Itô-Taylor Sampling Scheme for Denoising Diffusion Probabilistic Models using Ideal Derivatives

Hideyuki Tachibana       Mocho Go
Muneyoshi Inahara       Yotaro Katayama       Yotaro Watanabe
PKSHA Technology, Inc.
Hongo, Bunkyo City, Tokyo, Japan

Abstract

Denoising Diffusion Probabilistic Models (DDPMs) have been attracting attention recently as a new challenger to popular deep neural generative models including GAN, VAE, etc. However, DDPMs have a disadvantage that they often require a huge number of refinement steps during the synthesis. To address this problem, this paper proposes a new DDPM sampler based on a second-order numerical scheme for stochastic differential equations (SDEs), while the conventional sampler is based on a first-order numerical scheme. In general, it is not easy to compute the derivatives that are required in higher-order numerical schemes. However, in the case of DDPM, this difficulty is alleviated by the trick which the authors call “ideal derivative substitution”. The newly derived higher-order sampler was applied to both image and speech generation tasks, and it is experimentally observed that the proposed sampler could synthesize plausible images and audio signals in relatively smaller number of refinement steps.

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, VAEs, Normalizing Flows (NFs), and Autoregressive models. 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). We call these iterative refinement methods as the Denoising Diffusion Probabilistic Models (DDPMs) in this paper, following Ho et al. (2020). The DDPM-based generative models have recently been showing impressive results in many fields including image generation (Ho et al., 2020; Vahdat et al., 2021), image superresolution (Saharia et al., 2021; Ho et al., 2021), image translation (Sasaki et al., 2021), speech synthesis (Chen et al., 2020, 2021; Kong et al., 2021; Popov et al., 2021), symbolic music generation (Mittal et al., 2021), natural language generation (Hoogeboom et al., 2021; Austin et al., 2021), etc.

DDPM is unique in that it considers an iterative refinement, rather than a single shot synthesis (Figure 1). In other words, DDPM learns to improve the quality of an unfinished artwork. It would be intuitively natural to assume that the task of progressive improvement is easier to learn than completing an artwork in a single shot like GANs, VAEs and NFs are doing. A drawback of DDPM, however, is that it often requires very large number of, typically hundreds to thousands of, refinement steps during the synthesis. (However, it is possible to synthesize plausible data with a smaller number
of refinement steps if spatially-aligned conditional information is given; for example, the number of refinement steps was 100 for the super-resolution task (Saharia et al., 2021), and only 6 for the speech synthesis (vocoder) task (Chen et al., 2020).

The research question we are interested in in this paper is how to establish a systematic method to stably generate good data from DDPMs in a relatively small number of refinement steps, whether the task is conditional or unconditional. This is a common issue in DDPM studies, and there have been some studies aiming at improving the framework so that efficient sampling is possible (Song et al., 2020a; Nichol & Dhariwal, 2021; Dockhorn et al., 2021). In addition, there are also some studies aimed at improving the efficiency of the DDPM sampling, including (Jolicoeur-Martineau et al., 2021; Kong & Ping, 2021; San-Roman et al., 2021; Watson et al., 2021).

This paper takes the approach similar to the latter studies, and investigates a new sampler that generates samples from existing DDPMs. Our proposal technique is based on a second-order numerical scheme for stochastic differential equations (SDEs), which is supposed to be able to sample a data in a fewer steps (Figure 1) than existing first-order samplers. In our method, the network structures and the training objectives are the same as the existing DDPMs, but only the sampling scheme is different from the prior art.

The contribution of this paper would be three fold:

1) We propose a novel sampler for DDPMs based on a weak second-order SDE solver. We show that our new scheme is written in a closed form which is not very complicated, and can synthesize plausible data in relatively smaller number of refinement steps.

2) To derive the above sampling algorithm, we make effective use of the fact that the potential gradient \( \nabla \log p(x_t \mid x_0, t) \) (score function) has a special structure that allows us to compute its higher order derivatives easily. We call this technique ideal derivative substitution.

3) As a natural consequence of the above idea, we also propose a systematic method of designing the noise schedule. Our methodology is compliant with the Lipschitz condition for the SDE coefficient. The schedule ensures that all terms in the algorithm do not diverge.

2 Background: DDPM

2.1 DDPM Framework in SDE Viewpoint

Let us first briefly recall the overall framework of the DDPM. Following Song et al. (2020b), we describe the mechanisms using the language of stochastic differential equations (SDEs) for later convenience. Let us consider an Itô-type SDE (see e.g. Särkkä & Solin (2019); Øksendal (2013), etc.) as follows,

\[
x_t = x_0 + \int_0^t f(x_s, s) ds + \int_0^t g(s) dB_s
\]

where \( B_s \) is the Brownian motion (the Wiener process); the SDE is often formally written as

\[
dx_t = f(x_t, t) dt + g(t) dB_t.
\]

We often use the simplified notation for the partial derivatives \( \partial_x := \partial / \partial x \). We also use the following common notations: \( f := \partial_t f \), and \( \nabla f := \partial_x f \).
Starting from a data point \( x_0 \sim p(x_0) \), we can distort it by the forward noising SDE. After a sufficient amount of time, a Gaussian noise \( x_T \) that follows the standard multivariate Gaussian distribution \( \mathcal{N}(0, 1) \) is obtained, if the drift coefficient \( f(x, t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \) and the scalar diffusion coefficient \( g(t) : \mathbb{R} \to \mathbb{R} \) satisfy some conditions.

The forward noising process doesn’t make much sense by itself at this moment, as it is just distorting the clean data. Instead, what is more interesting is the backward process, i.e., the time reversal of this dynamics. The backward denoising process is given by the following SDE (Anderson, 1982):

\[
\begin{align*}
x_t &= x_T - \int_t^T \bar{f}(x_s, s) ds - \int_t^T g(s) dB_s, \\
\bar{f}(x, t) &= f(x, t) - g(t)^2 \nabla \log p(x | x_0, t).
\end{align*}
\]

From now on, to avoid confusion between the forward and backward processes, we will use the the antinoise level. By the backward denoising dynamics, a Gaussian noise \( x_T \sim x_0 \sim \mathcal{N}(0, 1) \) goes to a sample \( y_T \) that follows the “true density” \( p(x_0) \), at \( t \to 0 (\tau \to T) \).

In order to perform this calculation, we need to specify the undefined functions explicitly. In addition, as the continuous model above is computationally intractable, we need to discretize it. We will look at each of them in the following sections. (i) Specify the drift \( f(x, t) \) and the noising schedule \( g(t) \) in § 2.2. (ii) Specify the potential gradient (score function) \( \nabla \log p(x | x_0, t) \) in § 2.3 (iii) Discretize Eq. (4), and compute the path numerically in § 2.4 and § 2.3.

### 2.2 Forward Noising Process

#### 2.2.1 Closed-form Expression

The forward noising dynamics may seem meaningless at first glance, but in fact it is very important. This is because the functions \( f(x, t) \) and \( g(t) \) we consider here determine whether the backward dynamics is tractable or not. Firstly, it is convenient defining the drift \( f(x, t) \) as follows,

\[
f(x, t) := \frac{\sqrt{1 - g(t)^2} h - 1}{h} x_t = g(t)^2 x_t + O(h),
\]

where \( h > 0 \) is the step size. This allows us to derive a simple \( p(x_t | x_0, t) \) with an explicit closed form, by making use of the reproductive property of diagonal Gaussian variables. In specific, by applying the discretized forward process recursively, we obtain a tractable \( p(x_t | x_0, t) \) as follows,

\[
p(x_t | x_0, t) = \mathcal{N}(x_t | \sqrt{\alpha(t_n)} x_0; (1 - \alpha(t_n)) 1),
\]

where \( \alpha(t_n) = \prod_{m=1}^n (1 - g(t_m)^2 h) \).

\[
(Proof: \text{By induction; see § B}) \text{ Let us refer to the factor } 1 - \alpha(t) \text{ the noise level, and } \alpha(t) \text{ the antinoise level.}
\]

#### 2.2.2 Continuous Noising Functions

Next, let us examine the diffusion coefficient \( g(t) \) and the antinoise level \( \alpha(t) \) more in detail. In the prior art, \( g(t) \) and \( \alpha(t) \) have been defined on a discrete domain \( t \in \{0, t_1, t_2, \ldots, T\} \). However, it is more convenient defining them on a continuous domain \( t \in \mathbb{R}_{\geq 0} \), as it allows us to perform various calculations symbolically. To do that, let us consider the infinitesimal time step \( h \to 0 \). Then, the antinoise level \( \alpha(t) \) is analytically given by the Volterra product integral (see § C), and conversely, if \( \alpha(t) \) is designed first, we can compute \( g(t) \) from \( \alpha(t) \) as follows,

\[
\alpha(t) = \exp \int_0^t -g(\tau)^2 d\tau,
\]

\[
g(t) = \sqrt{-\frac{d}{dt} \log \alpha(t)}.
\]
Remark 1. Note, not all $\alpha(t)$ and $g(t)$ that have the above relation are eligible. As we shall see later (Remark 2), the backward drift coefficient $\bar{f}(y, t)$ has a term supposedly proportional to $g(t)^2(1 - \alpha(t))^{-1}y$. Since the drift coefficient of SDE should be Lipschitz w.r.t. $y$, the factor is required to be bounded above by a finite constant. That is, there should exist a finite constant $M < \infty$ s.t.,

$$\text{for all } t \ (0 \leq t \leq T), \quad \left| \frac{g(t)^2}{1 - \alpha(t)} \right| < M. \tag{10}$$

Therefore, the simplest noising schedules which have been commonly used, such as the linear noising $g(t) = t$, $\alpha(t) = \exp(-t^3/3)$ and the constant noising $g(t) = 1$, $\alpha(t) = \exp(-t)$, are not eligible, since they do not satisfy the Lipschitz condition above.

2.2.3 Parameterized Expressions for $g(t), \alpha(t)$

For later convenience, let us derive parameterized expressions for $g(t)$ and $\alpha(t)$. As we shall later meet $g(t)$ and $\alpha(t)$ often in the form of $g(t)^2/\sqrt{1 - \alpha(t)}$, it is convenient to design this factor directly. Let us write the factor as $\tilde{\lambda}(t)$. Then, we immediately have the differential equation

$$\dot{\alpha}(t) + \tilde{\lambda}(t)\alpha(t)\sqrt{1 - \alpha(t)} = 0, \tag{11}$$

and we also obtain the parameterized expressions for $\alpha(t), g(t)$ as follows by solving it,

$$\alpha(t) = \frac{1}{\cosh^2(\tilde{\lambda}(t)/2)} = 1 - \tanh^2 \frac{\tilde{\lambda}(t)}{2}, \tag{12}$$

$$g(t) = \sqrt{\tilde{\lambda}(t) \tanh \frac{\tilde{\lambda}(t)}{2}}. \tag{13}$$

From these expressions we can verify that the density $p(x_t \mid x_0, t)$ finally goes to the standard Gaussian $\mathcal{N}(0, 1)$ if $\tilde{\lambda}(t)$ is designed so that it becomes sufficiently large as $t$ increases. Here, we should note again that the function $\tilde{\lambda}(t)$ is not arbitrary because of the reason mentioned; the factor $g(t)^2(1 - \alpha(t))^{-1} = \tilde{\lambda}(t)/\tanh(\tilde{\lambda}(t)/2)$ should be finite. We will define a specific form of $\tilde{\lambda}(t)$ in §3.3.

2.3 Training Objective

One of the challenges of DDPMs is the evaluation of the term $\nabla \log p(x \mid x_0, t)$ in the backward denoising process. It is generally intractable, but fortunately, thanks to the discussion in the previous section, i.e. Eq. (6), the term $\nabla \log p(x \mid x_0, t)$ is actually written in a rather simple form as follows,

$$\nabla \log p(x \mid x_0, t) = \frac{-1}{\sqrt{1 - \alpha(t)}} \times \frac{x - \sqrt{\alpha(t)}x_0}{\sqrt{1 - \alpha(t)}}. \tag{14}$$

In DDPM, one estimates the second factor using a deep neural network\(^2\) $S(x, t)$. That is, the network $S(x_t, t)$ should predict $(x_t - \sqrt{\alpha(t)}x_0)/\sqrt{1 - \alpha(t)} \sim \mathcal{N}(0, 1)$. In other words, the following approximate equality should hold for any Gaussian noise $w \sim \mathcal{N}(0, 1)$ and any data point $x_0 \sim p_{\text{data}}$.

$$S(\sqrt{\alpha(t)}x_0 + \sqrt{1 - \alpha(t)}w, t) \approx w. \tag{15}$$

(Note that the random variable $w$ in the l.h.s. and r.h.s. are the same.) The simplest idea to train such a machine learning model $S(x_t, t)$ is to minimize the mean squared error between the both sides, and in fact, this straightforward idea can actually be justified in terms of learning theory. Ho et al. (2020) have shown that the minimization of the $L^2$ error is essentially understood as the ELBO maximization if some factors are ignored, and argued that the following training objective works in practice,

$$\mathcal{L} := \mathbb{E} \left[ \left\| w - S(\sqrt{\alpha(t)}x_0 + \sqrt{1 - \alpha(t)}w, t) \right\|^2_2 \right], \tag{16}$$

\(^2\)The network can take a conditioning parameter $c$ (e.g. low-resolution images, mel-spectrograms, class tags, etc.), and is actually be the form of $S(x, t, c)$. However, we will not show that for readability in this paper. Additionally, the network is sometimes implemented as $S(x, \alpha(t))$, instead of $S(x, t)$, but we always used the latter notation.
where the expectation is taken w.r.t. $x_0 \sim p_{\text{data}}$, $w \sim \mathcal{N}(0, 1)$, and $t \sim \text{Uniform}([0, T])$. This sort of learning has been referred to as the score matching [Hyvärinen & Dayan, 2005; Vincent, 2011]. 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.

### 2.4 Sampling by a Discretized Backward Process

Using the terms we have described above, we are able to draw a chain of samples $\{y_0, y_h, \cdots, y_T\}$ by a numerical scheme. One of the simplest numerical schemes for SDEs is the Euler-Maruyama method [Maruyama, 1955] Theorem 1). Despite its simplicity, the Euler-Maruyama scheme is sufficiently effective in many applications if the step size $h > 0$ is sufficiently small. Many of existing DDPM samplers could be considered to be based on the Euler-Maruyama sampling, even though it were not explicitly mentioned. By adopting the Euler-Maruyama method to Eq. (4), we obtain a sampling algorithm as follows.

**Algorithm 1** (Euler-Maruyama Sampling Scheme). Starting from a Gaussian random variable $y_0 \sim \mathcal{N}(0, 1)$, we may draw a chain of samples $\{y_0, y_h, \cdots, y_T\}$, by the following recursion,

$$y_{\tau+h} = \left(1 + \frac{g(t)^2 h}{2}\right)y_{\tau} - \frac{g(t)^2 h}{\sqrt{1-\alpha(t)}}S(y_{\tau}, t) - g(t)\sqrt{h}w_{\tau}$$

where $t = T - \tau$, and $w_{\tau}$ is a Gaussian variable drawn from $\mathcal{N}(0, 1)$.

If the network $S(y_{\tau}, t)$ accurately predicts the true potential gradient, the step size $h$ is sufficiently small, and the terminal time $T$ is sufficiently large, then the final outcome $y_T$ obtained by this algorithm will approximately follow the true density $p(x_0)$. Now we are interested in how the quality of the outcome depends on the step size $h > 0$. The answer is that the error from the continuous limit $h \rightarrow 0$ is the order of $O(h)$ on average. (See §A) In other words, if we use a larger step size $h$, the error roughly increases proportionally to $h$ if other factors are fixed.

### 3 Proposed Sampler

#### 3.1 Motivation: Higher-order Numerical Schemes

Now the research question is how we can raise the order of the numerical scheme to make the sampling faster and more accurate. A simple idea would be to adopt higher-order discretization schemes for deterministic ODEs e.g. the Runge-Kutta methods. However, this will not work effectively in the SDE cases without careful consideration. In fact, it is known that an ad hoc adaptation of the deterministic RK methods do not yield valid higher-order schemes for SDEs (Kloeden et al., 1994 § 4.2);(Burrage et al., 2006);(Särkkä & Solin 2019 § 8.5). It is also known that stochastic numerical schemes driven only by the Brownian increments (additive Gaussian noise) cannot surpass the strong order 1, i.e. $O(h)$, on average (Clark & Cameron, 1980; Rüemelin, 1982).

To date, avoiding such pitfalls, a number of RK-type schemes for SDEs have been developed (Pardoux & Talay, 1985; Newton, 1991; Burrage & Burrage, 1998; Tocino & Vigo-Aguiar, 2002; Milstein & Tretyakov, 2004; Roberts, 2006, 2010). However, higher-order stochastic RK schemes are much more complicated than the deterministic counterparts. In addition, a more critical problem is that the RK schemes require the drift coefficients at many different points, but it is too expensive in our case because the drift term of the backward dynamics contains the deep neural network $S(x, t)$.

Nevertheless, we do not have to give up on the higher-order schemes because RK methods are not the only options. Let us recall the derivation of the RK method; it is derived by approximating higher-order derivatives by linear combinations of function values, and determining the coefficients so that they are consistent. That is, if it is easy to evaluate the derivatives, we can simply employ them. In this section, we shall see that the derivatives are easily estimated in our case, and use them to construct a supposedly higher-order algorithm.
3.2 Itô-Taylor Expansion

In this section, let us consider the 1+1-dimensional case (1-dim space \( \times \) 1-dim time) for simplicity. Since all dimensions of \( x_t \in \mathbb{R}^d \) are separable in our case, we can easily generalize the argument to the vectorized (\( d+1 \)-dimensional) version by parallelizing each dimension.

In deterministic numerical schemes, it is common to consider the Taylor expansion of the system. Let us first consider a deterministic system \( \dot{x}(t) = a(x(t), t) \). Using a differential operator \( L := (\partial_t + a(t, x)\partial_x) \), we can write the Taylor expansion of the path \( x(t) \) as follows.

\[
x(t + h) = x(t) + h a(x, t) + \frac{h^2}{2!} L a(x, t) + \cdots.
\]  

(18)

If we ignore \( o(h^n) \) terms of the series, we may obtain a numerical scheme of order \( p \).

In stochastic systems, the Taylor expansion requires modifications. If \( x_t \) obeys the following stochastic system,

\[
dx_t = a(x_t, t)dt + b(x_t, t)dB_t,
\]  

(19)

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). See also § D. The Itô-Taylor expansion is based on the following differential operators \( L,G \), which are based on the celebrated Itô’s formula (Itô, 1944).

\[
\begin{align*}
L &= \partial_t + a(x, t)\partial_x + \frac{1}{2} b(x, t)^2 \partial_x^2 \\
G &= b(x, t)\partial_x
\end{align*}
\]  

(20)

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.

**Theorem 1** [Kloeden & Platen (1992) § 14.2]: An Itô-Taylor scheme of weak order \( \beta = 2 \). Let \( x_t \) obeys the Itô SDE Eq. (19) and let the differential operators \( L, G \) be given by Eq. (20). Then, the following numerical scheme weakly converges with the order of \( \beta = 2 \).

\[
x_{t+h} = x_t + h a + \tilde{w}_t b + \frac{\tilde{w}_t^2 - h}{2} Gb + \frac{h^2}{2} La + (\tilde{w}_t h - \tilde{z}_t) Lb + \tilde{z}_t Ga
\]  

(21)

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

\[
\begin{bmatrix}
w_t \\
z_t
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 \\
1/2 & 1/2\sqrt{3}
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix}
\]  

(22)

where \( u_1, u_2 \sim \mathcal{N}(0, 1) \). The notations \( a, La, etc. \) are the abbreviations for \( a(x_t, t), (La)(x_t, t), etc. \).

This scheme has the order \( \beta = 2 \) of weak convergence. In addition, in a special case where \( G^2 b = 0 \), the strong convergence of order \( \gamma = 1.5 \) is also guaranteed (Kloeden & Platen, 1992 § 10.4). Note that, if we used only the beginning three terms of Eq. (21), it is equivalent to the Euler-Maruyama method.

3.3 Itô-Taylor DDPM with Ideal Derivatives

In the backward process of DDPM Eq. (4) which we are interested in, the operators are written as follows,

\[
\begin{align*}
L &= \partial_t - \bar{f}(y, t)\partial_y + \frac{1}{2} g(t)^2 \partial_y^2 \\
G &= -g(t)\partial_y
\end{align*}
\]  

(23)

where \( t = T - \tau \) and \( \bar{f}(y, t) \) is

\[
\bar{f}(y, t) = -\frac{g(t)^2}{2} y + \frac{g(t)^2}{\sqrt{1 - \alpha(t)}} S(y, t).
\]  

(24)

It is not easy in general to evaluate expressions involving such many derivatives. Indeed, in \( L(-\bar{f}) \), there appear the derivatives \( \partial_x S(y, t) \) and \( \partial_y S(y, t) \) of a neural network, 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 terms above explicitly. Remembering that the network \( S(x, t) \) is trained so that it approximates \( (x - \sqrt{\alpha(t)} x_0)/\sqrt{1 - \alpha(t)} \), we may assume that the derivatives should ideally hold following equalities. (For derivation, see § E).
We may rewrite each term of Eq. (29) using $\lambda$

*At first glance, this algorithm may appear to be very complex. However, the computational complexity

**Algorithm 2**

The derivative $y$ follows the true density.

**Remark 2.** The ideal spatial derivative implies that the spatial derivative of backward drift $a = -\bar f$, i.e.,

$$\partial_y (-\bar f(y, t)) = \partial_y \left( \frac{g(t)^2}{2} y - \frac{g(t)^2}{\sqrt{1 - \alpha(t)}} S(y, t) \right),$$

has a term which is ideally proportional to $g(t)^2/(1 - \alpha(t))$. As mentioned before, the spatial derivative of drift should be bounded. Therefore, the factor should be finite for every $t$ of our interest, particularly near $t \sim 0$.

Using the above ideal derivatives, we can compute symbolic expressions for $L(-\bar f), G(-\bar f), L(-g),$ and $G(-g)$ as follows,

$$
\begin{align*}
L(-\bar f) &= \left( \frac{g(t)^4}{4} - g(t)\dot g(t) \right) y + \frac{2g(t)\dot g(t)}{\sqrt{1 - \alpha(t)}} S(y, t) \\
G(-\bar f) &= \frac{g(t)^3(1 + \alpha(t))}{2(1 - \alpha(t))} \\
L(-g) &= \dot g(t), \quad G(-g) = 0
\end{align*}
$$

and we can derive the following algorithm using Theorem.

**Algorithm 2** (Itô-Taylor Sampling Scheme with Ideal Derivatives). Starting from a Gaussian noise $y_0 \sim \mathcal{N}(0, 1)$, the following iterative refinement generates a data point $y_T$ that approximately follows the true density.

$$y_{t+h} = \rho(t, h)y_t + \mu(t, h)S(y_t, t) + n(t, h)$$

where each coefficient is given as follows,

$$
\begin{align*}
\rho(t, h) &= 1 + \frac{g(t)^2}{2} h + \frac{1}{2} \frac{g(t)^4}{4} - g(t)\dot g(t) \\
\mu(t, h) &= -\frac{g(t)^2}{\sqrt{1 - \alpha(t)}} h + \frac{g(t)\dot g(t)}{\sqrt{1 - \alpha(t)}} h^2 \\
n(t, h) &= -g(t)\sqrt{h}w_t + h^3/2 \left( (w_t - z_t)\dot g(t) + \frac{g(t)^3(1 + \alpha(t))}{2(1 - \alpha(t))} z_t \right)
\end{align*}
$$

where $t = T - \tau$. The Gaussian variables $w_t$ and $z_t$ have dimension-wise correlations, and each dimension is sampled similarly to Theorem.

We may rewrite each term of Eq. (29) using $\lambda(t)$ which was introduced in $\S$ 2.2. The diffusion $g(t)$ is already given in Eq. (13), and the ratio $g(t)^2/\sqrt{1 - \alpha(t)}$ is $\dot \lambda(t)$ by definition. Other composite terms are written as follows,

$$
\begin{align*}
g(t)\dot g(t) &= \frac{\dot \lambda(t)}{2} \tanh \frac{\lambda(t)}{2} + \frac{\dot \lambda(t)^2}{4 \cosh^2(\lambda(t)/2)} \\
\frac{g(t)\dot g(t)}{\sqrt{1 - \alpha(t)}} &= \frac{\dot \lambda(t)}{2} + \frac{\dot \lambda(t)^2}{2 \sinh \lambda(t)} \\
\frac{g(t)^3(1 + \alpha(t))}{2(1 - \alpha(t))} &= \sqrt{\frac{\lambda(t)^3}{\tanh(\lambda(t)/2)}} - \frac{g(t)^3}{2}
\end{align*}
$$

The derivative $\dot g(t)$ has a little complicated form, but we can compute it by $\dot g(t) = (g(t)\dot g(t))/g(t)$.

At first glance, this algorithm may appear to be very complex. However, the computational complexity hardly increases compared to the Euler-Maruyama method, because almost all of the computational cost is accounted for by the neural network $S(y, t)$, and the costs for scalar values $\rho(t, h), \mu(t, h)$ and noise generation $n(t, h)$ are almost negligible in this algorithm.
3.4 Specific Schedule of $\lambda(t)$

Now let us specify the function form of $\lambda(t)$. The function $\lambda(t)$ should satisfy the following three conditions.

(C1) It should make the factor $\varphi(t^2) = \frac{\lambda(t)}{\tanh(\lambda(t)/2)}$ bounded to make the drift $\bar{f}(x, t)$ Lipschitz w.r.t. $x$. It should also make each term which appears in Algorithm 2 finite. To satisfy these requirement, it is sufficient that $\lambda(t)$ is exponential near $t \sim 0$, and at most linear when $t$ becomes larger.

(C2) It should infinitely increase $\lim_{t \to \infty} \lambda(t) = \infty$ so that the antinoise level goes to 0, i.e., $\lim_{t \to \infty} \alpha(t) = 0$.

(C3) Eq. (5) implies $\alpha(0) = 1$, and it immediately implies $\lambda(0) = 0$. However, we will allow some relaxation of this condition. It should be $\lambda(t) \sim 0$ near $t \sim 0$.

One of the simplest functions that satisfies these requirements is a softplus function as follows.

$$\lambda(t) = \log(1 + Ae^{kt}), \quad A, k > 0.$$  

This function is also advantageous that its derivatives $\dot{\lambda}(t), \ddot{\lambda}(t)$ are easily computed, and the parameters $A, k$ are easy to tune. They are uniquely specified by using the inverse formula of Eq. (12) as follows, noting that \(\operatorname{arcosh} x = \log(x + \sqrt{x^2 - 1})\).

$$\lambda(t) = 2 \log(\sqrt{1 - \alpha(t)} + 1) - \log \alpha(t).$$

If we specify two specific values of $\alpha(t)$ for different $t$-s, e.g. the initial and terminal antinoise levels $\alpha(T), \alpha(0)$, then we obtain the parameters in closed forms as follows.

$$\left\{ \begin{array}{l} A = \frac{2(1 - \alpha(0)) + 2\sqrt{1 - \alpha(0)}}{\alpha(0)} \\ k = \frac{1}{T} \left( \log \frac{2(1 - \alpha(T)) + 2\sqrt{1 - \alpha(T)}}{\alpha(T)} - \log A \right) \end{array} \right.$$  

We now have all the information we need to run Algorithm 2. We summarize this entire process as pseudocode in §H.

4 Unconditional Image Synthesis

In this section, let us conduct an experiment to verify the effectiveness of the method developed in this paper (Algorithm 2) by using it as a sampler for an existing DDPM model for an unconditional image generation task. The baseline sampler is the conventional one (Algorithm 1).

The training dataset we used was CelebA (Liu et al. 2015), and the images were resized to $64 \times 64$. The network structure of $S(x, t)$ and the experiment code were based on the official implementation provided by Song et al. (2020b). The loss function we used was the $L^1$ loss instead of the $L^2$ loss (Eq. (16)), following Chen et al. (2020). The noising schedule is shown in Table 1. The batch size was 256. We trained the model for 100,000 training steps using 4 GPUs.

Figure 1 shows an example process of generating an image from noise. Figure 2 shows random samples generated by the proposed and Euler-Maruyama samplers. For both cases, the number of refinement steps was $N = 15$. We may observe that the proposed method clearly succeeds in producing clearer images than the conventional method. More examples are available in §I.

It can also be confirmed that both methods produce roughly similar images thanks to the use of the same initial values and the same random numbers. This would be interpreted as a reflection of the fact that the proposed method is a better approximation of the same continuous stochastic dynamics Eq. (4) than the Euler-Maruyama scheme.

As a quality assessment metric, we used the Fréchet Inception Distance (FID) (Heusel et al. 2017; Seitzer 2020). Figure 3 plots how the FID improves as the number of refinement steps $N = \frac{T}{h}$ is increased for each sampler. 20,000 images were randomly generated for each condition to compute the FID scores. We may clearly observe that the proposed sampler can achieve better FID scores much faster than the Euler-Maruyama sampler does.
Table 1: Parameters (initial and terminal antinoise levels) for both proposed (WaveGRIT) and Euler-Maruyama samplers.

| Task    | N (# steps) | T  | $\alpha(0)$  | $\alpha(T)$ |
|---------|-------------|----|--------------|-------------|
| Image   | any         | 1  | $1 - 1 \times 10^{-4}$ | 0.01        |
| Speech  | 25          | 1  | $1 - 1 \times 10^{-7}$ | 0.001       |
|         | 6           | 1  | $1 - 6 \times 10^{-5}$ | 0.9         |
|         | 4           | 1  | $1 - 5 \times 10^{-4}$ | 0.915       |

(a) Itô-Taylor sampler with ideal derivatives (proposed)  
(b) Euler-Maruyama sampler

Figure 2: Comparison of the synthesis results for the proposed and Euler-Maruyama samplers. The dataset is CelebA ($64 \times 64$). All conditions are the same except for the sampling algorithm: the same DDPM checkpoint, the same hyperparameters of noising schedule ($g(t), \alpha(t)$), the same number of steps ($N = 15$), the same initial Gaussian noise, and the same random seed.

Figure 3: Comparison of $N$-FID relations for each sampler. Note that the computation cost of a single step is almost the same for both samplers.
Table 2: Comparison of the mean opinion score (MOS), spectral distance from the ground truth spectrogram (SD), generalized Kullback-Leibler divergence (GKL), reversed GKL (RGKL), and real time factor (RTF), for each WaveGrad sampler. MOS for ground truth data was $4.87 \pm 0.35$.

| Sampler            | N (#steps) | MOS ↑ | SD ↓ | GKL ↓ | RGKL ↓ | RTF ↓ |
|--------------------|------------|-------|------|-------|--------|-------|
| Vovk’s schedule    | 25         | 3.33  | 0.454| 0.544 | 0.459  | 0.208 |
| Euler-Maruyama     | 25         | 2.97  | 0.445| 0.490 | 0.485  | 0.213 |
| WaveGRIT (Proposed)| 25         | 4.10  | 0.418| 0.544 | 0.459  | 0.208 |
| Vovk’s schedule    | 6          | 2.00  | 1.104| 2.072 | 5.239  | 0.047 |
| Euler-Maruyama     | 6          | 1.33  | 0.734| 1.004 | 2.339  | 0.048 |
| WaveGRIT (Proposed)| 6          | 3.80  | 0.638| 0.911 | 1.400  | 0.049 |
| Euler-Maruyama     | 4          | 1.03  | 1.095| 1.739 | 7.522  | 0.030 |
| WaveGRIT (Proposed)| 4          | 2.87  | 0.742| 1.134 | 2.117  | 0.031 |

5 Conditional Speech Synthesis (WaveGRIT)

Let us next adopt the sampler to the speech domain. In modern neural text-to-speech (TTS) systems, it is common to use two modules; one converts an input linguistic features (e.g. phonemes) to intermediate acoustic features (e.g. mel-spectrogram), and the other is the vocoder which converts the features to the waveform. In recent years, a number of neural vocoders have been proposed (van den Oord et al., 2016; Prenger et al., 2019; Yamamoto et al., 2020). The WaveGrad vocoder (Chen et al., 2020) is one of such vocoders, which is based on the DDPM conditioned on acoustic features (mel-spectrograms). In this section, we simply adapted our sampling algorithm to the WaveGrad, and call the method WaveGRIT (WaveGrad + Itô-Taylor).

Let us compare the performance of WaveGRIT with the sampling schedule tuned by Vovk (2020). We implemented our samplers on top of an unofficial open implementation of WaveGrad by Vovk (2020). Instead of retraining the model, we used an existing pretrained checkpoint provided in Vovk’s repository, which was trained on the LJSpeech dataset (Ito & Johnson, 2017), a popular dataset in TTS studies. We used this pretrained checkpoint as is, and did not retrain nor fine-tune the network parameters at all. The parameters for the Euler-Maruyama and WaveGRIT samplers are shown in Table 1. For more details on the experimental setup, see §E.

Figure 4 shows some mel-spectrograms of the synthesized audio signals. We may observe that the noise level of the proposed method is lower than the first-order samplers (Vovk’s schedule and the Euler-Maruyama sampler). We can also observe that the WaveGRIT synthesized clearer overtones (higher harmonics) than Vovk’s sampler. Additional results are available in §I.

Table 2 compares the results of subjective (MOS) and objective (SD, GKL, RGKL) evaluations of the quality of the speech synthesized by each sampler, as well as the synthesis speed (RTF). It clearly shows the effectiveness of the proposed sampler.

6 Concluding Remarks

One of the biggest drawbacks of the DDPMs has been that they often require a huge number of refinement steps. To address this problem, this paper developed a supposedly second order sampler...
for existing DDPM models, which enables faster sampling than existing first order counterparts (Euler-Maruyama schemes). It is expected that the fast sampling method will further expand the various application possibilities of the DDPM.

The idea for the proposed sampler is based on the ideal derivative substitution. Although this substitution may not be mathematically rigorous, it would be sufficiently compelling in practice. The proposed sampling method would be natural in a sense that it is based only on two minimalistic assumptions: the idea of ideal derivatives mentioned, as well as the Lipschitz condition of drift coefficient, which is a very common assumption in the theory of SDEs. For this reason, we may also expect that the idea will be universally applicable in various scenarios.

Future study could include developing higher order schemes: 3rd, 4th, ···, etc. Fortunately, it is easily confirmed that the higher order derivatives $L^nS(x, t), (n \in \mathbb{N})$ always have the form of $L^nS(x, t) = \xi(t)x + \eta(t)S(x, t)$ (see Eq. (77) in Appendix), and we can derive higher order algorithms with little modifications using them. It is subject for future study whether the ideal derivative substitution is still valid in practice for higher order schemes. Also, further tuning of some functions such as $\lambda(t)$ may accelerate the process further. This is also a subject for future research.

References

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

Austin, J., Johnson, D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34, 2021.

Burrage, K. and Burrage, P. General order conditions for stochastic Runge-Kutta methods for both commuting and non-commuting stochastic ordinary differential equation systems. *Applied Numerical Mathematics*, 28(2-4):161–177, 1998.

Burrage, K., Burrage, P., Higham, D. J., Kloeden, P. E., and Platen, E. Comment on “numerical methods for stochastic differential equations”. *Physical Review E*, 74(6):068701, 2006.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. WaveGrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations (ICLR)*, 2020.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., Dehak, N., and Chan, W. WaveGrad 2: Iterative refinement for text-to-speech synthesis. In *Proceedings of INTERSPEECH*, pp. 3765–3769. ISCA, 2021.

Clark, J. M. C. and Cameron, R. J. The maximum rate of convergence of discrete approximations for stochastic differential equations. In *Stochastic differential systems filtering and control*, pp. 162–171. Springer, 1980.

Dhariwal, P. and Nichol, A. Q. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Dockhorn, T., Vahdat, A., and Kreis, K. Score-based generative modeling with critically-damped Langevin diffusion. *arXiv preprint arXiv:2112.07068*, 2021.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two-time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*, 2021.

Hoogeboom, E., Nielsen, D., Jaini, P., Forrè, P., and Welling, M. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *Advances in Neural Information Processing Systems*, 2021.
Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

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

Ito, K. and Johnson, L. The LJ speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

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

Kloeden, P. E. and Platen, E. 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.

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

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

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations (ICLR)*, 2021.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pp. 3730–3738, 2015.

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

Milstein, G. N. and Tretyakov, M. V. *Stochastic numerics for mathematical physics*. Springer, 2004. doi: 10.1007/978-3-662-10063-9.

Mittal, G., Engel, J., Hawthorne, C., and Simon, I. Symbolic music generation with diffusion models. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Newton, N. J. Asymptotically efficient Runge-Kutta methods for a class of Ito and Stratonovich equations. *SIAM Journal on Applied Mathematics*, 51(2):542–567, 1991.

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

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

Pardoux, E. and Talay, D. Discretization and simulation of stochastic differential equations. *Acta Applicandae Mathematica*, 3(1):23–47, 1985.

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

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

Prenger, R., Valle, R., and Catanzaro, B. WaveGlow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

Rößler, A. Runge-Kutta methods for Itô stochastic differential equations with scalar noise. *BIT Numerical Mathematics*, 46(1):97–110, 2006.

Rößler, A. Strong and weak approximation methods for stochastic differential equations—some recent developments. In *Recent developments in applied probability and statistics*, pp. 127–153. Springer, 2010.
Rüemelin, W. Numerical treatment of stochastic differential equations. *SIAM Journal on Numerical Analysis*, 19(3):604–613, 1982.

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

San-Roman, R., Nachmani, E., and Wolf, L. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021.

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

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

Seitzer, M. pytorch-fid: FID Score for PyTorch. [https://github.com/mseitzer/pytorch-fid](https://github.com/mseitzer/pytorch-fid), August 2020. Version 0.1.1.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, pp. 2256–2265. PMLR, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2020a.

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

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

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2020b. [https://github.com/yang-song/score_sde_pytorch](https://github.com/yang-song/score_sde_pytorch)

Tocino, A. and Vigo-Aguiar, J. Weak second order conditions for stochastic Runge–Kutta methods. *SIAM Journal on Scientific Computing*, 24(2):507–523, 2002.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 2021.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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

Vovk, I. WaveGrad, 2020. [https://github.com/ivanvovk/WaveGrad](https://github.com/ivanvovk/WaveGrad)

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

Yamamoto, R., Song, E., and Kim, J.-M. Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020.
Appendix

A 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 2** (Strong Convergence). Let \( \tilde{x}_t \) be the path at the continuous limit \( h \to 0 \), and \( x_t \) 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 \( \gamma \) if the following inequality holds for a certain constant \( K_\gamma > 0 \),

\[
E[|x_t - \tilde{x}_t|] \leq K_\gamma h^\gamma.
\]

**Definition 3** (Weak Convergence). Similarly, it is said that the scheme has the weak order of convergence \( \beta \), if the following inequality holds for any test functions \( \phi(\cdot) \) in a certain class of functions, and a certain constant \( K_\beta > 0 \),

\[
|E[\phi(x_t)] - E[\phi(\tilde{x}_t)]| \leq K_\beta h^\beta.
\]

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 DDPMs that the diffusion coefficient \( g(t) \) is not dependent on \( x_t \), 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 \( y_T \) generated using a finite \( h > 0 \) approximates the continuous limit \( \tilde{y}_T \) \( (h \to 0) \) driven by the same Brownian motion. Instead, we are more interested in whether the density \( p(y_T) \) of the samples generated with a finite step size \( h > 0 \) approximates the ideal density \( p(\tilde{y}_T) \), \( (h \to 0) \) which is supposed to approximate the true density \( p(x_0) \). In this sense, the concept of strong convergence is not much important for us, but the weak convergence would be sufficient.

B Derivation of \( p(x_t | x_0, t) \)

Let us consider the Euler-Maruyama discretization of the forward process.

\[
x_{t+h} = x_t + f(x_t, t)h + \sqrt{h}g(t)w_t.
\]

As \( f(x_t, t) := h^{-1}(\sqrt{1 - g(t)^2}h - 1)x_t \), it is written as follows.

\[
x_{t+h} = \sqrt{1 - g(t)^2}hx_t + \sqrt{h}g(t)w_t.
\]

This is essentially the same as the formulation in (Ho et al., 2020), if letting \( \beta_t = g(t)^2h \). Let us use this notation. By using Eq. (37) twice, we have

\[
x_{2h} = \sqrt{1 - \beta_h}x_h + \sqrt{\beta_h}w_h
\]

\[
= \sqrt{1 - \beta_h} \left( \sqrt{1 - \beta_0}x_0 + \sqrt{\beta_0}w_0 \right) + \sqrt{\beta_h}w_h
\]

\[
= \sqrt{1 - \beta_h} \sqrt{1 - \beta_0}x_0 + \sqrt{1 - \beta_h} \sqrt{\beta_0}w_0 + \sqrt{\beta_h}w_h
\]

\[
= \sqrt{1 - \beta_h} \sqrt{1 - \beta_0}x_0 + \sqrt{1 - (1 - \beta_h)(1 - \beta_0)}w_h'
\]

where \( w'_h \sim \mathcal{N}(0, 1) \), which is other than (but is correlated to) \( w_h \). Here we used the law that the sum of independent Gaussian variables is also Gaussian,

\[
X \sim \mathcal{N}(0, a^2), Y \sim \mathcal{N}(0, b^2) \implies X + Y \sim \mathcal{N}(0, a^2 + b^2).
\]

We can similarly show the following by using Eq. (37) three times.

\[
x_{3h} = \sqrt{1 - \beta_{2h}} \sqrt{1 - \beta_h} \sqrt{1 - \beta_0}x_0 + \sqrt{1 - (1 - \beta_{2h})(1 - \beta_h)(1 - \beta_0)}w_{2h}.
\]
Iterating such operations recursively, we have

\[ x_{nh} = \sqrt{\prod_{k=0}^{n} (1 - \beta_{kh})} x_0 + \sqrt{1 - \prod_{k=0}^{n} (1 - \beta_{kh})} w'_{nh}. \]  (41)

Let \( \alpha_{nh} = \prod_{k=0}^{n} (1 - \beta_{kh}) \), then it is written as

\[ x_{nh} = \sqrt{\alpha_{nh}} x_0 + \sqrt{1 - \alpha_{nh}} w'_{nh}, \]  (42)

which means that the density \( p(x_{nh} \mid x_0, nh) \) is equal to the following Gaussian,

\[ p(x_{nh} \mid x_0, nh) = \mathcal{N}(\sqrt{\alpha_{nh}} x_0, (1 - \alpha_{nh}) 1). \]  (43)

Let \( t = nh \), then we have

\[ p(x_t \mid x_0, t) = \mathcal{N}(\sqrt{\alpha_t} x_0, (1 - \alpha_t) 1). \]  (44)

### C Derivation of Eq. (8) by the Volterra Product Integral

Similarly to the Riemann integral which is intuitively defined as “the summation of thin slices”:

\[ \int_a^b f(x)dx := \lim_{h \to 0} a/h \sum_{k=0}^{a/h} f(kh)h, \]  (45)

we may also define a product integral, under some conditions. The following “product of thin slices” is called the Volterra product integral.

\[ \prod_{0}^{a} (1 + f(x)dx) := \lim_{h \to 0} a/h \prod_{k=0}^{a/h} (1 + f(kh)h) \]

\[ = \lim_{h \to 0} \exp a/h \sum_{k=0}^{a/h} \log(1 + f(kh)h) \]

\[ = \exp \lim_{h \to 0} a/h \sum_{k=0}^{a/h} (f(kh)h + o(h)) \]

\[ = \exp \int_{0}^{a} f(x)dx. \]  (46)

Using the Volterra product integral, \( \alpha(t) \) in Eq. (8) is derived as follows,

\[ \alpha(t) := \lim_{h \to 0} t/h \prod_{k=0}^{t/h} (1 - g(kh)^2 h) \]

\[ = \prod_{0}^{t} (1 - g(\tau)^2 d\tau) \]

\[ = \exp \int_{0}^{t} -g(\tau)^2 d\tau. \]  (47)

By differentiating the both sides, we have

\[ \dot{\alpha}(t) = \frac{d}{dt} \left( \exp \int_{0}^{t} -g(\tau)^2 d\tau \right) \]

\[ = \left( \exp \int_{0}^{t} -g(\tau)^2 d\tau \right) \cdot \frac{d}{dt} \left( \int_{0}^{t} -g(\tau)^2 d\tau \right) \]

\[ = \alpha(t) \cdot (-g(t)^2). \]  (48)

Thus, we obtain the relation between \( g(t) \) and \( \alpha(t) \) as follows, and we can compute \( g(t) \) from \( \alpha(t) \).

\[ g(t)^2 = -\frac{\dot{\alpha}(t)}{\alpha(t)} = -\frac{d}{dt} \log \alpha(t). \]  (49)
D Itô-Taylor Expansion

Let us summarize the second order Itô-Taylor expansion. 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.

\[
x_{t+h} = x_t + \int_t^{t+h} f(x, t')dt' + \int_t^{t+h} g(t')dB_t
\]

\[
= x_t + \int_t^{t+h} \left( f(x, t) + \int_t^{t'} Lf(x, s)ds + \int_t^{t'} Gf(x, s)dB_s \right)dt' + \int_t^{t+h} \left( g(x, t) + \int_t^{t'} Lg(x, s)ds + \int_t^{t'} Gg(x, s)dB_s \right)dB_t
\]

\[
= x_t + \int_t^{t+h} dt' \int_t^{t'} ds Lf(x, s) + \int_t^{t+h} ds \int_t^{t'} dB_s Gf(x, s) + \int_t^{t+h} dB_t \int_t^{t'} ds \int_t^{t'} dB_s Gf(x, s) + \int_t^{t+h} dB_t \int_t^{t'} ds \int_t^{t'} dB_s Gg(x, s) + R
\]

where \( R \) is the remainder consisting of triple integrals, which is ignored now. If we also ignore the double integrals, we obtain the Euler-Maruyama scheme.

D.1 Evaluation of Each Integral in Eq. (51)

Let us evaluate the double integrals. We can put \( t = 0 \) for simplicity.

\[
\left\{ \begin{array}{l}
\int_0^h \int_0^{t'} ds dt' \quad \cdots \text{(deterministic)} \\
\int_0^h \int_0^{t'} dB_s dt' \quad \cdots \text{(stochastic 1)} \\
\int_0^h \int_0^{t'} ds dB_t' \quad \cdots \text{(stochastic 2)} \\
\int_0^h \int_0^{t'} dB_s dB_t' \quad \cdots \text{(stochastic 3)}
\end{array} \right.
\]

**Deterministic:** The deterministic one, \( \int_0^h \int_0^{t'} ds dt' \), is easy to evaluate.

\[
\int_0^h \int_0^{t'} ds dt' = \int_0^h t' dt' = \frac{1}{2} h^2.
\]

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

\[
\tilde{z} := \int_0^h \int_0^{t'} dB_s dt' = \int_0^h B_{t'} dt'.
\]
As \( \tilde{z} \) is the limit of a sum of Gaussian variables, i.e.,

\[
\tilde{z} = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} h \frac{B_{h_i/n}}{n}
\]

\[
= \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=1}^{i} (B_{h_j/n} - B_{h(j-1)/n})
\]

\[
= \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=1}^{i} W_j, \quad W_j \sim \mathcal{N}(0, \frac{h}{n})
\]

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 \( B_h \).

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

**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).

\[
\int_0^h \int_0^{t'} ds dB_s = \int_0^h t' dB_{t'}
\]

\[
= t' B_{t'}^h - \int_0^h B_{t'} dt'
\]

\[
= hB_h - \tilde{z}.
\]

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

\[
\int_0^h \int_0^{t'} dB_s dB_{t'} = \int_0^h B_{t'} dB_{t'}
\]

\[
= \frac{1}{2} (B_h^2 - h).
\]

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

\[
x_{t+h} = x_t + hf(x,t) + \tilde{w}g(x,t)
\]

\[
+ \frac{h^2}{2} Lf(x,t) + \tilde{z}Gf(x,t) + (\tilde{w}h - \tilde{z})Lg(x,t) + \tilde{w}^2 - \frac{h}{2} Gg(x,t).
\]

**D.2 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 \( \tilde{w} \). By Itô’s isometry, we have

\[
\mathbb{E}[(\tilde{w}h - \tilde{z})^2] = \mathbb{E} \left[ \left( \int_0^h s dB_s \right)^2 \right]
\]

\[
= \mathbb{E} \left[ \int_0^h s^2 ds \right] = \frac{1}{3} h^3.
\]

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,

\[
\mathbb{E}[\tilde{w} \tilde{z}] = \mathbb{E} \left[ B_h \int_0^h s dB_s \right].
\]
\[
\begin{align*}
E \left[ \left( \int_0^h dB_s \right) \left( \int_0^h s dB_s \right) \right] &= E \left[ \int_0^h s ds \right] \\
&= \frac{1}{2} h^2.
\end{align*}
\] (60)

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

\[
E[\tilde{z}^2] = E[(\tilde{w} h - \tilde{z})^2 - h^2 \tilde{w}^2 + 2h \tilde{w} \tilde{z}]
\]
\[
= \frac{1}{3} h^3 - h^2 \cdot h + 2h \cdot \frac{h^2}{2}
\]
\[
= \frac{1}{3} h^3.
\] (61)

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

\[
\begin{bmatrix}
\tilde{w} \\
\tilde{z}
\end{bmatrix}
= \begin{bmatrix}
\sqrt{h} \\
h \sqrt{h}/2
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix}
\] (62)

where \( u_1, u_2 \text{i.i.d.} \sim N(0, 1) \).

Let us compute the covariance matrix just to be sure.

\[
E \left[ \begin{bmatrix}
\tilde{w} \\
\tilde{z}
\end{bmatrix} \begin{bmatrix}
\tilde{w} & \tilde{z}
\end{bmatrix} \right]
\]
\[
= E \left[ \begin{bmatrix}
\sqrt{h} \\
h \sqrt{h}/2
\end{bmatrix} \begin{bmatrix}
u_1 \\
u_2
\end{bmatrix} \begin{bmatrix}
\sqrt{h} & h \sqrt{h}/2
\end{bmatrix} \begin{bmatrix}
u_1 \\
u_2
\end{bmatrix} \right]
\]
\[
= \begin{bmatrix}
h \\
h^2/2
\end{bmatrix}
\begin{bmatrix}
h^2/2 \\
h^3/3
\end{bmatrix}.
\] (63)

\section*{E Experimental Setup of WaveGRIT}

\subsection*{E.1 Network Structure}

In the waveform synthesis experiment, we used the WaveGrad vocoder as the potential gradient (score function) predictor, i.e.,

\[
S(x_t, t) := \text{WaveGrad}(x_t, \alpha(t); c),
\] (64)

where \( c \) is the conditioning information, specifically the mel-spectrogram of the target signal. Figure S shows the network structure of the WaveGrad vocoder For more details of each block, see the original literature [Chen et al., 2020] and an open implementation [Vovk, 2020]. In WaveGrad, instead of encoding the time parameter \( t \), the antinoise level \( \alpha(t) \) is fed into the network. In the open implementation by [Vovk, 2020], \( \sqrt{\alpha(t)} \) is used.
E.2 On Vovk’s WaveGrad Sampler

Vovk’s sampler is basically based on the Euler-Maruyama sampler. The sampling schedules provided in the repository (Vovk, 2020) are as follows.

$N(\#\text{steps})=25$: When the number of steps is 25, the Fibonacci noising schedule is used. The noise parameter $\beta_i$ (which is $g(t)^2 h$ in our notation) is as follows,

$$\beta_i = 10^{-6} \times \{1, 2, 3, 5, 8, 13, 21, \cdots \},$$

(65)

$N(\#\text{steps})=6$: As for the case where the number of steps is 6, a manually tuned schedule is used. The schedule is as follows.

$$\beta_i = \{10^{-6}, 1.9 \times 10^{-5}, 1.2 \times 10^{-4}, 10^{-3}, 1.1 \times 10^{-2}, 6 \times 10^{-2}\}.$$  

(66)

Using these $\beta_i$, the antinoise level $\alpha_i$ is defined using Eq. (7). Figure 6 compares Vovk’s schedules and our schedules.

Figure 6: Comparison of the noising schedules.
E.3 Subjective Evaluation (MOS)

In order to assess the sound quality generated by the WaveGRIT vocoder, seven colleagues working at the authors’ company participated in the subjective evaluation, and valid responses were obtained from six of them; the remaining one was excluded because he did not completed his assignments. Nevertheless, we have confirmed that even if his answers were taken into account, it would have little effect on the final results.

Each experimental participant was presented with 45 pieces of audio data. These data consisted of five samples for each of the nine different conditions (each row in Table 2 and the ground truth). The participants were asked to rate each of these 45 pieces of audio data on a scale of 1–5, with 1 being bad and 5 being good. The participants were requested to use headphones or earphones during the evaluation.

The experiment was conducted with a web-based simple questionnaire system. The file names of the audio data were anonymized by a hash function, so participants cannot easily tell which sample corresponds to which method.

We obtained 45 ratings from these six participants, for a total of 270 ratings, and calculated the mean and standard deviation of them. The results are shown in the MOS column of Table 2.

E.4 Objective Evaluation (SD, GKL and RGKL)

Let us show the definitions of the objective evaluation metrics we used in the sound quality assessment experiment. Let $X = (X_{tω})$ be the ground truth power spectrogram (i.e., squared magnitude STFT), and $Y = (Y_{tω})$ be the predicted power spectrogram. The STFT parameters were as follows: the window size and FFT size was $1024 / 22050$ [s], the frame hop size was $128 / 22050$ [s], and the hanning window was used.

Then the evaluation metrics we used in the evaluation were as follows.

$$
\begin{align*}
\text{SD}(X, Y) &:= \mathbb{E}_{tω} \left[ \left( \sqrt{X_{tω}} - \sqrt{Y_{tω}} \right)^2 \right]^{1/2} \\
\text{GKL}(X, Y) &:= \mathbb{E}_{tω} \left[ X_{tω} \log \frac{X_{tω}}{Y_{tω}} - X_{tω} + Y_{tω} \right] \\
\text{RGKL}(X, Y) &:= \text{GKL}(Y, X)
\end{align*}
$$

(67)

All of these metrics are non-negative, and the minimums ($= 0$) are achieved if and only if $X = Y$. 

20
F Derivation of the Derivatives

F.1 Spatial Derivative $\partial_x S(x, t)$

Let us first compute the spatial derivative $\partial_x S(x, t)$. It is easily computed as follows.

$$\partial_x S(x, t) := \frac{\partial}{\partial x} \left( x - \sqrt{\alpha(t)} x_0 \right)$$

$$= \frac{1}{\sqrt{1 - \alpha(t)}}.$$  \hfill (68)

F.2 Time Derivative $\partial_t S(x, t)$

Next, let us compute $\partial_t S(x, t)$. During the computation, $x_0$ is replaced by the relation

$$x_0 = \frac{1}{\sqrt{\alpha(t)}} \left( x - \sqrt{1 - \alpha(t)} S(x, t) \right). \hfill (69)$$

We also use the following relations between $\alpha(t), g(t)$.

$$\dot{\alpha}(t) = -g(t)^2 \alpha(t). \hfill (70)$$

Using the above information, we may compute $\partial_t S(x, t)$ as follows.

$$\partial_t S(x, t) := \frac{\partial}{\partial t} \left( x - \sqrt{\alpha(t)} x_0 \right)$$

$$= \frac{1}{\sqrt{1 - \alpha(t)}} \left( -\frac{1}{2} \dot{\alpha}(t) \alpha(t)^{-1/2} x_0 \right) + \left( x - \sqrt{\alpha(t)} x_0 \right) \left( \frac{1}{2} \dot{\alpha}(t) (1 - \alpha(t))^{-3/2} \right)$$

$$= \frac{\dot{\alpha}(t)}{2(1 - \alpha(t))^{3/2}} \left( -\frac{1 - \alpha(t)}{\sqrt{\alpha(t)}} x_0 + \left( x - \sqrt{\alpha(t)} x_0 \right) \right)$$

$$= \frac{\dot{\alpha}(t)}{2(1 - \alpha(t))^{3/2}} \left( \frac{1 - \alpha(t)}{\sqrt{\alpha(t)}} \left( x - \sqrt{1 - \alpha(t)} S(x, t) \right) \right)$$

$$= \frac{\dot{\alpha}(t)}{2(1 - \alpha(t))^{3/2}} \left( 1 - \frac{1}{\alpha(t)} \right) x + \frac{\sqrt{1 - \alpha(t)}}{\alpha(t)} S(x, t)$$

$$= \frac{\dot{\alpha}(t)}{2(1 - \alpha(t))^{3/2}} \left( -1 - \alpha(t) \right) x + \frac{\sqrt{1 - \alpha(t)}}{\alpha(t)} S(x, t)$$

$$= \frac{1}{2} \dot{\alpha}(t) \frac{\sqrt{1 - \alpha(t)}}{\alpha(t)} S(x, t)$$

$$= \frac{g(t)^2}{2 \sqrt{1 - \alpha(t)}} \left( -x + \frac{S(x, t)}{\sqrt{1 - \alpha(t)}} \right)$$

$$= \frac{g(t)^2}{2 \sqrt{1 - \alpha(t)}} \left( -x - \frac{S(x, t)}{\sqrt{1 - \alpha(t)}} \right) \hfill (71)$$

F.3 The Derivatives $La, Ga, Lb, Gb$

The backward dynamics we are considering is as follows. For simplicity, we denote $T - \tau$ by $t$.

$$d\gamma = \left( \frac{g(t)^2}{2} \gamma - g(t)^2 \frac{1}{\sqrt{1 - \alpha(t)}} S(\gamma, t) \right) d\tau - g(t)d\tilde{B}_\tau$$

$$= \left( \frac{g(t)^2}{2} \gamma - g(t)^2 \frac{1}{\sqrt{1 - \alpha(t)}} S(\gamma, t) \right) d\tau - g(t)d\tilde{B}_\tau \hfill (72)$$

21
Thus the drift $a$, diffusion $b$, and the operators $L, G$ are written as follows.

$$a = -\bar{f} = \frac{g(t)^2}{2} y_\tau - \frac{g(t)^2}{\sqrt{1 - \alpha(t)}} S(y_\tau, t) \quad (73)$$

$$b = g(t) \quad (74)$$

$$L = -\partial_t + \left( \frac{g(t)^2}{2} y_\tau - \frac{g(t)^2}{\sqrt{1 - \alpha(t)}} S(y_\tau, t) \right) \partial_\xi + \frac{g(t)^2}{2} \partial_y^2 \quad (75)$$

$$G = -g(t) \partial_y \quad (76)$$

Firstly, let us compute $La = L(\bar{f})$. But before that, we compute $L(\xi(t)y_\tau + \eta(t)S(y_\tau, t))$ for general $\xi(t), \eta(t)$ which are sufficiently smooth. For simplicity, let us drop the parameters of variables. Then $L(\xi y + \eta S)$ is computed as follows.

$$L(\xi y + \eta S) = -\partial_t + \left( \frac{g^2 y}{2} - \frac{g^2}{\sqrt{1 - \alpha}} S \right) \partial_\xi + \frac{g^2}{2} \partial_y^2 \quad (77)$$

From the above formula we realize that the class of functions of the form

$$\{\xi_n(t)y + \eta_n(t)S(y, t)\} \quad (78)$$

is closed under the operation of $L$. That is, all of the higher order derivatives $L^2 a, L^3 a, \cdots$ are written in the form of $L^2 a = \xi_n(t)y + \eta_n(t)S(y, t)$, though the coefficients $\xi_n(t), \eta_n(t)$ will become increasingly complicated.

Using the above formula, we can compute $La = L(\bar{f})$ as follows.

$$La = L(\bar{f}(y, t)) = L \left( \frac{g^2 y}{2} - \frac{g^2}{\sqrt{1 - \alpha}} S(y, t) \right)$$

$$= L(\xi y + \eta S) \bigg|_{\xi=g^2/2, \eta=-g^2/\sqrt{1-\alpha}}$$

$$= \left( -\partial_t + \frac{g^2}{2} \right) y + \left( \partial_t \left( \frac{g^2}{\sqrt{1 - \alpha}} \right) - \frac{g^4}{2\sqrt{1 - \alpha}} + \frac{g^4}{2(1 - \alpha)^{3/2}} \right) S(y, t)$$

$$= \left( -g\bar{g} + \frac{g^4}{4} \right) y + \left( \frac{2g\bar{g}}{\sqrt{1 - \alpha}} - \frac{-\alpha g^2}{2(1 - \alpha)^{3/2}} - \frac{g^4}{2\sqrt{1 - \alpha}} + \frac{g^4}{2(1 - \alpha)^{3/2}} \right) S(y, t)$$

$$= \left( -g\bar{g} + \frac{g^4}{4} \right) y + \left( \frac{2g\bar{g}}{\sqrt{1 - \alpha}} - \frac{-\alpha g^2}{2(1 - \alpha)^{3/2}} - \frac{g^4}{2\sqrt{1 - \alpha}} + \frac{g^4}{2(1 - \alpha)^{3/2}} \right) S(y, t)$$

$$= \left( -g\bar{g} + \frac{g^4}{4} \right) y + \frac{2g\bar{g}}{\sqrt{1 - \alpha}} \frac{g^4}{2\sqrt{1 - \alpha}} \left( -\frac{\alpha}{1 - \alpha} - 1 + \frac{1}{1 - \alpha} \right) S(y, t)$$

$$= \left( -g\bar{g} + \frac{g^4}{4} \right) y + \frac{2g\bar{g}}{\sqrt{1 - \alpha}} S(y, t) \quad (79)$$

$Ga, Lb$ and $Gb$ are similarly computed.

$$Ga = G(\bar{f}) = -g\partial_y \left( \frac{g^2 y}{2} - \frac{g^2}{\sqrt{1 - \alpha}} S(y, t) \right)$$
\[-g \left( \frac{g^2}{2} - \frac{g^2}{\sqrt{1 - \alpha}} \frac{1}{\sqrt{1 - \alpha}} \right) \]
\[-g \left( \frac{g^2}{2} - \frac{g^2}{(1 - \alpha)} \right) \]
\[-g^3 \left( \frac{1}{2} - \frac{1}{1 - \alpha} \right) \]
\[= \frac{1 + \alpha}{2(1 - \alpha)} g^3 \] (80)

\[Lb = L(-g) = \partial_t (-g) = \partial_t g = \frac{\dot{g}}{g} \] (81)

\[Gb = 0. \] (82)

F.4 Other Composite Quantities and the Derivatives of $\lambda(t)$

Let us compute each quantity of Eq. (30).

\[g(t)\dot{g}(t) = \frac{1}{2} \partial_t g(t)^2 \]
\[= \frac{1}{2} \partial_t \left( \dot{\lambda} \tanh \frac{\lambda}{2} \right) \]
\[= \frac{1}{2} \left( \dot{\lambda} \tanh \frac{\lambda}{2} + \frac{\dot{\lambda}^2}{2 \cosh^2(\lambda/2)} \right). \] (83)

\[g(t)\dot{g}(t) \frac{1}{\sqrt{1 - \alpha(t)}} = \frac{\dot{g}}{\tanh(\lambda/2)} \]
\[= \frac{1}{2 \tanh(\lambda/2)} \left( \dot{\lambda} \tanh \frac{\lambda}{2} + \frac{\dot{\lambda}^2}{2 \cosh^2(\lambda/2)} \right) \]
\[= \frac{1}{2} \left( \dot{\lambda} + \frac{\dot{\lambda}^2}{2 \tanh(\lambda/2) \cosh(\lambda/2)} \right) \]
\[= \frac{1}{2} \left( \dot{\lambda} + \frac{\dot{\lambda}^2}{\sinh \lambda} \right). \] (84)

\[g(t)^3(1 + \alpha(t)) \frac{1}{2(1 - \alpha(t))} = G_a = \frac{g^3}{1 - \alpha} - \frac{g^3}{2} \]
\[= \frac{(\dot{\lambda} \tanh(\lambda/2))^{3/2}}{\tanh^4(\lambda/2)} - \frac{g^3}{2} \]
\[= \frac{\dot{\lambda}^{3/2}}{\tanh^{1/2}(\lambda/2)} - \frac{g^3}{2} \]
\[= \sqrt{\frac{\dot{\lambda}^3}{\tanh(\lambda/2)}} - \frac{g^3}{2}. \] (85)

Note that Eq. (85) does not diverge, because $\lambda(t)$ is designed so that $\dot{\lambda}(t)/\tanh(\lambda(t)/2) < \infty$ in Eq. (31).

Let us also compute the first and second derivatives of $\lambda(t)$ defined in Eq. (31).

\[\dot{\lambda}(t) = \frac{Ak}{A + e^{-kt}}, \quad \ddot{\lambda}(t) = \frac{Ak^2 e^{-kt}}{(A + e^{-kt})^2}. \] (86)
G  Examples of Noising Schedules

Figure 7 shows noising schedules based on the softplus scheduler \( \lambda(t) = \log(1 + Ae^{kt}) \). We can see that none of the functions diverge within the range of interest, especially when \( t \to 0 \). However, if we carelessly design \( \lambda(t) \), some of the functions diverge. (Figure 8)

Noising schedule based on the parameters \( \alpha(0) = 1 - 10^{-4}, \alpha(1) = 0.01 \Rightarrow \lambda(t) = \log(1 + 0.0202 \times \exp(9.89 \times t)) \)

Noising schedule based on the parameters \( \alpha(0) = 1 - 10^{-7}, \alpha(1) = 10^{-3} \Rightarrow \lambda(t) = \log(1 + 0.000853 \times \exp(15.7 \times t)) \)

Noising schedule based on the parameters \( \alpha(0) = 1 - 5 \times 10^{-4}, \alpha(1) = 0.915 \Rightarrow \lambda(t) = \log(1 + 0.0457 \times \exp(2.89 \times t)) \)

Figure 7: Comparison of noising schedules based on the softplus scheduler: \( \lambda(t) = \log(1 + Ae^{kt}) \).

Noising schedule based on \( \lambda(t) = 5t \)

Noising schedule based on \( \lambda(t) = 5t^2 \)

Figure 8: Unsuccessful noising schedules (Linear and Quadratic).
H Pseudocode

The following pseudocode shows the proposed sampler (Algorithm 2). The Euler-Maruyama sampler (Algorithm 1) is almost the same as the following pseudocode, but the equations to compute $\rho$, $\mu$ and $n$ should be modified.

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

Require:
- Trained neural network model $S(x_t, t, c)$
- Data size $d > 0 \cdots \text{Int}$
- Total time $T > 0 \cdots \text{Float}$
- Number of steps $N > 0 \cdots \text{Int}$
- Initial and terminal antinoise levels $\alpha_0, \alpha_T \in (0, 1) \cdots \text{Float}$
- (Optional) Conditioning information $c$

Constants:
- $h \leftarrow T/N$
- $A \leftarrow 2(1 - \alpha_0) + 2\sqrt{1 - \alpha_0}$
- $A_n \leftarrow 1 - (\text{tanh}(\lambda/2))^2$, $g \leftarrow \sqrt{\lambda \text{tanh}(\lambda/2)}$
- $\alpha \leftarrow 1 - tanh(\lambda/2)$
- $\theta \leftarrow \sqrt{\frac{\lambda^3}{\text{tanh}(\lambda/2)}} - g^2/2$
- $\nu \leftarrow h^{3/2} \left( (w - z) g + \theta z \right) - g\sqrt{T}w$

Begin

$y \sim \mathcal{N}(0_d, I_d)$ \hfill \text{♯ Draw a $d$-dimensional Gaussian noise with unit variance}

for $n = 0$ to $N - 1$

[Compute the Noise Level]
- $t \leftarrow T - nh$
- $\lambda \leftarrow \log(1 + Ae^{kt})$, $\dot{\lambda} \leftarrow \frac{Ak}{A + e^{-kt}}$, $\ddot{\lambda} \leftarrow \frac{Ak^2 e^{-kt}}{(A + e^{-kt})^2}$
- $\alpha \leftarrow 1 - (\text{tanh}(\lambda/2))^2$, $g \leftarrow \sqrt{\lambda \text{tanh}(\lambda/2)}$

[Compute Coefficients]
- $\Gamma \leftarrow \frac{1}{2} \left( \lambda \text{tanh} \left( \frac{\lambda}{2} + \frac{\dot{\lambda}^2}{2(\cosh(\lambda/2))^2} \right) \right)$,
- $\Lambda \leftarrow \frac{1}{2} \left( \dot{\lambda} + \frac{\ddot{\lambda}^2}{\sinh \lambda} \right)$
- $\rho \leftarrow 1 + \frac{g^2}{2} h + \frac{h^2}{2} \left( \frac{g^2}{4} - \Gamma \right)$, $\mu \leftarrow -\dot{\lambda} h + \Lambda h^2$

[Draw a Correlated Driving Noise]
- if $n = N - 1$ then

  $n \leftarrow 0_d$ \hfill \text{♯ No noise is injected at the final step}

else

  $w \sim \mathcal{N}(0_d, I_d)$, $u \sim \mathcal{N}(0_d, I_d)$, $z = \frac{1}{2} w + \frac{1}{2\sqrt{3}} u$ \hfill \text{See Eq. (62) and Theorem 1}

  $\Theta \leftarrow \sqrt{\frac{\lambda^3}{\text{tanh}(\lambda/2)}} - g^2/2$ \hfill \text{Eq. (30)}

  $n \leftarrow h^{3/2} \left( (w - z) g + \Theta z \right) - g\sqrt{T}w$ \hfill \text{Eq. (29)}

end if

[Update Data]
- $y \leftarrow \rho y + \mu S(y, t, c) + n$ \hfill \text{Eq. (28)}

(Optional) Clip outliers of $y$ so that e.g. $-1 \leq y \leq 1$.

end for

End

Output: $y$
The following pseudocode shows an example for the DDPM training.

**Algorithm 3 DDPM training**

**Require:**
- Data size \( d > 0 \) \( \cdots \) \( \text{Int} \)
- Training Data \( D = \{ x_0^{(i)}, c^{(i)} \} \), where \( x_0^{(i)} \in \mathbb{R}^d \) \( \ddagger \) The conditioning information \( c \) is optional.
- Neural network model \( S(x_t, \alpha_t, c) \), parameterized by \( \theta \).
- Neural network optimizer (Adam) and its parameters (e.g. learning rate)
- Total time \( T > 0 \) \( \cdots \) \( \text{Float} \) \( \ddagger \) Not necessarily the same as the one for the synthesis.
- Initial and terminal antinoise levels \( \alpha_0, \alpha_T \in (0, 1) \) \( \cdots \) \( \text{Float} \) \( \ddagger \) Not necessarily the same as the one for the synthesis.
- Batch size \( b > 0 \) \( \cdots \) \( \text{Int} \)

**Constants:**
- \( A \leftarrow 2(1 - \alpha_0) + 2\sqrt{1 - \alpha_0} \)
- \( k \leftarrow \frac{1}{T} \left( \log \frac{2(1 - \alpha_T) + 2\sqrt{1 - \alpha_T}}{\alpha_T} - \log A \right) \) \( \ddagger \) Eq. (33)

**Begin**

for sufficiently many times until convergence do

[**Draw a Batch**]

\( \text{batch} \leftarrow [ ] \) \( \ddagger \) Empty list

for \( b \) times (batch size) do

\( (x_0, c) \sim D \) \( \ddagger \) Draw a data from the set of training data; the conditioning \( c \) is optional.

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

\( \lambda \leftarrow \log(1 + Ae^{kt}) \)

\( \alpha \leftarrow 1 - (\tanh(\lambda/2))^2 \)

\( w \sim \mathcal{N}(0_d, 1_d) \) \( \ddagger \) Draw a \( d \)-dimensional Gaussian noise with unit variance.

Append the tuple \( (x_0, (c), t, \alpha, w) \) to \( \text{batch} \)

end for

[**Forward Computation**]

\( ([x_0], [c], [t], [\alpha], [w]) \leftarrow \text{batch} \)

\( [s] \leftarrow S([\sqrt{\alpha}] \odot [x_0] + [\sqrt{1 - \alpha}] \odot [w], [t], [c]) \)

[**Back-propagation**]

\( \mathcal{L} \leftarrow \mathbb{E}_{\text{batch}} \| [w] - [s] \|_1 \) \( \ddagger \) Loss evaluation.

Compute \( \nabla_\theta \mathcal{L} \) by the back-propagation.

\( \theta \leftarrow \text{Adam}(\theta, \nabla_\theta \mathcal{L}) \) \( \ddagger \) Update the parameters of \( S(\cdot, \cdot, \cdot) \).

end for

**End**

**Output:** The network parameter \( \theta \).
I Additional Results

I.1 Unconditional Image Synthesis

I.1.1 Comparison of the Denoising Process

Figure 9: Comparison of the image synthesis process of proposed (above) and Euler-Maruyama (below) samplers. The dataset is CelebA (64 × 64). All the conditions but the sampling algorithm are the same for both cases. The number of refinement steps is $N = 15$. 
1.1.2 Random Samples

Below images show the examples of random sampling from a same DDPM checkpoint. All conditions are the same except for the sampling algorithm and the number of refinement steps $N$. The same hyperparameters of noising schedule $(g(t), \alpha(t))$, the same initial Gaussian noise, and the same random seed are used.

(a) Itô-Taylor sampler with ideal derivatives (proposed)

(b) Euler-Maruyama sampler

Figure 10: The case $N = 5$ ($h = 0.2$).
(a) Itô-Taylor sampler with ideal derivatives (proposed)

(b) Euler-Maruyama sampler

Figure 11: The case $N = 10$ ($h = 0.1$).
(a) Itô-Taylor sampler with ideal derivatives (proposed)

(b) Euler-Maruyama sampler

Figure 12: The case $N = 25$ ($h = 0.04$).
Figure 13: The case $N = 50$ ($h = 0.02$).
(a) Itô-Taylor sampler with ideal derivatives (proposed)

(b) Euler-Maruyama sampler

Figure 14: The case $N = 75 \ (h = 0.0133)$. 
Figure 15: The case $N = 100 \ (h = 0.01)$. 
I.2 Conditional Speech Synthesis

Some audio samples will be uploaded to the author’s website.

- [https://tachi-hi.github.io/research/](https://tachi-hi.github.io/research/)

I.2.1 Backward denoising process \( (N = 25 \text{ steps}) \)

![Backward denoising process](image)

Figure 16: Speech synthesis result (No. 1). Synthesized waveforms and the corresponding mel-spectrograms.

![Backward denoising process](image)

Figure 17: Speech synthesis result (No. 2).
1.2.2 Backward denoising process ($N = 6$ steps)

Figure 18: Speech synthesis result (No. 1).

Figure 19: Speech synthesis result (No. 2).
1.2.3 Backward denoising process ($N = 4$ steps)

Figure 20: Speech synthesis result (No. 1).

Figure 21: Speech synthesis result (No. 2).

Figure 22: Speech synthesis result (No. 3).
J Author Contributions

This study was originally a project in the speech team (HT, MI, YK and YW) to develop a better neural vocoder. Initially, we considered a method like “Runge-Kutta WaveGrad”, but soon found it difficult. It was HT’s idea that we can adopt the Itô-Taylor scheme if we substitute the derivatives of the drift term with “ideal derivatives”. Based on this idea, HT derived the algorithm. The vocoder experiments were conducted by the speech team. The experimental code for image synthesis was implemented by MG, and the image experiments were conducted by MG and HT. MG also corrected some mathematical errors in earlier versions of the proposed method. The manuscript was written by HT. The corresponding author is HT.