\frac{3}{2}}(t)} + \frac{3S_\theta(x,t)\beta^3(t)}{8\nu^{\frac{5}{2}}(t)} - \frac{3S_\theta(x,t)\beta(t)\frac{d}{dt}\beta(t)}{4\nu^{\frac{3}{2}}(t)} + \frac{S_\theta(x,t)\frac{d^2}{dt^2}\beta(t)}{2\sqrt{\nu(t)}}$$

and so on. Some additional coding techniques can further improve the readability of these expressions, but there will be no need to go any deeper into such subsidiary issues here.

Thus obtained symbolic expressions can be automatically converted into executable code in practical programming languages including Python and C++ using a code generator, though the authors hand-coded the obtained expressions in Python for the experiments in this paper.

B.5 DDIM IS AN IDEAL-DERIVATIVE-BASED SAMPLER

Using SymPy, we can easily compute the Taylor expansion of a given function. For example, the following code

```
sympy.series(B(t+h), h, 0, 4)
```

yields the result like

$$\beta(t) + h \left.\frac{d}{d\xi_1}\beta(\xi_1)\right|_{\xi_1=t} + \frac{h^2 \left.\frac{d^2}{d\xi_1^2}\beta(\xi_1)\right|_{\xi_1=t}}{2} + \frac{h^3 \left.\frac{d^3}{d\xi_1^3}\beta(\xi_1)\right|_{\xi_1=t}}{6} + O\left(h^4\right).$$

Similarly, using the relation $\dot{\nu}_t = (1-\nu_t)\beta_t$, we can easily compute the Taylor expansion of ν_{t-h} as follows.

```
sympy.series(nu(t-h), h, 0, 3)
```

$$\nu_{t-h} = \nu(t) + h\left(\beta(t)\nu(t) - \beta(t)\right) + h^2\left(\frac{\beta^2(t)\nu(t)}{2} - \frac{\beta^2(t)}{2} - \frac{\nu(t)\left.\frac{d}{d\xi_1}\beta(\xi_1)\right|_{\xi_1=t}}{2} + \frac{\left.\frac{d}{d\xi_1}\beta(\xi_1)\right|_{\xi_1=t}}{2}\right) + O\left(h^3\right)$$

Using this functionality of SymPy, we can easily compute the Taylor expansion of the DDIM (Song et al., 2020a). Let us recall that the DDIM algorithm is given by eq. (15),

$$\text{DDIM:} \quad \mathbf{x}_{t-h} \leftarrow \underbrace{\sqrt{\frac{1-\nu_{t-h}}{1-\nu_t}}}_{=:\rho_{t,h}^{\text{DDIM}}}\mathbf{x}_t + \underbrace{\frac{\sqrt{(1-\nu_t)\nu_{t-h}} - \sqrt{(1-\nu_{t-h})\nu_t}}{\sqrt{1-\nu_t}}}_{=:\mu_{t,h}^{\text{DDIM}}}\mathbf{S}_\theta(\mathbf{x}_t, t).$$

Then using SymPy, the Taylor expansion of $\rho_{t,h}^{\text{DDIM}}$ and $\mu_{t,h}^{\text{DDIM}}$ are computed as follows,

$$\rho_{t,h}^{\text{DDIM}} = 1 + \frac{\beta_t}{2}h - \frac{h^2}{4}\left(\frac{\beta_t^2}{2} - \dot{\beta}_t\right) + \frac{h^3}{4}\left(\frac{\beta_t^3}{12} - \frac{\beta_t\dot{\beta}_t}{2} + \frac{\ddot{\beta}_t}{3}\right) + o(h^3), \tag{71}$$

$$\sqrt{\nu_t}\mu_{t,h}^{\text{DDIM}} = -\frac{\beta_t}{2}h + \frac{h^2}{4}\left(\dot{\beta}_t - \frac{\beta_t^2}{2\nu_t}\right) + \frac{h^3}{4}\left(-\frac{\beta_t^3}{12} + \frac{\beta_t^3}{4\nu_t} - \frac{\beta_t^3}{4\nu_t^2} + \frac{\beta_t\dot{\beta}_t}{2\nu_t} - \frac{\ddot{\beta}_t}{3}\right) + o(h^3). \tag{72}$$

Although it has been known that DDIM corresponds to the Euler method up to 1st order terms (Song et al., 2020a; Salimans & Ho, 2022), this expansion gives better understanding of higher order terms. That is, these are exactly equivalent to our deterministic Quasi-Taylor sampler eq. (23) and eq. (24) up to 3rd-order terms. This fact may suggest that the assumptions behind the DDIM derivation will be logically equivalent to our assumptions of ideal derivatives.

The advantage of the proposed Quasi-Taylor method is that we can decide the hyperparameter at which order the Taylor expansion is truncated. On the other hand, DDIM automatically incorporates terms of much higher order, leaving no room for order tuning.

C ON THE NOISE SCHEDULE

C.1 DERIVATION

The noise scheduling function $\beta_t, \nu_t, \lambda(t)$, defined by

$$\beta_t = \dot{\lambda}(t) \tanh \frac{\lambda(t)}{2}, \quad \nu_t = \tanh^2 \frac{\lambda(t)}{2}, \quad \lambda(t) = \log(1 + Ae^{kt}), \tag{73}$$

may appear to be more complex than necessary, but these were naturally derived in the process of searching for a suitable form to control the factor $\beta_t/\sqrt{\nu_t}$ to scale the output of neural network $\mathbf{S}_\theta(x_t, t)$. In this section, let us derive these functions.

Since this scaling factor $\beta_t/\sqrt{\nu_t}$ plays a fairly important role in the behavior of the algorithm, there is a natural motivation to design this factor itself directly. Let us write it as $\xi(t) = \beta_t/\sqrt{\nu_t}$. As $\nu_t = 1 - e^{-\int_0^t \beta(t')dt'}$ by definition, we immediately have $\beta_t = \dot{\nu}_t/(1-\nu_t)$, and we can rewrite the factor as follows,

$$\xi(t) = \frac{\dot{\nu}_t}{(1-\nu_t)\sqrt{\nu_t}}. \tag{74}$$

If this is regarded as an ODE of ν_t, the solution is given by

$$\nu_t = \tanh^2 \left(\frac{1}{2} \int^t \xi(t')dt' \right). \tag{75}$$

Since the integral contained in $\tanh$ is cumbersome, it will henceforth be written as $\lambda(t)$. From this, the above β_t and ν_t are immediately obtained.

Next, let us derive $\lambda(t)$. Unlike β_t and ν_t, this is not determined by logical necessity, leaving room to design the functional form. As the Picard-Lindelöf theorem states, for an ODE $\dot{x} = a(x,t)$ to have a unique solution the coefficient $a(x,t)$ should be continuous w.r.t. t and Lipschitz continuous w.r.t. x, and otherwise, ODEs often behave less favorably[5]. In PF-ODE, the Lipschitz condition is written as follows,

$$\left| \partial_{x_t} \left(\frac{\beta_t}{2} x_t - \frac{\beta_t}{2\sqrt{\nu_t}} \mathbf{S}_\theta(x_t, t) \right) \right| < \infty. \tag{76}$$

Using the ideal derivative of $\mathbf{S}_\theta(x_t, t)$, this condition translates as $\beta_t |1 - 1/\nu_t| < \infty$. It further simplifies as follows,

$$\begin{aligned} \beta_t \left| 1 - \frac{1}{\nu_t} \right| &= \beta_t \frac{1-\nu_t}{\nu_t} = \frac{\dot{\nu}_t}{\nu_t} = \frac{d}{dt} \log \nu_t \\ &\propto \frac{d}{dt} \log \tanh \frac{\lambda(t)}{2} = \frac{\dot{\lambda}(t)}{\sinh(\lambda(t)/2)\cosh(\lambda(t)/2)} \\ &\propto \frac{\dot{\lambda}(t)}{\sinh \lambda(t)} < \infty. \end{aligned} \tag{77}$$

When the argument is large, $\sinh$ increases exponentially, so it is very easy to make $\dot{\lambda}(t)/\sinh\lambda(t)$ bounded above by a finite constant. To the contrary, when the $\lambda(t)$ is small, $\dot{\lambda}(t)/\sinh\lambda(t)$ is approximated by $\dot{\lambda}(t)/\lambda(t)$. In order to make this bounded above by a finite constant, we have chosen the softplus function as one of the simplest functions. It meets the Lipschitz condition and the requirement that $\nu_0 \approx 0$. However, there may be better alternatives.

[5]For example, the ODE $\dot{x} = x^2, x(0) = 1$ has the solution $x = 1/(1-t)$ when $t < 1$, and it blows up at $t = 1$. It is usually impossible to consider what happens after $t > 1$ in ordinary contexts. Another well-known example is $\dot{x} = \sqrt{x}, x(0) = 0$. It has a solution $x = t^2/4$, but $x \equiv 0$ is also a solution. It actually has infinitely many solutions $x = 0$ (if $t \le t_0$), $x = (t-t_0)^2/4$ (if $t > t_0$), where $t_0 \ge 0$ is an arbitrary constant. In both cases, the coefficient $a(\cdot,\cdot)$ is not Lipschitz continuous. Even these seemingly simplest ODEs behave very complexly unless the coefficients are carefully designed.

C.2 Derivation of A and k

If ν_0 and ν_T are specified, we can explicitly compute A and k from them by solving

$$\begin{cases} \nu_0 = \tanh^2 \frac{1}{2} \log(1 + A) \\ \nu_T = \tanh^2 \frac{1}{2} \log(1 + Ae^{kT}). \end{cases} \tag{78}$$

This simplifies using the following relation,

$$\begin{aligned} & \nu_t = \tanh^2 \frac{1}{2} \log(1 + Ae^{kt}) \\ \Leftrightarrow \quad & \operatorname{arctanh} \sqrt{\nu_t} = \frac{1}{2} \log(1 + Ae^{kt}) \\ \Leftrightarrow \quad & \frac{1}{2} \log \frac{1 + \sqrt{\nu_t}}{1 - \sqrt{\nu_t}} = \frac{1}{2} \log(1 + Ae^{kt}) \\ \Leftrightarrow \quad & \frac{1}{2} \log \left(1 + \frac{2\sqrt{\nu_t}}{1 - \sqrt{\nu_t}} \right) = \frac{1}{2} \log(1 + Ae^{kt}) \\ \Leftrightarrow \quad & \frac{2\sqrt{\nu_t}}{1 - \sqrt{\nu_t}} = Ae^{kt}. \end{aligned} \tag{79}$$

C.3 Plots

Some noise schedules obtained by the above methods are shown in Figure 10.

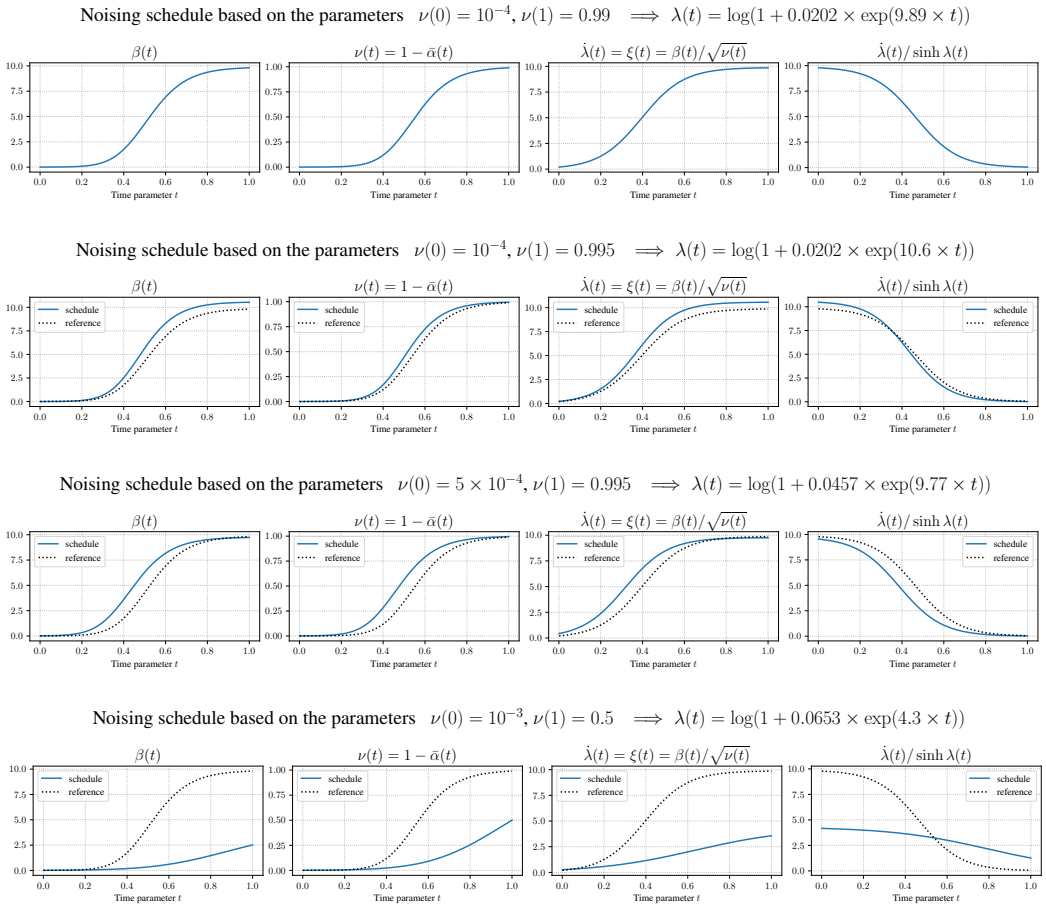


Figure 10: Some noise schedules. The dotted lines are the reference baselines, corresponding to the case where $\nu_0 = 10^{-4}, \nu(1) = 0.99$ (same as the top one).

D SUPPLEMENT ON FUNDAMENTALS

For convenience, let us summarize some basics behind the ideas in this paper. The contents of this section are not particularly novel, but the authors expect that this section will give a better understanding of the ideas of this paper and the continuous-time approach to diffusion generative models.

D.1 TAYLOR EXPANSION AND ITÔ-TAYLOR EXPANSION

D.1.1 TAYLOR EXPANSION OF DETERMINISTIC SYSTEMS

1-dimensional case We first consider a 1-dim deterministic system $\dot{x}(t) = a(x(t), t)$. It is well known that this ordinary differential equation has a unique solution if $a(x, t)$ is Lipschitz continuous w.r.t. x and continuous w.r.t. t (Picard-Lindelöf Theorem). Let $\varphi(x(t), t)$ be a differentiable function. Its total derivative is written as

$$\begin{aligned} d\varphi &= \frac{\partial \varphi}{\partial t} dt + \frac{\partial \varphi}{\partial x} dx = \frac{\partial \varphi}{\partial t} dt + \frac{\partial \varphi}{\partial x} \frac{dx}{dt} dt = \left(\frac{\partial \varphi}{\partial t} + \frac{\partial \varphi}{\partial x} a(x,t) \right) dt \\ &= \underbrace{\left(\frac{\partial}{\partial t} + a(x,t) \frac{\partial}{\partial x} \right)}_{=: L_\flat} \varphi dt. \end{aligned} \tag{80}$$

By integrating both sides from 0 to t, we have

$$\varphi(x(t), t) = \varphi(x(0), 0) + \int_0^t (L_\flat \varphi)(x(s), s) ds. \tag{81}$$

We use this formula recursively to obtain the Taylor series of the above system. Let $\varphi(x(t), t) = x(t)$, then we have

$$\begin{aligned} x(t) &= x(0) + \int_0^t (L_\flat x)(x(s), s) ds \\ &= x(0) + \int_0^t a(x(s), s) ds. \end{aligned} \tag{82}$$

Let $\varphi(x(t), t) = a(x(t), t)$, then we have

$$a(x(t), t) = a(x(0), 0) + \int_0^t (L_\flat a)(x(s), s) ds. \tag{83}$$

Using the above two equations, we have

$$\begin{aligned} x(t) &= x(0) + \int_0^t \left(a(x(0), 0) + \int_0^{t_1} (L_\flat a)(x(t_2), t_2) dt_2 \right) dt_1 \\ &= x(0) + \int_0^t a(x(0), 0) dt_1 + \int_0^t \int_0^{t_1} (L_\flat a)(x(t_2), t_2) dt_2 dt_1. \\ &= x(0) + t a(x(0), 0) + \int_0^t \int_0^{t_1} (L_\flat a)(x(t_2), t_2) dt_2 dt_1. \end{aligned} \tag{84}$$

We can expand again the term inside of the integral by using the following relation,

$$(L_\flat a)(x(t), t) = (L_\flat a)(x(0), 0) + \int_0^t (L_\flat^2 a)(x(s), s) ds, \tag{85}$$

and obtain the expansion as follows,

$$x(t) = x(0) + t a(x(0), 0) + \frac{t^2}{2} (L_\flat a)(x(0), 0) + \int_0^t \int_0^{t_1} \int_0^{t_2} (L_\flat^2 a)(x(t_3), t_3) dt_3 dt_2 dt_1. \tag{86}$$

Applying this argument recursively, we can obtain the Taylor series of the deterministic system, if $a(\cdot, \cdot)$ is sufficiently smooth.

$$x(t) = x(0) + t a(x(0), 0) + \sum_{k=2}^{\infty} \frac{t^k}{k!} (L_\flat^{k-1} a)(x(0), 0). \tag{87}$$

Since the above discussion is valid when the integration interval is $(t, t+h)$ instead of $(0, t)$, it can be written as follows,

$$x(t+h) = x(t) + ha(x(t), t) + \sum_{k=2}^{\infty} \frac{h^k}{k!} (L_\flat^{k-1} a)(x(t), t). \tag{88}$$

Multi-dimensional case Let us consider the multi-dimensional ODE $\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, t)$. The total derivative of a smooth scalar function $\varphi(\mathbf{x}, t)$ is written as

$$\begin{aligned} d\varphi &= \frac{\partial \varphi}{\partial t} dt + \sum_i \frac{\partial \varphi}{\partial x_i} dx_i = \frac{\partial \varphi}{\partial t} dt + \sum_i \frac{\partial \varphi}{\partial x_i} \frac{dx_i}{dt} dt = \frac{\partial \varphi}{\partial t} dt + \left(\sum_i a_i(\mathbf{x}, t) \frac{\partial}{\partial x_i} \right) \varphi dt \\ &= \left(\frac{\partial \varphi}{\partial t} + (\mathbf{a}(\mathbf{x}, t) \cdot \nabla) \varphi \right) dt = \left(\frac{\partial}{\partial t} + \mathbf{a}(\mathbf{x}, t) \cdot \nabla \right) \varphi dt. \end{aligned} \tag{89}$$

Let $\boldsymbol{\varphi}(\mathbf{x}, t)$ be a vector-valued smooth function, then we immediately have

$$d\boldsymbol{\varphi} = (\partial_t + \mathbf{a}(\mathbf{x}, t) \cdot \nabla) \boldsymbol{\varphi} dt. \tag{90}$$

Using the scalar operator $L_\flat = (\partial_t + \mathbf{a}(\mathbf{x}, t) \cdot \nabla)$, we can obtain the following Taylor expansion similarly to the 1-dim case,

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(0) + t(L_\flat \mathbf{x})\Big|_{t=0} + \frac{t^2}{2}(L_\flat^2 \mathbf{x})\Big|_{t=0} + \cdots \\ &= \mathbf{x}(0) + t\left(\partial_t + \mathbf{a}(\mathbf{x}, t) \cdot \nabla\right) \mathbf{x}\Big|_{t=0} + \frac{t^2}{2}\left(\partial_t + \mathbf{a}(\mathbf{x}, t) \cdot \nabla\right)^2 \mathbf{x}\Big|_{t=0} + \cdots \\ &= \mathbf{x}(0) + t\mathbf{a}(\mathbf{x}(0), 0) + \frac{t^2}{2}\left(\partial_t + \mathbf{a}(\mathbf{x}, t) \cdot \nabla\right) \mathbf{a}(\mathbf{x}, t)\Big|_{t=0} + \cdots \end{aligned} \tag{91}$$

The second order term $(\mathbf{a} \cdot \nabla)\mathbf{a}$ is written as follows,

$$(\mathbf{a} \cdot \nabla)\, \mathbf{a} = \left(\sum_i a_i \partial_{x_i} \right) \begin{bmatrix} a_1(\mathbf{x}, t) \\ \vdots \\ a_d(\mathbf{x}, t) \end{bmatrix} = \begin{bmatrix} a_1 \partial_{x_1} a_1 & \cdots & a_d \partial_{x_d} a_1 \\ \vdots & \ddots & \vdots \\ a_1 \partial_{x_1} a_d & \cdots & a_d \partial_{x_d} a_d \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{92}$$

In a special case where each dimension is separable, i.e. $\partial_{x_i} a_j = 0 (i \neq j)$, the above $d \times d$ matrix is diagonal, and we have

$$(\mathbf{a} \cdot \nabla)\, \mathbf{a} = \begin{bmatrix} a_1 \partial_{x_1} a_1 \\ \vdots \\ a_d \partial_{x_1} a_d \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_d \end{bmatrix} \odot \begin{bmatrix} \partial_{x_1} \\ \vdots \\ \partial_{x_d} \end{bmatrix} \odot \begin{bmatrix} a_1 \\ \vdots \\ a_d \end{bmatrix} = \mathbf{a} \odot \nabla \odot \mathbf{a}. \tag{93}$$

where $\odot$ is the element-wise product or operation. In this case, it is sufficient to consider each dimension separately, and it is formally equivalent to the 1-dim case.

D.1.2 Itô-Taylor Expansion of Stochastic Systems

For the above reasons, it is sufficient to consider the 1-dim case even in the stochastic case. As is well known, Taylor expansion is not valid for stochastic systems $x_t = a(x_t, t)dt + b(x_t, t)dB_t$. This is because of the relation "$dB_t^2 \sim dt$". This effect is taken into account in the celebrated Itô's lemma, i.e., the stochastic version of eq. (80),

$$d\varphi = \underbrace{\left(\frac{\partial}{\partial t} + a(x, t) \frac{\partial}{\partial x} + \frac{b(x, t)^2}{2} \frac{\partial^2}{\partial x^2} \right)}_{=: L_\sharp} \varphi dt + \underbrace{\left(b(x, t) \frac{\partial}{\partial x} \right)}_{=: G_\sharp} \varphi dB_t. \tag{94}$$

By using Itô's formula recursively, we can obtain the following higher-order expansion of a stochastic system, which is called the Itô-Taylor expansion.

$$x_h = x_0 + \int_0^h a(x_t, t) dt + \int_0^h b(x_t, t) dB_t \tag{95}$$

$$
\begin{aligned}
&= x_0 + \int_0^h \left(a(x_0, 0) + \int_0^t (L_\sharp a)(x_s, s) ds + \int_0^t (G_\sharp a)(x_s, s) dB_s \right) dt \\
&\quad + \int_0^h \left(b(x_0, 0) + \int_0^t (L_\sharp b)(x_s, s) ds + \int_0^t (G_\sharp b)(x_s, s) dB_s \right) dB_t \\
&= x_0 + a(x_0, 0) \int_0^h dt + \int_0^h \int_0^t (L_\sharp a)(x_s, s) ds dt + \int_0^h \int_0^t (G_\sharp a)(x_s, s) dB_s dt \\
&\quad + b(x_0, 0) \int_0^h dB_t + \int_0^h \int_0^t (L_\sharp b)(x_s, s) ds dB_t + \int_0^h \int_0^t (G_\sharp b)(x_s, s) dB_s dB_t \\
&= x_0 + a(x_0, 0) \int_0^h dt + b(x_0, 0) \int_0^h dB_t \\
&\quad + \int_0^h \int_0^t \left((L_\sharp a)(x_0, 0) + \cdots \right) ds dt + \int_0^h \int_0^t \left((G_\sharp a)(x_0, 0) + \cdots \right) dB_s dt \\
&\quad + \int_0^h \int_0^t \left((L_\sharp b)(x_0, 0) + \cdots \right) ds dB_t + \int_0^h \int_0^t \left((G_\sharp b)(x_0, 0) + \cdots \right) dB_s dB_t \\
&= x_0 + a(x_0, 0) h + b(x_0, 0) B_h \\
&\quad + (L_\sharp a)(x_0, 0) \int_0^h \int_0^t ds dt + (G_\sharp a)(x_0, 0) \int_0^h \int_0^t dB_s dt \\
&\quad + (L_\sharp b)(x_0, 0) \int_0^h \int_0^t ds dB_t + (G_\sharp b)(x_0, 0) \int_0^h \int_0^t dB_s dB_t \\
&\quad + \textit{Remainder}, \qquad (96)
\end{aligned}
$$

where the remainder consists of triple integrals as follows,

$$
\textit{Remainder} = \int_0^h \int_0^t \int_0^s (L_\sharp^2 a)(x_u, u) du ds dt + \cdots . \qquad (97)
$$

We ignore these terms now. If we also ignore the double integrals, we obtain the Euler-Maruyama scheme.

Evaluation of Each Integral in eq. (96) Let us evaluate the double integrals.

$$
\begin{cases}
\int_0^h \int_0^t ds dt & \cdots \text{(deterministic)} \\
\int_0^h \int_0^t dB_s dt & \cdots \text{(stochastic 1)} \\
\int_0^h \int_0^t ds dB_t & \cdots \text{(stochastic 2)} \\
\int_0^h \int_0^t dB_s dB_t & \cdots \text{(stochastic 3)}
\end{cases}
$$

■ *Deterministic*: The deterministic one, $\int_0^h \int_0^t ds dt$, is easy to evaluate.

$$
\int_0^h \int_0^t ds dt = \int_0^h t dt = \frac{1}{2} h^2. \qquad (98)
$$

■ *Stochastic 1*: Other integrals contain stochastic integrations. Let us denote the first one by $\tilde{z}$.

$$
\tilde{z} \coloneqq \int_0^h \int_0^t dB_s dt = \int_0^h B_t dt. \qquad (99)
$$

As $\tilde{z}$ is the limit of a sum of Gaussian variables, i.e.,

$$\begin{aligned}\tilde{z} &= \lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{h}{n}B_{hi/n} = \lim_{n\to\infty}\frac{h}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{i}(B_{hj/n}-B_{h(j-1)/n})\\ &= \lim_{n\to\infty}\frac{h}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{i}W_j, \quad W_j\sim\mathcal{N}(0,\frac{h}{n}),\end{aligned} \tag{100}$$

so $\tilde{z}$ is also a Gaussian, whose mean is 0. The variance, however, requires some discussions, which shall be seen later. In addition, we shall also see that $\tilde{z}$ is correlated with B_h,

$$\mathbb{E}[\tilde{z}\cdot B_h]\neq 0. \tag{101}$$

■ *Stochastic 2*: The second one has the correlation with the first one as follows. Here, we use the integral-by-parts formula. See e.g. (Øksendal, 2013, Theorem 4.1.5).

$$\int_0^h\int_0^t dsdB_t = \int_0^h tdB_t = tB_t\Big|_0^h - \int_0^h B_t dt = hB_h - \tilde{z}. \tag{102}$$

■ *Stochastic 3*: The third one is computed as follows, using a famous formula of Itô integral.

$$\int_0^h\int_0^t dB_s dB_t = \int_0^h B_t dB_t = \frac{1}{2}\left(B_h^2 - h\right). \tag{103}$$

This is derived by substituting $\varphi(x,t)=x^2$ and $x_t = B_t$ (i.e., $dx_t = 0\cdot dt + 1\cdot dB_t$) into Itô's formula eq. (94).

■ *Substituting the Integrals to the Itô-Taylor Expansion*: Let us denote $\tilde{w} := B_h \sim \mathcal{N}(0,h)$, and let us shift the integral interval from $(0,h)$ to $(t,t+h)$. Then, we may rewrite the above second order expansion as follows, which we have already seen in the main text,

$$\begin{aligned}x_{t+h} = x_t &+ ha(x_t,t) + \tilde{w}b(x_t,t)\\ &+ (L_\sharp a)(x_t,t)\cdot h^2/2 + (G_\sharp a)(x_t,t)\cdot\tilde{z}\\ &+ (L_\sharp b)(x_t,t)\cdot(\tilde{w}h-\tilde{z}) + (G_\sharp b)(x_t,t)\cdot(\tilde{w}^2-h)/2.\end{aligned} \tag{104}$$

Covariance of the Random Variables $\tilde{w},\tilde{z}$ Next, let us evaluate the the variance of $\tilde{z}$, and the correlation between the Gaussian variables $\tilde{w}$ and $\tilde{z}$. Let us first calculate the variance of $(\tilde{w}h-\tilde{z})$. By Itô's isometry, we have

$$\mathbb{E}[(\tilde{w}h-\tilde{z})^2] = \mathbb{E}\left[\left(\int_0^h sdB_s\right)^2\right] = \mathbb{E}\left[\int_0^h s^2 ds\right] = \frac{1}{3}h^3. \tag{105}$$

The correlation between $\tilde{w}$ and $\tilde{z}$ is similarly evaluated by Itô's isometry (see (Øksendal, 2013, Proof of Lemma 3.1.5)) as follows,

$$\mathbb{E}[\tilde{w}\tilde{z}] = \mathbb{E}\left[B_h\int_0^h sdB_s\right] = \mathbb{E}\left[\left(\int_0^h dB_s\right)\left(\int_0^h sdB_s\right)\right] = \mathbb{E}\left[\int_0^h sds\right] = \frac{1}{2}h^2. \tag{106}$$

Using the above variance and covariance, we can calculate the variance of $\tilde{z}$ as follows,

$$\mathbb{E}[\tilde{z}^2] = \mathbb{E}[(\tilde{w}h-\tilde{z})^2 - h^2\tilde{w}^2 + 2h\tilde{w}\tilde{z}] = \frac{1}{3}h^3 - h^2\cdot h + 2h\cdot\frac{h^2}{2} = \frac{1}{3}h^3. \tag{107}$$

We need to find random variables $\tilde{w},\tilde{z}$ that satisfy the requirements for (co)variances that $\mathbb{E}[\tilde{w}^2]=h$, $\mathbb{E}[\tilde{w}\tilde{z}]=h^2/2$ and $\mathbb{E}[\tilde{z}^2]=h^3/3$, and we can easily verify that the following ones do,

$$\begin{bmatrix}\tilde{w}\\ \tilde{z}\end{bmatrix} = \begin{bmatrix}\sqrt{h} & 0\\ h\sqrt{h}/2 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{108}$$

where $u_1, u_2 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. Let us compute the covariance matrix just to be sure.

$$\begin{aligned}
\mathbb{E}\left[\begin{bmatrix}\tilde{w}\\ \tilde{z}\end{bmatrix}\begin{bmatrix}\tilde{w} & \tilde{z}\end{bmatrix}\right] &= \mathbb{E}\left[\begin{bmatrix}\sqrt{h} & 0\\ h\sqrt{h}/2 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}\begin{bmatrix}u_1 & u_2\end{bmatrix}\begin{bmatrix}\sqrt{h} & h\sqrt{h}/2\\ 0 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\right]\\
&= \begin{bmatrix}\sqrt{h} & 0\\ h\sqrt{h}/2 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\mathbb{E}\left[\begin{bmatrix}u_1^2 & u_1u_2\\ u_1u_2 & u_2^2\end{bmatrix}\right]\begin{bmatrix}\sqrt{h} & h\sqrt{h}/2\\ 0 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\\
&= \begin{bmatrix}\sqrt{h} & 0\\ h\sqrt{h}/2 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\begin{bmatrix}\sqrt{h} & h\sqrt{h}/2\\ 0 & h\sqrt{h}/2\sqrt{3}\end{bmatrix}\\
&= \begin{bmatrix}h & h^2/2\\ h^2/2 & h^3/3\end{bmatrix}. \qquad (109)
\end{aligned}$$

D.2 DIFFUSION PROCESS AND FOKKER-PLANCK EQUATION

Derivation of Fokker-Planck Equation In this section, let us derive the Fokker-Planck equation

$$\partial_t p(\mathbf{x}_t, t) = -\nabla_{\mathbf{x}_t} \left(f(\mathbf{x}_t, t) p(\mathbf{x}_t, t) \right) + \frac{1}{2} g(t)^2 \nabla^2_{\mathbf{x}_t} p(\mathbf{x}_t, t). \tag{110}$$

from the Itô SDE,

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt + g(t)d\mathbf{B}_t. \tag{111}$$

For simplicity, we consider a 1-dim case here. The following approach using the Fourier methods (characteristic function) will be easy and intuitive. See also e.g. (Cox & Miller, 1965, § 5),(Karlin & Taylor, 1981, § 15),(Shreve, 2004, § 6) and (Särkkä & Solin, 2019, § 5).

Let us consider an infinitesimal time step h. Then x_{t+h} is written as follows,

$$x_{t+h} = x_t + w_t, \quad w_t \sim \mathcal{N}(f(x_t, t)h, g(t)^2 h). \tag{112}$$

Let us consider the characteristic function $\phi_x(\omega) \coloneqq \mathbb{E}_{p(x)}[\exp i\omega x]$ (where $i = \sqrt{-1}$), which is the Fourier transform of the density function $p(x)$. Because of the convolution theorem $\phi_{x+y}(\omega) = \phi_x(\omega) \cdot \phi_y(\omega)$, the characteristic functions of $p(x_{t+h}, t+h)$, $p(x_t, t)$, and $p(w_t)$ have the following relation,

$$\phi_{x_{t+h}}(\omega) = \phi_{w_t}(\omega)\phi_{x_t}(\omega). \tag{113}$$

It is easily shown that the characteristic function of the above Gaussian is given by

$$\phi_{w_t}(\omega) = \exp\left(i\omega f(x_t, t)h - \frac{1}{2} g(t)^2 h\omega^2 \right). \tag{114}$$

Expanding the r.h.s. up to the first order terms of h, we have

$$\phi_{w_t}(\omega) = 1 + i\omega f(x_t, t)h - \frac{1}{2} g(t)^2 h\omega^2 + O(h^2). \tag{115}$$

Thus we obtain

$$\frac{\phi_{x_{t+h}}(\omega) - \phi_{x_t}(\omega)}{h} = \left(-(-i\omega) f(x_t, t) + (-i\omega)^2 \frac{g(t)^2}{2} \right) \phi_t(\omega) + O(h). \tag{116}$$

When $h \to 0$,

$$\partial_t \phi_{x_t}(\omega) = -(-i\omega) f(x_t, t)\phi_t(\omega) + (-i\omega)^2 \frac{g(t)^2}{2} \phi_t(\omega). \tag{117}$$

Since $(-i\omega)$ in the Fourier domain corresponds to the spatial derivative ∂_{x_t} in the real domain [6], it translates as follows,

$$\partial_t p(x_t, t) = -\partial_{x_t}(f(x_t, t)p(x_t, t)) + \partial^2_{x_t} \frac{g(t)^2}{2} p(x_t, t), \tag{118}$$

and thus we have obtained the Fokker-Planck equation. In particular, if $f = 0$, this equation is called the heat equation, which was also first developed by Fourier.

Example: Overdamped Langevin Dynamics When the drift term is the gradient of a potential function $U(\cdot)$, the SDE is often called the overdampled Langevin equation[7],

$$d\mathbf{x}_t = -\nabla_{\mathbf{x}_t} U(\mathbf{x}_t)dt + \sqrt{2D}d\mathbf{B}_t, \tag{119}$$

[6] Integral by parts and $p(-\infty) = p(\infty) = 0$. Formally writing,

$$\mathbb{E}_{\frac{d}{dx}p(x)}[e^{i\omega x}] = \int_{-\infty}^{\infty} e^{i\omega x} \frac{d}{dx} p(x)dx = -\int_{-\infty}^{\infty} i\omega e^{i\omega x} p(x)dx = (-i\omega)\mathbb{E}_{p(x)}[e^{i\omega x}].$$

[7] The Langevin equation should actually be the following equation system, which is called the underdamped Langevin equation,

$$\dot{\mathbf{x}}_t = \mathbf{v}_t, \quad M\dot{\mathbf{v}}_t = -\gamma \mathbf{v}_t - \nabla_{\mathbf{x}_t} U(\mathbf{x}_t) + \sqrt{2D}\frac{d\mathbf{B}_t}{dt},$$

where $\mathbf{v}_t$ is the velocity (momentum) variable, M is the mass of particle, and γ is a constant called *friction* or *viscosity* coefficient. In this case, the energy function should also be modified as $E = \frac{1}{2}M\|\mathbf{v}\|^2 + U(\mathbf{x})$. Assuming that the mass is very small compared to the friction, the derivative of momentum $M\dot{\mathbf{v}}_t$ can be ignored (i.e., if the force F is constant, the ODE $M\dot{v} = -\gamma v + F$ has the solution $v = Ce^{-t\gamma/M} + F/\gamma$, and the velocity immediately converges to F/γ if $M \ll \gamma$) and the overdamped equation is obtained.

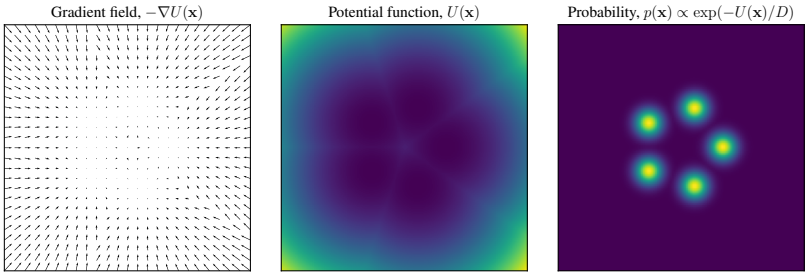


Figure 11: Force field (potential gradient, score function) $-\nabla U(\mathbf{x})$, potential function $U(\mathbf{x})$, and the Boltzmann distribution $p(\mathbf{x}) = \frac{1}{Z} e^{-U(\mathbf{x})/D}$. The scalar potential $U(\mathbf{x})$ is given by eq. (122)

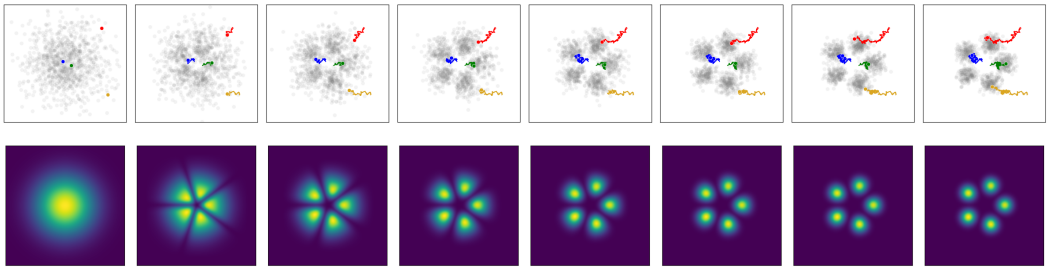


Figure 12: Numerical simulation of Langevin and Fokker-Planck equations. (Top: microscopic particle-wise dynamics) Time evolution of 1000 random samples $\{\mathbf{x}_t^{(i)}\}_{i=1}^{1000}$ obeying the Langevin equation. Paths of four particular samples are shown in color. The time evolves from left to right. (Bottom: macroscopic population dynamics) Time evolution of the density function $p(\mathbf{x}, t)$ obeying the Fokker-Planck equation.

where D is a scalar constant. Its associated Fokker-Planck equation is

$$\begin{aligned} \partial_t p(\mathbf{x}_t, t) &= \nabla_{\mathbf{x}_t} \cdot \left(\nabla_{\mathbf{x}_t} U(\mathbf{x}_t) \cdot p(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} D p(\mathbf{x}_t, t) \right) \\ &= \nabla_{\mathbf{x}_t} \cdot \left[\nabla_{\mathbf{x}_t} \left(U(\mathbf{x}_t) + D \log p(\mathbf{x}_t, t) \right) \cdot p(\mathbf{x}_t, t) \right]. \end{aligned} \tag{120}$$

If $U(\mathbf{x}_t) + D \log p(\mathbf{x}_t, t)$ is constant, the r.h.s. will be zero, and therefore, $\partial_t p(\mathbf{x}_t, t) = 0$. That is,

$$p(\mathbf{x}) = \frac{1}{Z} e^{-U(\mathbf{x})/D}, \quad \text{where } Z = \int_{\mathbb{R}^d} e^{-U(\mathbf{x})/D} d\mathbf{x} \tag{121}$$

is the stationary solution of the FPE, and it does no longer evolve over time. Therefore, it is expected that the particles obeying the Langevin equation will eventually follow this Boltzmann distribution.

Let us compare the microscopic dynamics of each particle obeying the Langevin equation with the macroscopic dynamics of the population obeying the FPE using a 2-dim toy model. The energy function we use here is the following Gaussian mixture model,

$$U(x, y) = -\log \sum_{k=1}^{5} \exp \left[-\frac{(x - \cos \frac{2k\pi}{5})^2 + (y - \sin \frac{2k\pi}{5})^2}{2\sigma^2} \right], \tag{122}$$

where $\sigma = 0.1$. The diffusion parameter was $D = 5$. Figure 11 shows the force field $-\nabla U(\mathbf{x})$ where $\mathbf{x} = (x, y)$, potential function $U(\mathbf{x})$, and the Boltzmann distribution $p(\mathbf{x}) = \frac{1}{Z} e^{-U(\mathbf{x})/D}$.

Figure 12 shows the time evolution of Langevin and Fokker-Planck equations, where the initial density was $p(\mathbf{x}_0, 0) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The time step size was $h = 5 \times 10^{-5}$, and the figures are plotted every 50 steps. These figures will be helpful to intuitively understand that the SDE (Langevin equation) and the FPE are describing the same phenomenon from different perspectives: individual description or population description.

D.3 Derivation of the Probability Flow ODE

For convenience, the derivation of the PF-ODE is briefly described here. For more general case details, please refer to the original paper (Song et al., 2020b).

Firstly, let us consider an SDE,

$$dx_t = f(x_t, t)dt + g(t)dB_t. \tag{123}$$

The associated FPE is,

$$\partial_t p(x_t, t) = \partial_{x_t}\left(f(x_t, t)p(x_t, t)\right) - \frac{1}{2}g(t)^2\partial_{x_t}^2 p(x_t, t). \tag{124}$$

Here, let us forcibly incorporate the diffusion term (i.e., the second-order derivative) into the drift term (first-order derivative).

$$\begin{aligned}
\text{r.h.s.} &= \partial_{x_t}\Big[f(x_t, t)p(x_t, t) - \frac{1}{2}g(t)^2\partial_{x_t}p(x_t, t)\Big] \\
&= \partial_{x_t}\Big[\Big(f(x_t, t) - \frac{1}{2}g(t)^2\frac{1}{p(x_t, t)}\partial_{x_t}p(x_t, t)\Big)p(x_t, t)\Big] \\
&= \partial_{x_t}\Big[\Big(\underbrace{f(x_t, t) - \frac{1}{2}g(t)^2\partial_{x_t}\log p(x_t, t)}_{=:\bar{\mathbf{f}}_\flat(x_t, t)}\Big)p(x_t, t)\Big].
\end{aligned} \tag{125}$$

Now we have obtained a 'FPE' of the form,

$$\partial_t p(x_t, t) = \partial_{x_t}\left(\bar{\mathbf{f}}_\flat(x_t, t)p(x_t, t)\right) - \cancel{\partial_{x_t}^2(0 \cdot p(x_t, t))}, \tag{126}$$

and its associated 'SDE',

$$dx_t = \bar{\mathbf{f}}_\flat(x_t, t)dt + \cancel{0 \cdot dB_t}. \tag{127}$$

Although this 'SDE' does not give the same particle-wise dynamics as the original SDE, its macroscopic dynamics of population is exactly the same as the original one, since the associated FPE has not changed from the original one at all. That is, the density evolves from $p(x_0, 0)$ to $p(x_T, T)$ exactly the same way.

Since the obtained 'SDE' is actually a deterministic ODE, its time-reversal is simply obtained by flipping the sign, i.e.,

$$dx_t = (-\bar{\mathbf{f}}_\flat(x_t, t))(-dt). \tag{128}$$

Thus the Probability Flow ODE is obtained. If the terminal random variables are drawn from $p(x_T, T)$, then the particles obeying this ODE will reconstruct the initial density $p(x_0, 0)$ as a population.

Note that the vector field of the form $\mathbf{J} \coloneqq \rho\mathbf{v}$ is often referred to as the *flux*, particularly in physical contexts e.g. fluid dynamics, where ρ is density and $\mathbf{v}$ is velocity, and the PDE of the form

$$\partial_t \rho + \nabla \cdot \mathbf{J} = 0 \tag{129}$$

is often called the *continuity equation*, which is closely related to the conservation laws. In the present case, $\mathbf{J} \coloneqq p\bar{\mathbf{f}}_\flat$ is understood as the flux, and therefore, $\bar{\mathbf{f}}_\flat$ could be understood as a sort of velocity field.

D.4 ON THE CONVERGENCE OF NUMERICAL SDE SCHEMES.

Let us introduce two convergence concepts which are commonly used in numerical SDE studies. See also (Kloeden et al., 1994, § 3.3, § 3.4)

Definition 1 (Strong Convergence)**.** *Let $\tilde{\mathbf{x}}_t$ be the path at the continuous limit $h \to 0$, and $\mathbf{x}_t$ be the discretized numerical path, computed by a numerical scheme with the step size $h > 0$. Then, it is said the numerical scheme has the strong order of convergence γ if the following inequality holds for a certain constant $K_s > 0$,*

$$\mathbb{E}[|\mathbf{x}_t - \tilde{\mathbf{x}}_t|] \leq K_s h^{\gamma}. \tag{130}$$

Definition 2 (Weak Convergence)**.** *Similarly, it is said that the scheme has the weak order of convergence β, if the following inequality holds for any test functions $\phi(\cdot)$ in a certain class of functions, and a certain constant $K_w > 0$,*

$$|\mathbb{E}[\phi(\mathbf{x}_t)] - \mathbb{E}[\phi(\tilde{\mathbf{x}}_t)]| \leq K_w h^{\beta}. \tag{131}$$

It is known that the Euler-Maruyama scheme has the strong convergence of order $\gamma = 0.5$, and weak order of $\beta = 1$, in general. However, for more specific cases including the diffusion generative models that the diffusion coefficient $g(t)$ is not dependent on $\mathbf{x}_t$, the Euler-Maruyama scheme has a little better strong convergence of order $\gamma = 1$.

The strong convergence is concerned with the precision of the path, while the weak convergence is with the precision of the moments. In our case, we are not much interested in whether a data $\hat{\mathbf{x}}_0$ generated using a finite $h > 0$ approximates the continuous limit $\tilde{\mathbf{x}}_0, (h \to 0)$ driven by the same Brownian motion. Instead, we are more interested in whether the density $p(\hat{\mathbf{x}}_0)$ of the samples generated with a finite step size $h > 0$ approximates the ideal density $p(\tilde{\mathbf{x}}_0), (h \to 0)$ which is supposed to approximate the true density $p(\mathbf{x}_0)$. In this sense, the concept of strong convergence is not much important for us, but the weak convergence would be sufficient.

D.5 RUNGE-KUTTA METHODS

Let us briefly introduce the derivation of Runge-Kutta methods for a 1-dim ODE $\dot{x} = f(x,t)$. We are interested in deriving the following derivative-free formula

$$x_{t+h} = x_t + \sum_{i=1}^{n} hb_i k_i + o(h^p), \quad \text{where } k_i = f(x_t + \sum_{j<i} ha_{ij}k_j, t + hc_i), \quad n \geq p. \tag{132}$$

The array of coefficients are often shown in the following form, which is called the Butcher tableau.

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_n & a_{n1} & a_{n2} & \cdots & a_{n,n-1} & \\ \hline & b_1 & b_2 & \cdots & b_{n-1} & b_n \end{array} \tag{133}$$

Now the problem is how to design each value in this tableau. First, for simplicity, let us consider the following 2nd order case,

$$\begin{aligned} x_{t+h} &= x_t + hb_1k_1 + hb_2k_2 + o(h^2) \\ k_1 &= f(x_t, t) \\ k_2 &= f(x_t + ha_{21}k_1, t + hc_2), \end{aligned}$$

or,

$$x_{t+h} = x_t + hb_1 f(x_t,t) + hb_2 f(x_t + ha_{21}f(x_t,t), t + hc_2) + o(h^2). \tag{134}$$

Noting that the Taylor expansions of the l.h.s. and the third term in r.h.s. are written as follows,

$$\text{(l.h.s.)} = x_t + hf(x_t,t) + \frac{h^2}{2}(\dot{f} + ff')(x_t,t) + o(h^2), \tag{135}$$

$$\text{(third term of r.h.s.)} = hb_2 \left(f(x_t,t) + ha_{21}f(x_t,t)f'(x_t,t) + hc_2\dot{f}(x_t,t) + o(h) \right), \tag{136}$$

we obtain the following relation.

$$\begin{aligned} &x_t + hf(x_t,t) + \frac{h^2}{2}(\dot{f} + ff')(x_t,t) = \\ &\qquad x_t + hb_1 f(x_t,t) + hb_2 \left(f(x_t,t) + ha_{21}f(x_t,t)f'(x_t,t) + hc_2\dot{f}(x_t,t) \right) + o(h^2). \end{aligned} \tag{137}$$

Comparing both sides, we can find that the derivatives in both sides can be eliminated if the following equation are satisfied,

$$hf(x_t,t) = hb_1 f(x_t,t) + hb_2 f(x_t,t), \tag{138}$$

$$\frac{h^2}{2}(\dot{f} + ff')(x_t,t) = hb_2 \left(ha_{21}f(x_t,t)f'(x_t,t) + hc_2\dot{f}(x_t,t) \right). \tag{139}$$

By simplifying the above equations, we obtain the following relations of coefficients.

$$b_1 + b_2 = 1, \quad b_2 a_{21} = \frac{1}{2}, \quad b_2 c_2 = \frac{1}{2}, \tag{140}$$

and thus we obtain the following Butcher tableau.

$$\begin{array}{c|cc} 0 & & \\ c_2 & a_{21} & \\ \hline & b_1 & b_2 \end{array} = \begin{array}{c|cc} 0 & & \\ c_2 & c_2 & \\ \hline & 1 - 1/(2c_2) & 1/(2c_2) \end{array} \tag{141}$$

(When $c_2 = 1$, the method is particularly called Heun's method.) Thus we have confirmed that the second order Taylor expansion of x_{t+h} is expressed as eq. (134), which is independent of any derivatives of f.

We can similarly consider higher-order methods. In the 3rd-order case, following relations are automatically obtained after a little effort of symbolic math programming using e.g. SymPy,

$$b_1 + b_2 + b_3 = 1, \quad b_2 c_2 + b_3 c_3 = \frac{1}{2}, \quad b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3}, \quad a_{32} a_{21} b_3 = \frac{1}{6}, \quad a_{32} b_3 c_2 = \frac{1}{6},$$
$$a_{21} b_2 + (a_{31} + a_{32}) b_3 = \frac{1}{2}, \quad a_{21}^2 b_2 + (a_{32} + a_{31})^2 b_3 = \frac{1}{3}, \quad a_{21} b_2 c_2 + (a_{31} + a_{32}) b_3 c_3 = \frac{1}{3}.$$

By solving this equation system using SymPy, the following Butcher tableau is obtained.

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & a_{21} & & \\ c_3 & a_{31} & a_{32} & \\ \hline & b_1 & b_2 & b_3 \end{array} = \begin{array}{c|ccc} 0 & & & \\ c_2 (\neq 2/3) & c_2 & & \\ c_3 (\neq c_2) & \frac{c_3(3c_2^2 - 3c_2 + c_3)}{c_2(3c_2 - 2)} & \frac{c_3(c_2 - c_3)}{c_2(3c_2 - 2)} & \\ \hline & 1 - \frac{1}{2c_2} - \frac{1}{2c_3} + \frac{1}{3c_2 c_3} & \frac{2 - 3c_3}{6c_2(c_2 - c_3)} & \frac{3c_2 - 2}{6c_3(c_2 - c_3)} \end{array} \tag{142}$$

It will naturally become much more complicated when we consider higher-order methods, though it can be simpler if substituting specific values. In the 4th-order case, when the Butcher tableau is written as follows, the method is particularly called the Classical 4th-order Runge-Kutta (RK4) method.

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ c_4 & a_{41} & a_{42} & a_{43} & \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array} = \begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{143}$$

Numerical Example Let us consider a toy example $\dot{x} = x \sin t, x_0 = 1$. The exact solution is $x_t = e^{1 - \cos t}$. Figure 13 compares the numerical solutions of the Euler, Heun, Classical RK4 and Taylor 2nd methods, where the step size is $h = 0.5$. It is clearly observed that the Euler method immediately deviates significantly from the exact solution but higher-order methods follow it for longer periods; while 2nd-order methods (Heun and Taylor 2nd) gradually deviate from the exact solution, the Classical 4th-order Runge-Kutta method more finely approximates the exact solution.

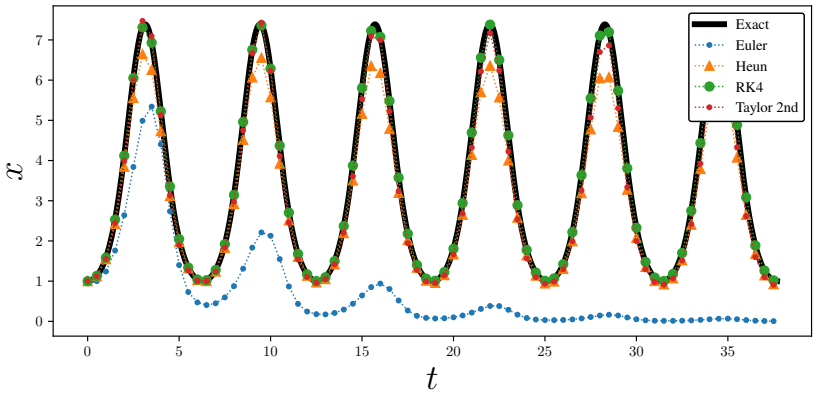


Figure 13: Comparison of numerical solutions of the ODE $\dot{x} = x \sin t, x_0 = 1$.

Note that the Taylor 2nd method is given as follows,

$$x_{t+h} = x_t + h(x_t \sin t) + \frac{h^2}{2} \underbrace{(x_t \cos t + x_t \sin^2 t)}_{\text{derivative of } x \sin t}. \tag{144}$$

In this case the computation of the derivative $(\partial_t + x \sin t \partial_x)(x \sin t)$ is tractable. However, it will be infeasible if this term is a little more complicated. The Runge-Kutta method is advantageous if the derivatives are not easy to compute.

E PSEUDOCODE

E.1 QUASI-TAYLOR SAMPLER

The following pseudocode shows the proposed Quasi-Taylor sampler.

Algorithm 1 Quasi-Taylor Sampling Scheme with Ideal Derivatives

Require:
Trained neural network model $\mathbf{S}_\theta(\mathbf{x}_t, t, \mathbf{c})$ ♯ The conditioning information $\mathbf{c}$ is optional.
Data size $d > 0$ $\cdots$ *Int*
Total time $T > 0$ $\cdots$ *Float*
Number of steps $N > 0$ $\cdots$ *Int*
Initial and terminal noise levels $\nu_0, \nu_T \in (0, 1)$ $\cdots$ *Float*
(Optional) Conditioning information $\mathbf{c}$
Step size schedule $h_1, h_2, \cdots, h_N$ $\cdots$ *List of Float*

Constants:
$A \leftarrow \dfrac{2\sqrt{\nu_0}}{1-\sqrt{\nu_0}}, \quad k \leftarrow \dfrac{1}{T}\left(\log \dfrac{2\sqrt{\nu_T}}{1-\sqrt{\nu_T}} - \log A\right)$ ♯ eq. (29), § C.2

Begin
$\mathbf{x} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ ♯ Draw a d-dimensional Gaussian noise with unit variance
for $n = 1$ **to** N **do**

[Step size and time parameter]
$h \leftarrow h_n$
$t \leftarrow T - \sum_{i=1}^{n-1} h_i$
[Compute the Noise Level] ♯ eq. (28), § C
$\lambda \leftarrow \log(1 + Ae^{kt}), \quad \dot{\lambda} \leftarrow \dfrac{Ake^{kt}}{Ae^{kt}+1}, \quad \ddot{\lambda} \leftarrow \dfrac{Ak^2e^{kt}}{(Ae^{kt}+1)^2}, \quad \dddot{\lambda} \leftarrow \dfrac{-Ak^3e^{kt}(Ae^{kt}-1)}{(Ae^{kt}+1)^3}.$
$\nu \leftarrow \tanh^2 \dfrac{\lambda}{2}.$
$\beta \leftarrow \dot{\lambda} \tanh \dfrac{\lambda}{2}, \quad \dot{\beta} \leftarrow \ddot{\lambda} \tanh \dfrac{\lambda}{2} + \dfrac{\dot{\lambda}^2}{2\cosh^2 \frac{\lambda}{2}}, \quad \ddot{\beta} \leftarrow -\dfrac{\dot{\lambda}^3 \sinh \frac{\lambda}{2}}{2\cosh^3 \frac{\lambda}{2}} + \dddot{\lambda} \tanh \dfrac{\lambda}{2} + \dfrac{3\dot{\lambda}\ddot{\lambda}}{\cosh^2 \frac{\lambda}{2}}$
[Compute Coefficients]
Compute $\rho^\flat$ using eq. (23)
Compute $\mu^\flat$ using eq. (24)
[Update Data]
$\mathbf{x} \leftarrow \rho^\flat \mathbf{x} + \mu^\flat \mathbf{S}_\theta(\mathbf{x}, t, \mathbf{c})/\sqrt{\nu}$ ♯ eq. (22)
(Optional) Clip outliers of $\mathbf{x}$ so that e.g. $-1 \leq \mathbf{x} \leq 1$.
end for
End
Output: $\mathbf{x}$

Note that, $\rho_\flat$ and $\mu_\flat$, as well as ν, λ, β and their derivatives, are only dependent on $T, \{h_i\}, \nu_0$ and ν_T. Therefore, they can be pre-computed before the actual synthesis.

E.2 QUASI-ITÔ-TAYLOR SAMPLER

The following pseudocode shows the proposed Quasi-Itô-Taylor sampler.

Algorithm 2 Quasi-Itô-Taylor Sampling Scheme with Ideal Derivatives

Require:
Trained neural network model $\mathbf{S}_\theta(\mathbf{x}_t, t, \mathbf{c})$ ♯ The conditioning information $\mathbf{c}$ is optional.
Data size $d > 0$ $\cdots$ *Int*
Total time $T > 0$ $\cdots$ *Float*
Number of steps $N > 0$ $\cdots$ *Int*
Initial and terminal noise levels $\nu_0, \nu_T \in (0, 1)$ $\cdots$ *Float*
(Optional) Conditioning information $\mathbf{c}$
Step size schedule $h_1, h_2, \cdots, h_N$ $\cdots$ *List of Float*

Constants:
$A \leftarrow \dfrac{2\sqrt{\nu_0}}{1-\sqrt{\nu_0}}, \quad k \leftarrow \dfrac{1}{T}\left(\log \dfrac{2\sqrt{\nu_T}}{1-\sqrt{\nu_T}} - \log A\right)$ ♯ eq. (29), § C.2

Begin

$\mathbf{x} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ ♯ Draw a d-dimensional Gaussian noise with unit variance

for $n = 1$ **to** N **do**

[Step size and time parameter]

$h \leftarrow h_n$

$t \leftarrow T - \sum_{i=1}^{n-1} h_i$

[Compute the Noise Level] ♯ eq. (28), § C

$\lambda \leftarrow \log(1 + Ae^{kt}), \quad \dot{\lambda} \leftarrow \dfrac{Ake^{kt}}{Ae^{kt}+1}, \quad \ddot{\lambda} \leftarrow \dfrac{Ak^2e^{kt}}{(Ae^{kt}+1)^2}.$

$\nu \leftarrow \tanh^2 \dfrac{\lambda}{2}.$

$\beta \leftarrow \dot{\lambda} \tanh \dfrac{\lambda}{2}, \quad \dot{\beta} \leftarrow \ddot{\lambda} \tanh \dfrac{\lambda}{2} + \dfrac{\dot{\lambda}^2}{2\cosh^2 \frac{\lambda}{2}}.$

[Compute Coefficients]

Compute $\rho^\sharp$ using eq. (26)

Compute $\mu^\sharp$ using eq. (26)

[Draw a Correlated Driving Noise]
if $n = N$ **then**
$\mathbf{n}^\sharp = \mathbf{0}_d$ ♯ No noise is injected at the final step
else
$\mathbf{w} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d), \quad \mathbf{u} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d), \quad \mathbf{z} = \dfrac{1}{2}\mathbf{w} + \dfrac{1}{2\sqrt{3}}\mathbf{u}$ ♯ See eq. (108) and Theorem 1

Compute $\mathbf{n}^\sharp$ using eq. (27) and $\mathbf{w}, \mathbf{z}$ above.
end if

[Update Data]

$\mathbf{x} \leftarrow \rho^\sharp \mathbf{x} + \mu^\sharp \mathbf{S}_\theta(\mathbf{x}, t, \mathbf{c})/\sqrt{\nu} + \mathbf{n}^\sharp$ ♯ eq. (25)

(Optional) Clip outliers of $\mathbf{x}$ so that e.g. $-1 \leq \mathbf{x} \leq 1$.
end for
End
Output: $\mathbf{x}$

E.3 TRAINING

The following pseudocode shows an example for the training of diffusion-based generative models.

Algorithm 3 Training of Diffusion models

Require:
Data size $d > 0$ $\cdots$ *Int*
Training Data $\mathcal{D} = \{\mathbf{x}_0^{(i)}, \mathbf{c}^{(i)}\}$, where $\mathbf{x}_0^{(i)} \in \mathbb{R}^d$ ♯ The conditioning information $\mathbf{c}$ is optional.
Neural network model $\mathbf{S}_\theta(\mathbf{x}_t, t, \mathbf{c})$, parameterized by θ.
Neural network optimizer (*Adam*) and its parameters (e.g. learning rate)
Total time $T > 0$ $\cdots$ *Float* ♯ Not necessarily the same as the one for the synthesis.
Initial and terminal noise levels $\nu_0, \nu_T \in (0, 1)$ ♯ Not necessarily the same as the one for the synthesis.
Batch size $b > 0$ $\cdots$ *Int*
Constants:
$A \leftarrow \dfrac{2\sqrt{\nu_0}}{1-\sqrt{\nu_0}}, \quad k \leftarrow \dfrac{1}{T}\left(\log \dfrac{2\sqrt{\nu_T}}{1-\sqrt{\nu_T}} - \log A\right)$ ♯ eq. (29), § C.2
Begin
for sufficiently many times until convergence **do**
[Draw a Batch]
batch $\leftarrow$ [ ] ♯ Empty list
for b times (batch size) **do**
$(\mathbf{x}_0, \mathbf{c}) \sim \mathcal{D}$ ♯ Draw a data from the set of training data; the conditioning $\mathbf{c}$ is optional.

$t \sim \mathrm{Uniform}(0, T)$ ♯ Draw a time parameter t from the uniform distribution.

$\lambda \leftarrow \log(1 + Ae^{kt})$ ♯ § C

$\beta \leftarrow \dot{\lambda} \tanh \dfrac{\lambda}{2}, \nu \leftarrow \tanh^2 \dfrac{\lambda}{2}$ ♯ eq. (28), § C

$\mathbf{w} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ ♯ Draw a d-dimensional Gaussian noise with unit variance.

$SNR' \leftarrow \dfrac{(1-\nu)\beta}{\nu^2}$ ♯ Compute the SNR-based weight. See eq. (145) and (Kingma et al., 2021)

Append the tuple $(\mathbf{x}_0, \mathbf{c}, t, \nu, \mathbf{w}, SNR')$ to *batch*
end for

[Forward Computation]
$([\mathbf{x}_0], [\mathbf{c}], [t], [\nu], [\mathbf{w}], [SNR']) \leftarrow$ *batch*
$[\mathbf{s}] \leftarrow \mathbf{S}_\theta([\sqrt{1-\nu}] \odot [\mathbf{x}_0] + [\sqrt{\nu}] \odot [\mathbf{w}], [t], [\mathbf{c}])$ ♯ See also eq. (10).

[Back-propagation]
$\mathcal{L} \leftarrow \mathbb{E}_{\text{batch}}\left[[SNR'] \times \|[\mathbf{w}] - [\mathbf{s}]\|_2^2\right]$ ♯ SNR-weighted Loss (Kingma et al., 2021). See also eq. (10).
Compute $\nabla_\theta \mathcal{L}$ by the back-propagation.
$\theta \leftarrow Adam(\boldsymbol{\theta}, \nabla_\theta \mathcal{L})$ ♯ Update the parameters of $\mathbf{S}_\theta(\cdot, \cdot, \cdot)$.
end for
End
Output: The network parameter θ.

On the SNR weight In our case, the SNR is given by $SNR(t) = (1-\nu_t)/\nu_t$. The SNR-based weight is defined by its negative time derivative, so

$$SNR' = -\frac{d}{dt}\frac{1-\nu_t}{\nu_t} = \frac{\dot{\nu}_t}{\nu_t^2} = \frac{(1-\nu_t)\beta_t}{\nu_t^2}. \tag{145}$$

See also eq. (54).

F ADDITIONAL RESULTS OF IMAGE SYNTHESIS

F.1 COMPARISON OF THE STEP-SIZE SCHEDULE (CONSTANT VS EXPONENTIAL)

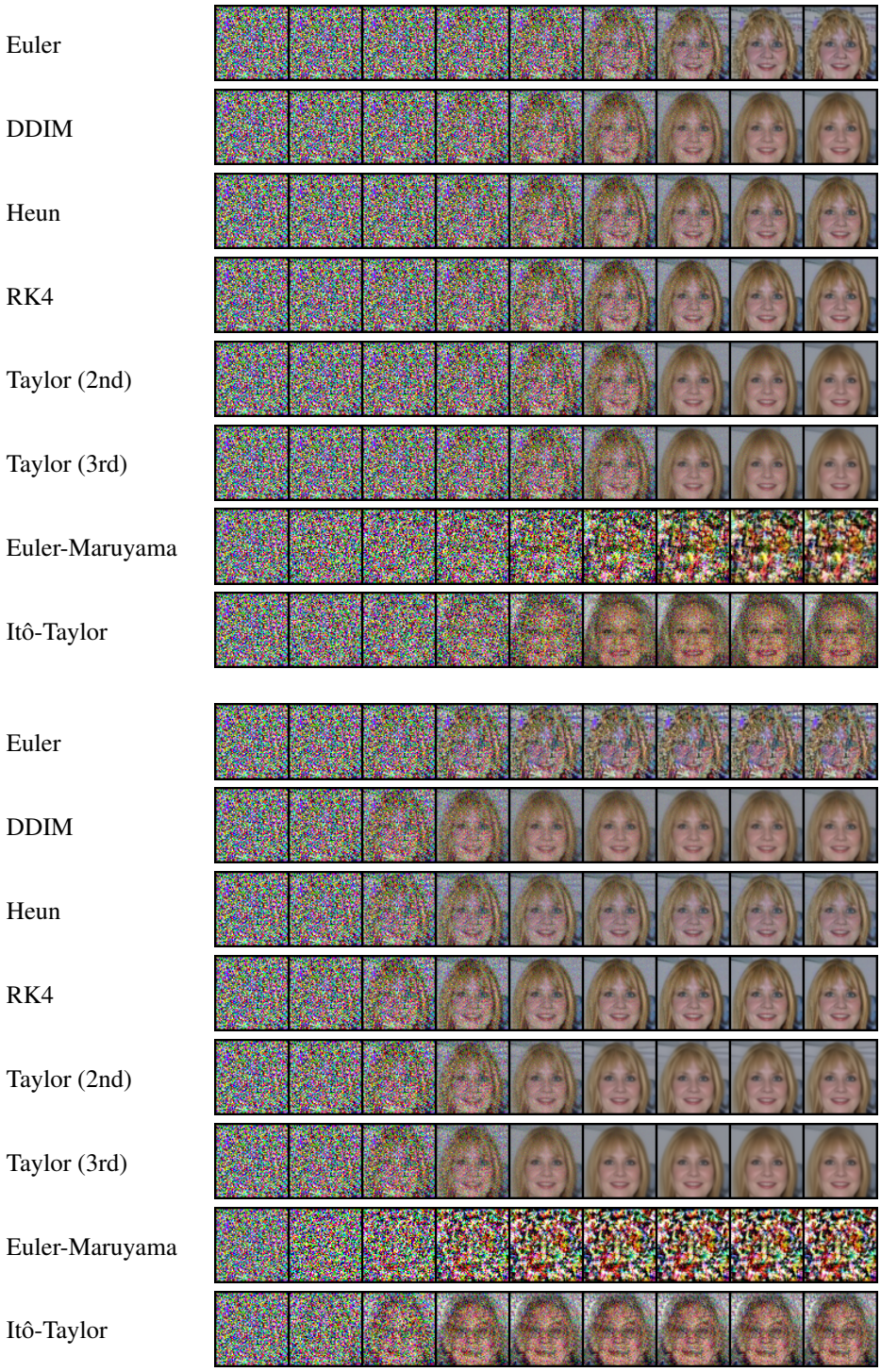


Figure 14: Comparison of the image synthesis process using different step-size schedules. above: constant schedule, below: exponential schedule. The number of refinement steps is $N = 8$. The synthesis noise schedule was condition (ii).

F.2 OTHER SAMPLING EXAMPLES

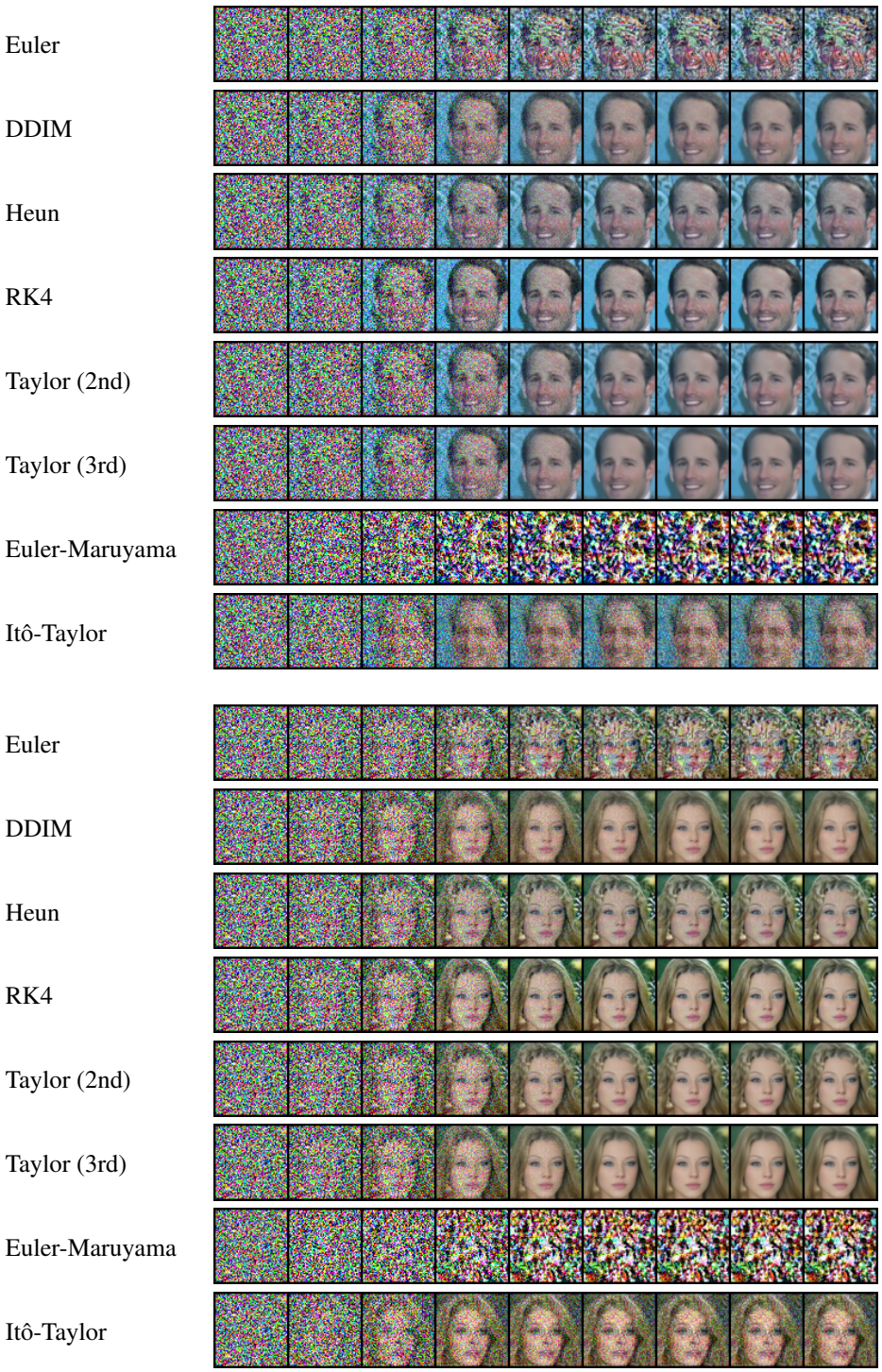
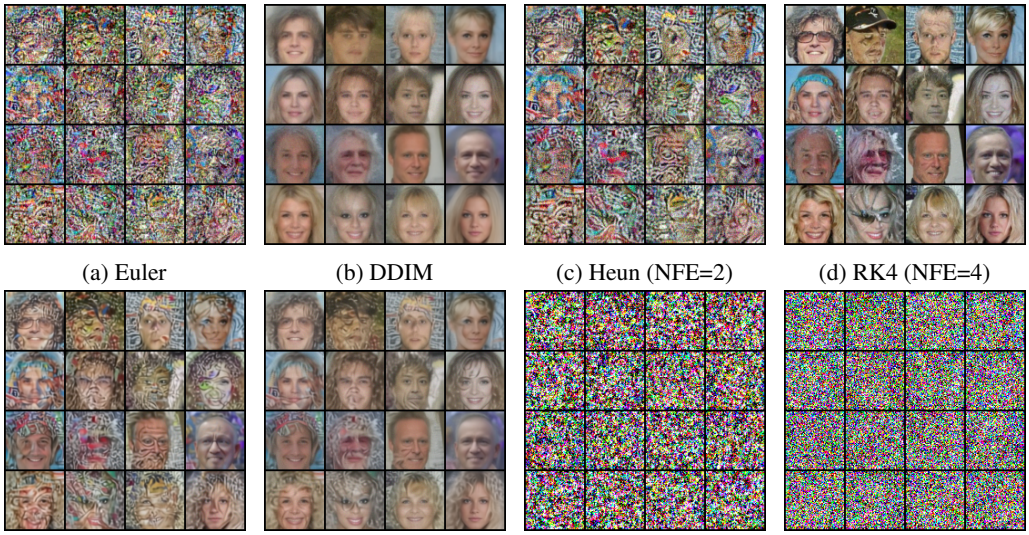


Table 5: Comparison of the image synthesis process. The dataset is CelebA (64×64). All the conditions but the sampling algorithm are the same including the random seed. The number of refinement steps is $N = 8$.

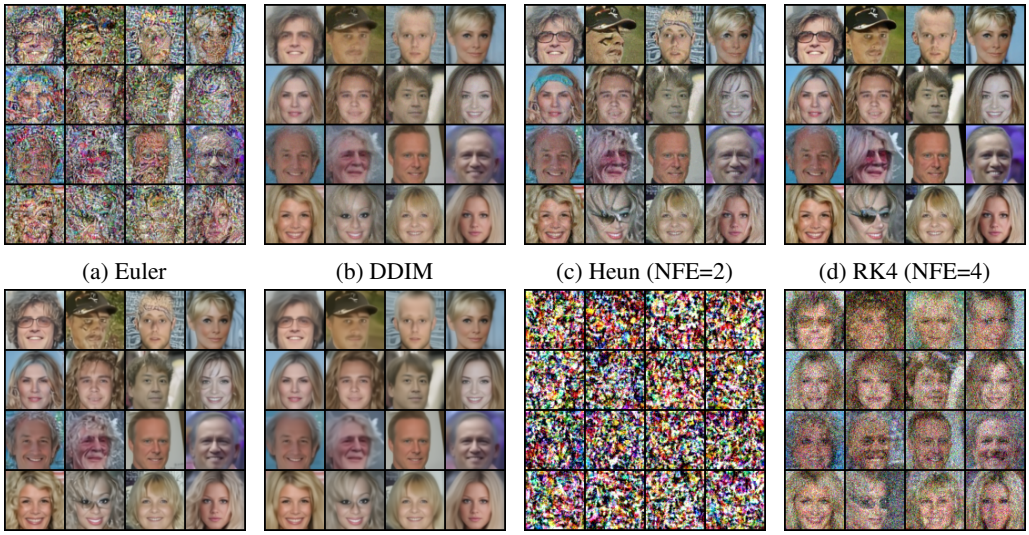
F.3 RANDOM SAMPLES



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

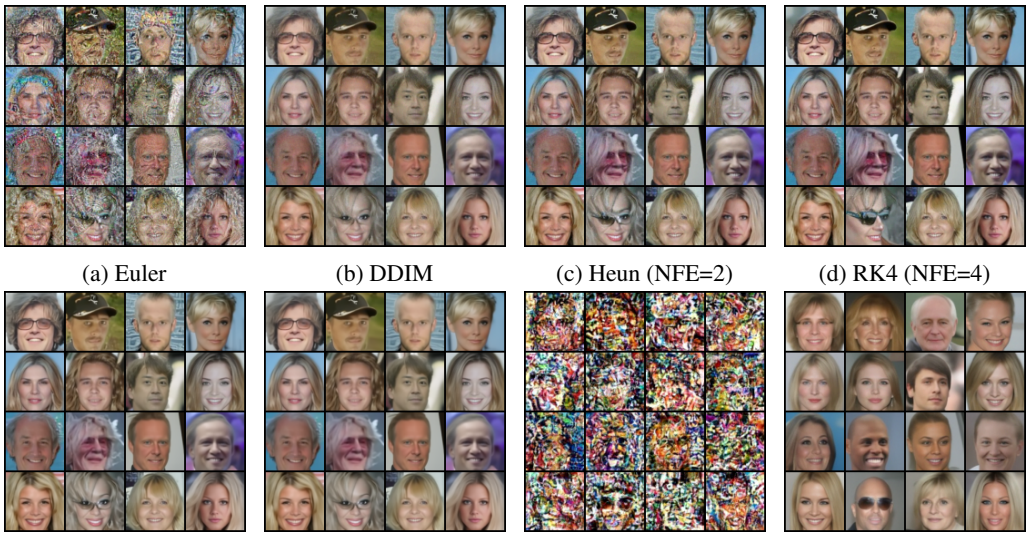
Figure 15: CelebA (64×64) synthesis samples. $\underline{N = 4}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

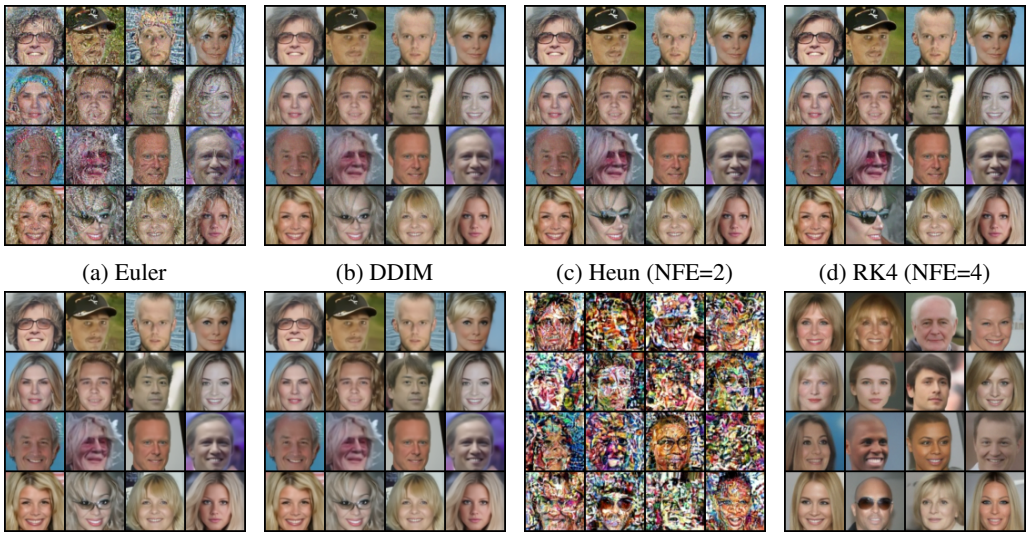
Figure 16: CelebA (64×64) synthesis samples. $\underline{N = 8}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

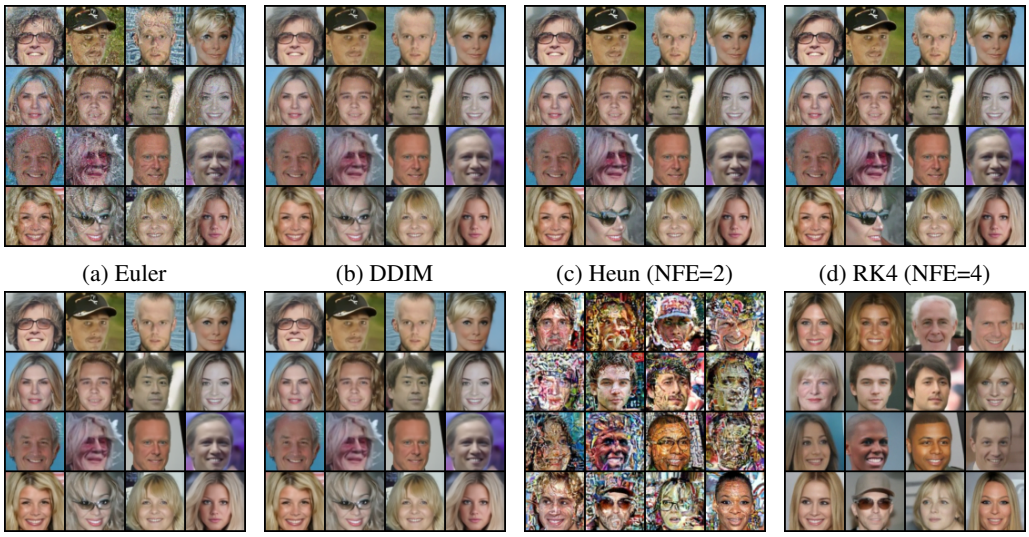
Figure 17: CelebA (64×64) synthesis samples. $\underline{N = 16}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

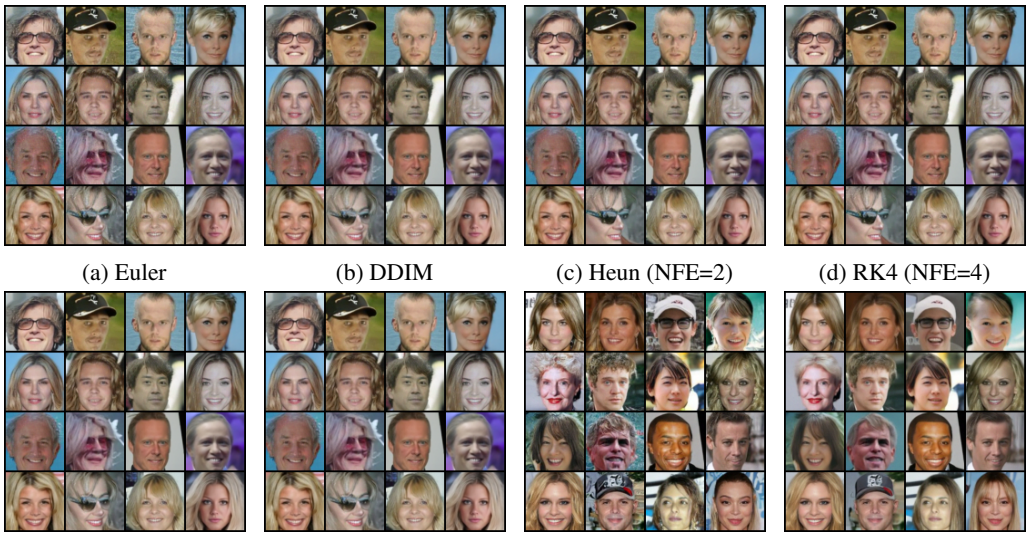
Figure 18: CelebA (64×64) synthesis samples. $\underline{N = 20}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

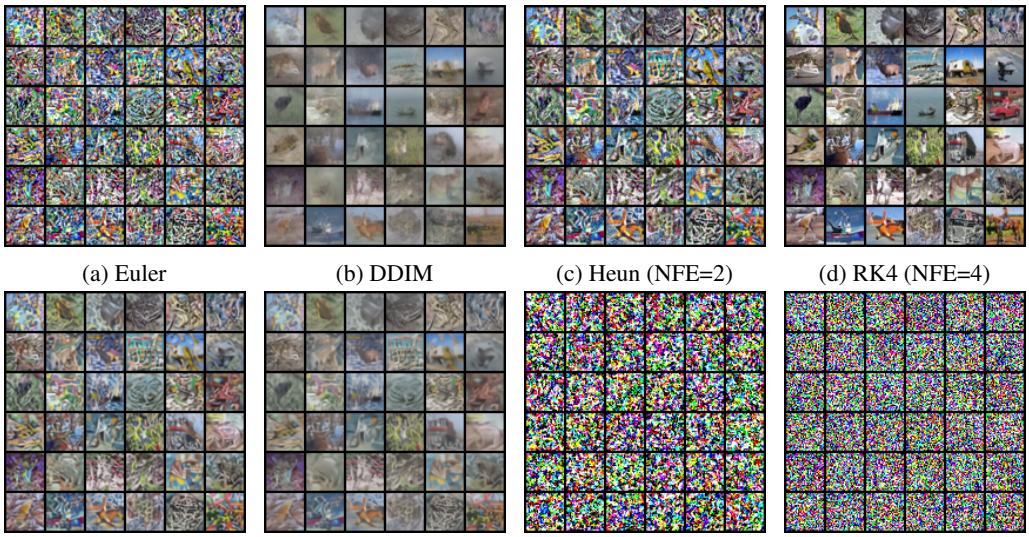
Figure 19: CelebA (64×64) synthesis samples. $\underline{N = 30}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

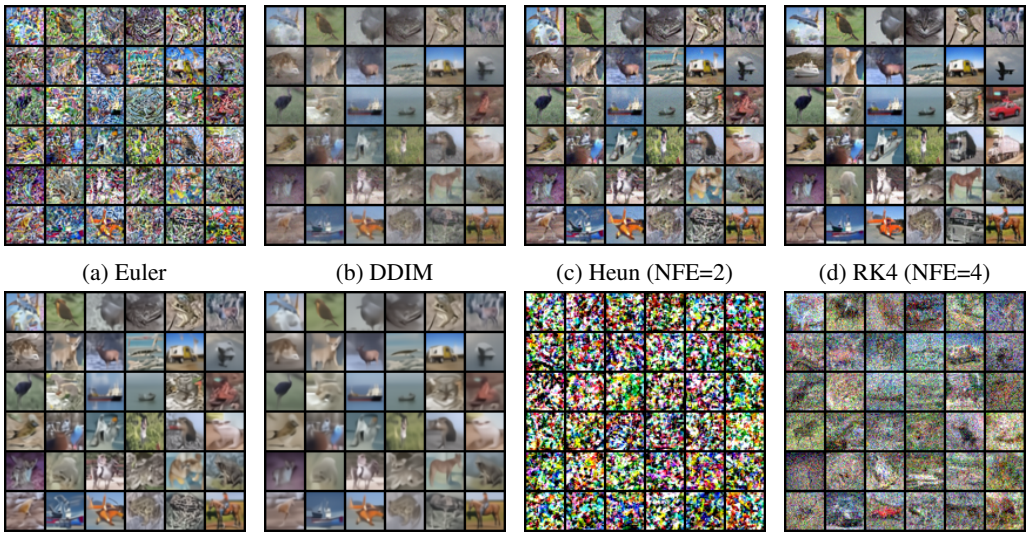
Figure 20: CelebA (64×64) synthesis samples. $\underline{N = 100}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

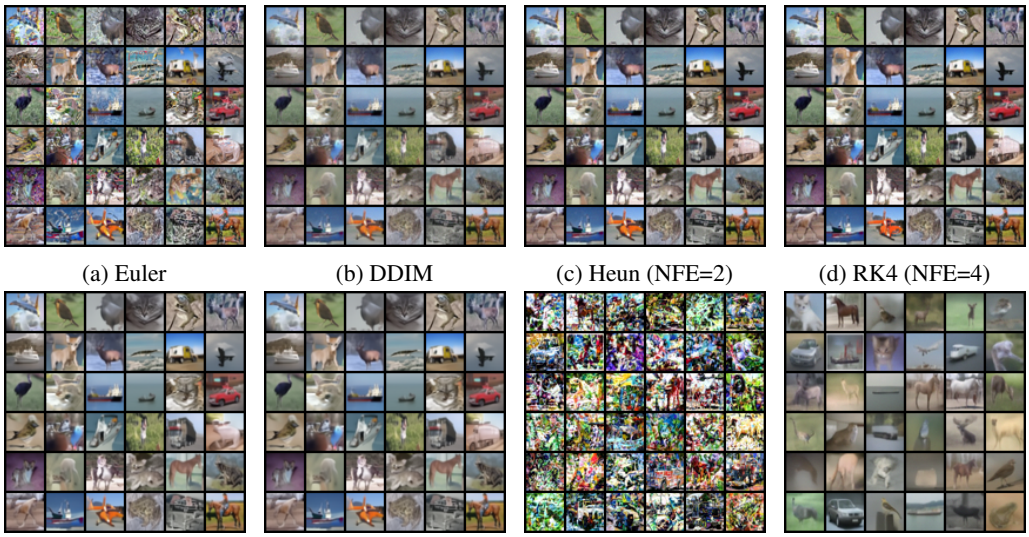
Figure 21: CIFAR-10(32×32) synthesis samples. $\underline{N = 4}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

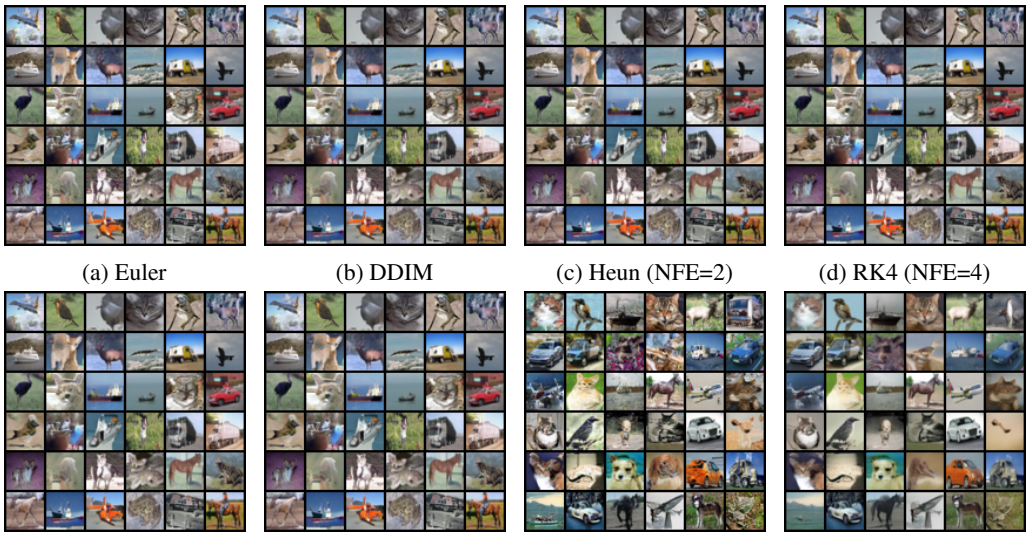
Figure 22: CIFAR-10 (32×32) synthesis samples. $\underline{N = 8}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

Figure 23: CIFAR-10 (32×32) synthesis samples. $\underline{N = 20}$. Noise schedule was (ii).



(a) Euler (b) DDIM (c) Heun (NFE=2) (d) RK4 (NFE=4)

(e) **Taylor (2nd)** (f) **Taylor (3rd)** (g) Euler-Maruyama (h) **Itô-Taylor**

Figure 24: CIFAR-10 (32×32) synthesis samples. $\underline{N = 100}$. Noise schedule was (ii).