Chapter 4: Continuous-Time Formulations of Diffusion Models

In Chapters 2-3 we developed diffusion models in *discrete time*: a forward corruption (encoder) chain and a learned reverse-time denoising (decoder) chain. This viewpoint is implementation-friendly and leads directly to DDPM training and DDIM-style fast sampling. In this chapter we shift to a complementary viewpoint: *continuous-time diffusion*. Instead of a finite sequence of noise levels, we describe diffusion as the evolution of a time-indexed random process and its marginal distributions.

Why continuous time? The continuous-time formulation is not merely a limit of discrete-time models; it provides a unifying language that clarifies what is essential in diffusion modeling (the design of the forward corruption, the structure of the reverse-time generation, and the role of score estimation), and it cleanly separates *modeling choices* from *numerical discretization*. As a result, many discrete-time algorithms can be interpreted as different discretizations of the same underlying continuous-time dynamics, which helps explain why certain samplers work and how to improve them.

We organize the chapter around five goals:

1. **Forward/backward processes in continuous-time.** We formulate the forward noising process and the corresponding reverse-time backward generative process in a continuous-time framework.
2. **Connection to discrete-time diffusion.** We relate continuous-time descriptions back to the discrete encoder/decoder chains used in DDPM and DDIM, making precise which quantities match exactly and which are discretization artifacts.
3. **Score matching for training.** We introduce training objectives for learning the time-dependent score function required by the reverse-time dynamics, and connect them to the familiar discrete-time noise-prediction loss.
4. **Sampling algorithms.** We discuss how generation in continuous-time leads to a family of samplers through numerical solvers, clarifying tradeoffs between stochasticity, step size, and computational cost.
5. **Statistical and sampling theory.** We summarize what can be formally guaranteed (and what cannot) about estimation, approximation, and sampling error, and how these error sources interact with discretization and finite training data.

A more technical tutorial on continuous-time formulations of diffusion models can be found at [6].

1 Brownian Increments and the Notation $\mathrm{d}W_t$

To pass from a discrete-time diffusion chain to a continuous-time formulation, we need a mathematical object that captures *infinitesimal Gaussian perturbations*. The canonical choice is *Brownian motion* (also called a *Wiener process*), denoted by $\{W_t\}_{t\geq 0}$. In diffusion models, the symbol $\mathrm{d}W_t$ is a compact way to represent the random noise injected over an infinitesimal time interval.

Brownian motion via increments. A standard d-dimensional Brownian motion $W_t \in \mathbb{R}^d$ is characterized by: (i) $W_0 = 0$; (ii) *independent increments*: for $0 \le s < t$, $W_t - W_s$ is independent of $\{W_u : u \le s\}$; (iii) *Gaussian increments with linear variance growth*:

$$W_t - W_s \sim \mathsf{N}\big(0, (t-s)I\big) \quad \text{for} \quad t > s;$$

and (iv) $t \mapsto W_t$ is almost surely continuous.

A key consequence of (iii) is the short-time scaling: for a small $\Delta t > 0$,

$$\Delta W_t = W_{t+\Delta t} - W_t \sim \mathsf{N}(0, \Delta t \cdot I) \quad \Longleftrightarrow \quad \Delta W_t \overset{d}{=} \sqrt{\Delta t}\, Z, \qquad Z \sim \mathsf{N}(0, I).$$

Thus, Brownian increments have a standard deviation scale with $\sqrt{\Delta t}$, rather than Δt.

Why $\mathrm{d}W_t$ is not an ordinary derivative. Although W_t has continuous sample paths, it is almost surely *nowhere differentiable*. In particular, the limit $\lim_{\Delta t \to 0}(W_{t+\Delta t} - W_t)/\Delta t$ does not exist almost surely. Therefore, one should *not* interpret $\mathrm{d}W_t$ as a classical differential.

Instead, in stochastic calculus the notation

$$\mathrm{d}W_t$$

is read as *the infinitesimal increment of Brownian motion over* $[t, t + \mathrm{d}t]$. Heuristically, we have

$$\mathrm{d}W_t \approx W_{t+\mathrm{d}t} - W_t, \qquad \mathrm{d}W_t \sim \mathsf{N}(0, \mathrm{d}t \cdot I),$$

similar to the consequence of the Gaussian increments of Brownian motion.

From differentials to stochastic integrals. Writing a Stochastic Differential Equation (SDE)

$$\mathrm{d}X_t = b(X_t, t)\,\mathrm{d}t + \sigma(X_t, t)\,\mathrm{d}W_t$$

is shorthand for the integral equation

$$X_t = X_0 + \int_0^t b(X_s, s)\,\mathrm{d}s + \int_0^t \sigma(X_s, s)\,\mathrm{d}W_s,$$

where the last term is an Itô stochastic integral, interpretable as a limit (in the L^2 sense) of sums

$$\int_0^t \sigma(X_s, s)\,\mathrm{d}W_s = \lim_{n \to \infty} \sum_{k=0}^{n-1} \sigma(X_{t_k}, t_k)\,\big(W_{t_{k+1}} - W_{t_k}\big),$$

where $\{t_k\}_{k=1}^n$ is a partition of interval $[0, t]$. As can be seen, we interpret an SDE as describing adding random fluctuations to some deterministic evolution. The term $b(X_t, t)$ is called the drift and $\sigma(X_t, t)$ is called the diffusion.

2 Continuous-Time Forward and Backward Processes

This section introduces the continuous-time object that underlies diffusion models: a *forward noising process* that transports the data distribution to a simple reference distribution, and a corresponding *reverse-time backward generative process* that transports reference noise back to the data distribution.

2.1 Forward Process

Let $X_0 \sim P_{\text{data}}$ denote the clean data and its distribution. A general continuous-time diffusion (forward corruption) is specified by a Stochastic Differential Equation (SDE):

$$\mathrm{d}X_t = f(X_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t, \qquad t \in [0, T], \tag{2.1}$$

where W_t is a Wiener process, $f(\cdot, t)$ is a drift term, $g(t) \geq 0$ is the diffusion coefficient, and T is a terminal time. Note that in previous chapters, we use an integer T to denote the length of the encoder and decoder chains. Here, we reload the notation to denote a continuous terminal time. We denote the marginal distribution (resp. density) of X_t as P_t (resp. p_t), and clearly $P_0 = P_{\text{data}}$.

With $X_0 \sim P_{\text{data}}$, a common design goal is to ensure the terminal distribution of (2.1) approximates (if not exactly equals) an easy-to-sample distribution, e.g.,

$$P_T \approx \mathsf{N}(0, I).$$

Different choices of (f, g) yield different families of forward corruption. Two canonical examples used throughout the diffusion literature are *variance-preserving* (VP) and *variance-exploding* (VE) forward processes.

Example 2.1 (Variance-Preserving (VP) diffusion)**.** The VP family is designed so that the signal variance does not blow up as noise is injected; in its simplest (scalar-isotropic) form,

$$\mathrm{d}X_t = -\frac{1}{2}\beta(t)X_t\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}W_t, \qquad t \in [0, T], \tag{2.2}$$

where $\beta(t) \geq 0$ is a time-dependent rate function. The drift term in the VP diffusion is linear in X_t and the diffusion term is independent of X_t. When $\beta(t)$ is independent of t, i.e., a constant, (2.2) is known the Ornstein-Uhlenbeck (O-U) process.

The process (2.2) admits a closed-form solution (Exercise: try to solve it by yourself). Based on the closed-form solution, we denote

$$\bar{\alpha}(t) = \exp\Big(-\int_0^t \beta(s)\mathrm{d}s\Big).$$

Then the forward process leads to a closed-form conditional Gaussian marginal,

$$X_t \mid X_0 \sim \mathsf{N}\Big(\sqrt{\bar{\alpha}(t)}X_0, \big(1 - \bar{\alpha}(t)\big)I\Big).$$

This is the familiar marginal distribution in DDPM encoders. As a sanity check, as t increases, $\bar{\alpha}(t)$ decays from 1 to (typically) near 0, gradually destroying the information in X_0.

Example 2.2 (Variance-Exploding (VE) diffusion)**.** The VE family injects noise without a shrinkage drift; the variance of X_t grows ("explodes") over time:

$$\mathrm{d}X_t = \sqrt{\beta(t)}\mathrm{d}W_t, \qquad t \in [0, T], \tag{2.3}$$

with $\beta(t) \geq 0$. Define the cumulative variance

$$\sigma^2(t) = \int_0^t \beta(s)\mathrm{d}s.$$

Then the conditional marginal is again Gaussian with a particularly simple form:

$$X_t \mid X_0 \sim \mathsf{N}\Big(X_0, \sigma^2(t) I\Big),$$

or equivalently

$$X_t = X_0 + \sigma(t)\epsilon, \qquad \epsilon \sim \mathsf{N}(0, I).$$

Unlike VP, the mean stays at X_0 while the noise level grows in VE. For sufficiently large $\sigma(T)$, the terminal distribution becomes dominated by noise and is well-approximated by a Gaussian distribution with a large variance.

Both VP and VE diffusions admit closed-form forward marginals $P_t(X_t \mid X_0)$, which is crucial for diffusion model training via score matching in later sections. They also lead to different reverse-time dynamics and different practical behaviors (e.g., how time is parameterized and how samplers allocate steps), but the overarching continuous-time framework (2.1) treats them uniformly.

2.2 Backward Process and Score Function

An interesting property of the forward process (2.1) is that it allows us to travel reverse in time, namely, start from the terminal distribution P_T and evolve towards $P_0 = P_{\text{data}}$. This is the so-called backward process, also given in an SDE,

$$\mathrm{d}\overline{X}_t = \Big(- f(\overline{X}_t, T-t) + g(T-t)^2 \nabla_x \log p_{T-t}(\overline{X}_t) \Big)\mathrm{d}t + g(T-t)\mathrm{d}\overline{W}_t, \qquad t \in [0, T]. \tag{2.4}$$

Here $\overline{X}_0 \sim P_T$ and $\overline{W}_t$ is another independent Wiener process. Under mild conditions, it is shown that $X_t \overset{d}{=} \overline{X}_{T-t}$ [1], maintaining the marginal distribution of the forward process (2.1). Note that $\overline{X}_T$ follows the data distribution P_{data}. This implies that if we can simulate the backward process (2.4), new samples are generated.

Score function. Unfortunately, simulating the backward process (2.4) is never simple. There are two unknown quantities in defining (2.4): 1) The initial distribution P_T is unknown, and 2) the additional drift $\nabla_x \log p_{T-t}$ is also unknown. Among the two quantities, $\nabla_x \log p_{T-t}$ is of fundamental importance, as we may approximate P_T by a standard Gaussian distribution as long as T is large.

The additional drift $\nabla_x \log p_{T-t}$ is known as the (Stein's) **score function**. The gradient is taken with respect to the noisy state. In later context, we will drop the subscript x on ∇ for the score function. Roughly speaking, the score function dictates the evolution of the noisy state, so that the data distribution can be recovered.

To simulate (2.4) for sample generation, we need to estimate the score function. In literature, it is known as the score matching [7], which we will introduce in the next section.

3 Score Matching

For the sake of simplicity, in the remainder of this chapter, we instantiate f and g in (2.1) to the VP family with $\beta(t) = 1$:

$$\mathrm{d}X_t = -\frac{1}{2}X_t \mathrm{d}t + \mathrm{d}W_t. \tag{3.1}$$

The corresponding backward process is

$$\mathrm{d}\overline{X}_t = \Big(\frac{1}{2}\overline{X}_t + \nabla \log p_{T-t}(\overline{X}_t)\Big)\mathrm{d}t + \mathrm{d}\overline{W}_t. \tag{3.2}$$

To estimate the score function $\nabla \log p_t$, we first parameterize it using a trainable concept class $\mathcal{F}$, such as neural networks. We aim to find a $\widehat{s} \in \mathcal{F}$ verifying

$$\widehat{s}(x, t) \approx \nabla \log p_t(x).$$

This becomes a pure estimation problem. We need to find a good loss function and then choose a proper concept class.

3.1 A Conceptual Loss

We train $\widehat{s}$ by minimizing its deviation to $\nabla \log p_t$. This gives rise to the following conceptual loss function,

$$\widehat{s} \in \operatorname*{argmin}_{s \in \mathcal{F}} \int_0^T \mathbb{E}_{X \sim P_t}\left[\|s(X, t) - \nabla \log p_t(X)\|_2^2\right] \mathrm{d}t. \tag{3.3}$$

Note that we have an additional integral on t, since the score function depends on the time t. We also recall that P_t is the marginal distribution defined by the forward process (3.1).

Suppose the concept class $\mathcal{F}$ is parameterized by trainable parameter θ. Then we can apply an optimization algorithm to minimize the loss (3.3). However, this is overly ideal. A closer inspection of (3.3) says that we only know P_t and $\nabla \log p_t$ implicitly (through the forward process). There is not analytical expressions for them. Therefore, it is computationally intractable to calculate the gradient of (3.3).

3.2 Score Matching Loss

The key obstacle in (3.3) is that both the sampling distribution P_t and the score $\nabla \log p_t$ are implicit: we do not have analytic access to p_t for general data distributions. The central idea of *score matching* is to replace the intractable target $\nabla \log p_t$ by a *computable surrogate* obtained from the forward process.

Step 1: An alternative objective via integration by parts. In the seminal work [2], one considers the objective $\mathbb{E}_{X \sim P_t}\left[\|s(X, t) - \nabla \log p_t(X)\|_2^2\right]$ but eliminates the unknown score term by integration by parts. Concretely, for any time t, expanding the square gives rise to

$$\begin{aligned}\mathbb{E}_{X \sim P_t}\left[\|s(X, t) - \nabla \log p_t(X)\|_2^2\right] = \mathbb{E}_{X \sim P_t}\left[\|s(X, t)\|_2^2\right] - 2\mathbb{E}_{X \sim P_t}\left[\langle s(X, t), \nabla \log p_t(X)\rangle\right] \\ + \mathbb{E}_{X \sim P_t}\left[\|\nabla \log p_t(X)\|_2^2\right].\end{aligned}$$

The last term does not depend on s and thus can be dropped. We examine the cross term involving the inner product. Using $\nabla \log p_t = (\nabla p_t)/p_t$, we have

$$\mathbb{E}_{X\sim P_t}\big[\langle s(X,t), \nabla \log p_t(X)\rangle\big] = \int \langle s(x,t), \nabla p_t(x)\rangle \mathrm{d}x.$$

Under the assumption that p_t vanishes at infinity (which is almost always true), integration by parts yields

$$\int \langle s(x,t), \nabla p_t(x)\rangle \mathrm{d}x = -\int (\nabla \cdot s(x,t))\, p_t(x)\mathrm{d}x = -\mathbb{E}_{X\sim P_t}\big[\nabla \cdot s(X,t)\big],$$

where $\nabla \cdot s(\cdot, t)$ denotes the divergence with respect to x. Therefore, minimizing (3.3) is (formally) equivalent to minimizing

$$\int_0^T \mathbb{E}_{X\sim P_t}\Big[\|s(X,t)\|_2^2 + 2\,\nabla \cdot s(X,t)\Big]\mathrm{d}t, \tag{3.4}$$

which no longer involves $\nabla \log p_t$ explicitly.

Practical note. While (3.4) removes the unknown score, it introduces the divergence $\nabla \cdot s(X,t)$, which is expensive to compute for high-dimensional neural networks (it involves a trace of the Jacobian). In diffusion models, the more common route is *denoising score matching* (DSM) [7], which exploits the Gaussian structure of the corruption mechanism and leads to the familiar ϵ-prediction loss.

Step 2: From divergence to denoising (one more integration by parts). Under our instantiated VP forward process (3.1), (3.4) can be further simplified without the need to evaluate divergence of the score network. The forward transition of (3.1) is

$$P_t(x_t \mid x_0) = \mathsf{N}\Big(e^{-t/2}x_0,\ (1-e^{-t})I\Big).$$

Therefore, the marginal density p_t admits the representation

$$p_t(x_t) = \int p_t(x_t \mid x_0) p_{\mathrm{data}}(x_0)\mathrm{d}x_0.$$

Using this representation, we can rewrite the expectation with respect to P_t in (3.4) as a joint expectation: for any measurable function φ, it holds that

$$\mathbb{E}_{X\sim P_t}[\varphi(X)] = \mathbb{E}_{X_0\sim P_{\mathrm{data}}}\mathbb{E}_{X_t\sim P_t(\cdot\mid X_0)}[\varphi(X_t)].$$

We apply the last display to $\varphi(x) = \nabla \cdot s(x,t)$. For each fixed x_0 and t, consider

$$\mathbb{E}_{X_t\sim P_t(\cdot\mid x_0)}\big[\nabla \cdot s(X_t,t)\big] = \int (\nabla \cdot s(x_t,t)) p(x_t \mid x_0)\mathrm{d}x_t.$$

Now apply integration by parts *with respect to* x_t under the Gaussian density $p_t(\cdot \mid x_0)$:

$$\begin{aligned}\int (\nabla \cdot s(x_t,t)) p_t(x_t \mid x_0)\mathrm{d}x_t &= -\int \langle s(x_t,t), \nabla_{x_t} p_t(x_t \mid x_0)\rangle \mathrm{d}x_t \\ &= -\mathbb{E}_{X_t\sim P_t(\cdot\mid x_0)}\left[\langle s(X_t,t), \nabla_{x_t}\log p_t(X_t \mid x_0)\rangle\right], \end{aligned}\tag{3.5}$$

where again boundary terms vanish due to p_t being Gaussian. Substituting (3.5) into (3.4) yields an equivalent objective,

$$\int_0^T \mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}\left[\|s(X_t,t)\|_2^2 - 2\left\langle s(X_t,t), \nabla_{x_t}\log p_t(X_t \mid X_0)\right\rangle\right] \mathrm{d}t. \tag{3.6}$$

Lastly, we complete the square in $s(X_t,t)$ by adding $\|\nabla_{x_t}\log p_t(X_t|X_0)\|_2^2$. Minimizing (3.6) is equivalent to minimizing

$$\widehat{s} \in \operatorname*{argmin}_{s\in\mathcal{F}} \int_0^T \mathbb{E}_{X_0\sim P_{\text{data}}}\mathbb{E}_{X_t\sim P_t(\cdot|X_0)}\left[\left\|s(X_t,t) - \nabla_{x_t}\log p_t(X_t \mid X_0)\right\|_2^2\right]\mathrm{d}t. \tag{3.7}$$

This is precisely the *Denoising Score Matching* (DSM) objective: we regress the score model $s(\cdot,t)$ towards the *conditional score* of the known forward Gaussian corruption kernel. Moreover, $\nabla_{x_t}\log p_t(X_t \mid X_0)$ is explicit:

$$\nabla_{x_t}\log p_t(X_t \mid X_0) = -\frac{X_t - e^{-t/2}X_0}{1-e^{-t}} = -\frac{1}{\sqrt{1-e^{-t}}}\frac{X_t - e^{-t/2}X_0}{\sqrt{1-e^{-t}}}.$$

Connection to DDPM training. The term in blue is precisely the added Gaussian noise. As a result, we can reparameterize the score function by a network that predicts the added noise. Define $\epsilon(x,t)$ and set

$$s(x,t) = -\frac{1}{\sqrt{1-e^{-t}}}\epsilon(x,t). \tag{3.8}$$

Substituting (3.8) into (3.7) yields an equivalent objective (up to a known weight function $w(t)$):

$$\widehat{\epsilon} \in \operatorname*{argmin}_{\epsilon} \int_0^T \mathbb{E}_{X_0\sim P_{\text{data}}}\mathbb{E}_{\epsilon\sim\mathsf{N}(0,I)}\left[w(t)\|\epsilon(e^{-t/2}X_0 + \sqrt{1-e^{-t}}\,\epsilon, t) - \epsilon\|_2^2\right]\mathrm{d}t. \tag{3.9}$$

Equation (3.9) is the continuous-time analogue of the discrete-time DDPM noise-prediction objective: sample a time, generate a noisy observation from the forward process, and regress the network output to the injected noise. In the next section, we will revisit discretization and show that the DDPM training loss and the DDPM/DDIM samplers can be viewed as discrete approximations of the continuous-time objects above.

4 Connecting Continuous and Discrete Time

This section relates the continuous-time formulation to the discrete-time encoder/decoder chains from earlier chapters. The punchline is that DDPM-style models can be viewed as *time discretizations* of a continuous-time diffusion, and conversely, a continuous-time model induces a family of discrete-time algorithms depending on the solver.

4.1 Discretizing the Forward Process

Fix a grid $0 = t_0 < t_1 < \cdots < t_K = T$ with K sub-intervals. A basic Euler-Maruyama discretization of (3.1) yields

$$X_{t_{k+1}} = X_{t_k} - \frac{1}{2}\Delta t_k X_{t_k} + \sqrt{\Delta t_k}Z_k, \qquad Z_k \sim \mathsf{N}(0,I),$$

where $\Delta t_k = t_{k+1} - t_k$. Suppose Δt_k is small. We have

$$\begin{aligned} X_{t_{k+1}} &= \left(1 - \frac{1}{2}\Delta t_k\right) X_{t_k} + \sqrt{\Delta t_k} Z_k \\ &= \sqrt{1 - \Delta t_k} X_{t_k} + \sqrt{\Delta t_k} Z_k + o(\Delta t_k). \end{aligned}$$

By choosing $\alpha_{t_k} = 1 - \Delta t_k$, we obtain

$$X_{t_{k+1}} = \sqrt{\alpha_{t_k}} X_{t_k} + \sqrt{1 - \alpha_{t_k}} Z_k + o(\Delta t_k),$$

which is the encoder distribution in DDPM. It is clear that for a different sequence α_{t_k}, we only need to change the rate function $\beta(t)$ accordingly.

4.2 Discretizing the Backward Process

Discretizing (3.2) on the same grid $0 = t_0 < \cdots < t_K = T$ via Euler-Maruyama update yields

$$\overline{X}_{t_{k+1}} = \overline{X}_{t_k} + \Big(\frac{1}{2}\overline{X}_{t_k} + \nabla \log p_{T-t_k}(\overline{X}_{t_k})\Big)\Delta t_k + \sqrt{\Delta t_k}\, Z_k, \qquad Z_k \sim \mathsf{N}(0, I). \tag{4.1}$$

We approximate the unknown score function by $\widehat{s}$, and the reverse-time sampler becomes

$$\overline{X}_{t_{k+1}} = \overline{X}_{t_k} + \Big(\frac{1}{2}\overline{X}_{t_k} + \widehat{s}(\overline{X}_{t_k}, T - t_k)\Big)\Delta t_k + \sqrt{\Delta t_k}\, Z_k. \tag{4.2}$$

To connect this to the *discrete-time* DDPM decoder, we need to translate the score parameterization into the familiar noise-prediction (or data-prediction) parameterization.

Score-denoising conversion via Tweedie's formula (VP marginal). From Example 2.1, the VP forward process implies the conditional Gaussian marginal

$$X_t \mid X_0 \sim \mathsf{N}\Big(\sqrt{\bar{\alpha}(t)} X_0, \big(1 - \bar{\alpha}(t)\big) I\Big), \qquad \bar{\alpha}(t) = \exp(-t).$$

Equivalently, we have

$$X_t = \sqrt{\bar{\alpha}(t)} X_0 + \sqrt{1 - \bar{\alpha}(t)}\, \epsilon, \qquad \epsilon \sim \mathsf{N}(0, I). \tag{4.3}$$

A key identity underpinning the continuous- and discrete-time formulations is that the *posterior denoiser* $\mathbb{E}[X_0 \mid X_t = x]$ can be written *exactly* in terms of the score function $\nabla \log p_t(x)$. This is a form of *Tweedie's formula* for Gaussian noise.

Lemma 4.1 (Tweedie's formula for VP)**.** Assume $X_t = \sqrt{\bar{\alpha}(t)} X_0 + \sqrt{1 - \bar{\alpha}(t)}\, \epsilon$ with $\epsilon \sim \mathsf{N}(0, I)$ independent of X_0, and let p_t be the marginal density of X_t. Then for any $t \in (0, T]$, it holds that

$$\mathbb{E}[X_0 \mid X_t = x] = \frac{1}{\sqrt{\bar{\alpha}(t)}}\Big(x + \big(1 - \bar{\alpha}(t)\big)\, \nabla \log p_t(x)\Big).$$

Equivalently, the marginal score can be expressed in terms of the posterior mean:

$$\nabla \log p_t(x) = \frac{1}{1 - \bar{\alpha}(t)}\Big(\sqrt{\bar{\alpha}(t)}\, \mathbb{E}[X_0 \mid X_t = x] - x\Big).$$

Proof. Fix $t \in (0, T]$. Under the VP forward process, we have

$$X_t = \sqrt{\bar{\alpha}(t)}\, X_0 + \sqrt{1 - \bar{\alpha}(t)}\, \epsilon, \qquad \epsilon \sim \mathsf{N}(0, I).$$

Thus the conditional density of X_t given $X_0 = x_0$ is Gaussian:

$$p_t(x \mid x_0) = \frac{1}{(2\pi(1 - \bar{\alpha}(t)))^{d/2}} \exp\Big(- \frac{\|x - \sqrt{\bar{\alpha}(t)}\, x_0\|_2^2}{2(1 - \bar{\alpha}(t))} \Big).$$

The marginal density of X_t is the Gaussian convolution,

$$p_t(x) = \int p_{\text{data}}(x_0)\, p_t(x \mid x_0)\, \mathrm{d}x_0. \tag{4.4}$$

We differentiate (4.4) with respect to x and interchange ∇_x and the integral (justified by dominated convergence) to obtain

$$\nabla_x p_t(x) = \int p_{\text{data}}(x_0)\, \nabla_x p_t(x \mid x_0)\, \mathrm{d}x_0. \tag{4.5}$$

For a Gaussian density in x, we have the identity,

$$\nabla_x p_t(x \mid x_0) = -\frac{x - \sqrt{\bar{\alpha}(t)}\, x_0}{1 - \bar{\alpha}(t)}\, p_t(x \mid x_0).$$

Substituting into (4.5) yields

$$\nabla p_t(x) = -\frac{1}{1 - \bar{\alpha}(t)} \int \big(x - \sqrt{\bar{\alpha}(t)}\, x_0\big)\, p_{\text{data}}(x_0)\, p_t(x \mid x_0)\, \mathrm{d}x_0.$$

Multiply both sides by $1 - \bar{\alpha}(t)$ and expand the integrand:

$$\big(1 - \bar{\alpha}(t)\big) \nabla p_t(x) = -x \int p_{\text{data}}(x_0)\, p_t(x \mid x_0)\, \mathrm{d}x_0 + \sqrt{\bar{\alpha}(t)} \int x_0\, p_{\text{data}}(x_0)\, p_t(x \mid x_0)\, \mathrm{d}x_0.$$

The first integral equals $p_t(x)$ by (4.4). For the second integral, applying Bayes' rule gives rise to

$$p(x_0 \mid x) = \frac{p_{\text{data}}(x_0)\, p_t(x \mid x_0)}{p_t(x)} \quad \Longrightarrow \quad \int x_0\, p_{\text{data}}(x_0)\, p_t(x \mid x_0)\, \mathrm{d}x_0 = p_t(x)\, \mathbb{E}[X_0 \mid X_t = x].$$

Therefore, we have

$$\big(1 - \bar{\alpha}(t)\big) \nabla p_t(x) = -x\, p_t(x) + \sqrt{\bar{\alpha}(t)}\, p_t(x)\, \mathbb{E}[X_0 \mid X_t = x].$$

Divide both sides by $p_t(x)$ to obtain

$$\big(1 - \bar{\alpha}(t)\big) \nabla \log p_t(x) = -x + \sqrt{\bar{\alpha}(t)}\ \mathbb{E}[X_0 \mid X_t = x].$$

Solving for the posterior mean gives the desired Tweedie's formula,

$$\mathbb{E}[X_0 \mid X_t = x] = \frac{1}{\sqrt{\bar{\alpha}(t)}} \Big(x + \big(1 - \bar{\alpha}(t)\big) \nabla \log p_t(x) \Big).$$

□

Tweedie's formula says: *up to a known affine transformation, learning the score is the same as learning the posterior mean of X_0 given X_t.* This is the precise bridge between the SDE and the DDPM/DDIM viewpoint. Combining (4.3) and Tweedie's formula, we obtain

$$\begin{aligned}\mathbb{E}[\epsilon \mid X_t = x] &= \frac{x - \sqrt{\bar{\alpha}(t)}\,\mathbb{E}[X_0 \mid X_t = x]}{\sqrt{1-\bar{\alpha}(t)}} \\ &= -\sqrt{1-\bar{\alpha}(t)}\,\nabla \log p_t(x).\end{aligned}$$

This implies that the score function is *exactly* proportional to the conditional mean of the injected noise. Connecting to the noise-prediction in DDPM, we have

$$\widehat{s}(x,t) \approx \nabla \log p_t(x) = -\frac{1}{\sqrt{1-\bar{\alpha}(t)}}\,\epsilon_\theta(x,t), \tag{4.6}$$

where $\epsilon_\theta(x,t)$ is the noise prediction network in DDPM, trained to approximate $\mathbb{E}[\epsilon \mid X_t = x]$.

Plugging (4.6) into (4.2) yields

$$\overline{X}_{t_{k+1}} = \overline{X}_{t_k} + \Big(\frac{1}{2}\overline{X}_{t_k} - \frac{1}{\sqrt{1-\bar{\alpha}(T-t_k)}}\,\epsilon_\theta(\overline{X}_{t_k}, T-t_k)\Big)\Delta t_k + \sqrt{\Delta t_k}\,Z_k. \tag{4.7}$$

This is a valid sampler and similar to DDPM with a different time scheduling.

4.3 Turning off Stochasticity in DDIM

The backward process (3.2) does not allow you to control the stochasticity. To connect to DDIM, there is a family of processes that preserve the same marginals of the forward process (3.1) and allows an active control on the stochasticity. Such processes are given by

$$\mathrm{d}\overline{X}_t = \left[\frac{1}{2}\overline{X}_t + \frac{1+\lambda^2}{2}\nabla \log p_{T-t}(\overline{X}_t)\right]\mathrm{d}t + \lambda\,\mathrm{d}\overline{W}_t, \tag{4.8}$$

where $\lambda \in [0,1]$. As a side remark, we typically term (3.2) as the backward process, but do not term (4.8) as backward processes.

As λ increases, the Brownian motion part has increasing variance. When $\lambda = 1$, we recover the forward process in (3.1). When $\lambda = 0$, we obtain a fully deterministic process

$$\mathrm{d}\overline{X}_t = \left[\frac{1}{2}\overline{X}_t + \frac{1}{2}\nabla \log p_{T-t}(\overline{X}_t)\right]\mathrm{d}t.$$

Discretizing (4.8) on the time grid $\{t_k\}_{k=0}^K$ gives the update

$$\overline{X}_{t_{k+1}} = \overline{X}_{t_k} + \Big(\frac{1}{2}\overline{X}_{t_k} + \frac{1+\lambda^2}{2}\nabla \log p_{T-t_k}(\overline{X}_{t_k})\Big)\Delta t_k + \lambda\sqrt{\Delta t_k}Z_k. \tag{4.9}$$

Replacing the score function by its estimator $\widehat{s}$ and noticing the equivalence of score function and noise-prediction formula, we derive the continuous-time DDIM-style sampling.

Summary of the discretization viewpoint. We can now interpret the discrete-time sampling algorithms from Chapters 2-3 as follows:

1. The *forward* DDPM encoder is an Euler discretization of the VP forward SDE (3.1).
2. The *reverse* DDPM decoder is a discretization of the reverse-time SDE (3.2), with the score represented by a learned noise-predictor.
3. DDIM-style sampling arises by a discretization to modified SDEs (4.8), which maintains the marginal distribution and surpresses stochasticity.

5 Sampling Algorithms

Given a trained score network $\widehat{s}$, new sample generation amounts to simulating either the reverse-time SDE (3.2) or the DDIM family (4.8). In addition to the Euler-Maruyama discretization, this section organizes several major algorithmic options.

5.1 Exponential Integrator (Piecewise Linear Backward Dynamics)

A useful way to understand many samplers in diffusion models is to notice that the reverse-time dynamics contain a *linear drift* plus a *learned (nonlinear) score function*. For example, the backward process (3.2) has drift

$$\frac{1}{2}\overline{X}_t + \widehat{s}(\overline{X}_t, T-t) \quad \text{with} \quad \nabla \log p_t \text{ replaced by } \widehat{s}.$$

The exponential-integrator philosophy is to integrate the linear part *exactly* over each step, and only approximate the nonlinear term. To make this concrete, consider a single step from t_k to $t_{k+1} = t_k + \Delta t_k$ for the (learned) reverse-time SDE/ODE drift. Over this short interval, we approximate the nonlinear score by a *constant function*. Concretely, for $t \in [t_k, t_{k+1}]$, we approximate the drift using

$$\frac{1}{2}\overline{X}_t + \widehat{s}(\overline{X}_{t_k}, T-t_k).$$

Note that, the input to the score network is fixed at $\overline{X}_{t_k}$ and $T-t_k$, the starting state of the interval $[t_k, t_{k+1}]$. An important consequence of such a drift approximation is that the nonlinear term $\widehat{s}(\overline{X}_{t_k}, T-t_k)$ is fixed on the interval $[t_k, t_{k+1}]$, rendering a piecewise linear SDE for approximating the original backward SDE. In particular, the piecewise linear approximation is

$$\mathrm{d}\overline{X}_t = \left[\frac{1}{2}\overline{X}_t + \widehat{s}(\overline{X}_{t_k}, T-t_k)\right]\mathrm{d}t + \mathrm{d}\overline{W}_t \quad \text{for} \quad t \in [t_k, t_{k+1}]. \tag{5.1}$$

A closed-form solution of (5.1) can be derived (left as an exercise), which reads as

$$\overline{X}_{t_{k+1}} = e^{\frac{\Delta t_k}{2}}\overline{X}_{t_k} + 2\big(e^{\frac{\Delta t_k}{2}} - 1\big)\,\widehat{s}(\overline{X}_{t_k}, T-t_k) + \sqrt{e^{\Delta t_k}-1}\,Z_k, \qquad Z_k \sim \mathsf{N}(0, I). \tag{5.2}$$

This is an *exponential-integrator step*: the linear drift $\frac{1}{2}\overline{X}_t$ is integrated exactly through the factor $e^{\Delta t_k/2}$, while the nonlinear score term is approximated by a piecewise constant function. When Δt_k is small, we have

$$e^{\frac{\Delta t_k}{2}} = 1 + \tfrac{1}{2}\Delta t_k + \mathcal{O}(\Delta t_k^2), \qquad 2\big(e^{\frac{\Delta t_k}{2}} - 1\big) = \Delta t_k + \mathcal{O}(\Delta t_k^2), \qquad e^{\Delta t_k} - 1 = \Delta t_k + \mathcal{O}(\Delta t_k^2).$$

Substituting these expansions into (5.2) shows that the exponential-integrator step reduces to the familiar Euler-Maruyama update

$$\overline{X}_{t_{k+1}} = \overline{X}_{t_k} + \Big(\tfrac{1}{2}\overline{X}_{t_k} + \widehat{s}(\overline{X}_{t_k}, T - t_k)\Big)\Delta t_k + \sqrt{\Delta t_k}\, Z_k + \text{higher-order terms},$$

i.e., it matches Euler-Maruyama at leading order, but with a more faithful treatment of the linear drift beyond first order. In practice, this often stabilizes large step sizes.

5.2 A General Template: Approximating the Score Field Within Each Step

The derivation above highlights a general recipe for sampler design:

1. Start from a reverse-time dynamics (SDE (3.2) or the DDIM family (4.8)).
2. On each interval $[t_k, t_{k+1}]$, *approximate the time-varying score* $\widehat{s}(\overline{X}_t, T - t)$ by a simple surrogate that is easy to integrate together with the linear drift.
3. Use the resulting locally-approximated dynamics to derive a closed-form (or cheaply computable) one-step update.

In the exponential-integrator step above, the surrogate is *piecewise constant in time* and evaluated at the left endpoint:

$$\widehat{s}(\overline{X}_t, T - t) \;\approx\; \widehat{s}(\overline{X}_{t_k}, T - t_k), \qquad t \in [t_k, t_{k+1}].$$

This choice uses *one* score-network evaluation per step.

Higher-order approximations. The natural next move is to use a more sophisticated approximation to the score term. A minimal upgrade is a *piecewise linear* approximation:

$$\widehat{s}(\overline{X}_t, T - t) \approx (1 - \rho)\, \widehat{s}(\overline{X}_{t_k}, T - t_k) + \rho\, \widehat{s}(\widetilde{X}_{t_{k+1}}, T - t_{k+1}), \qquad \rho = \frac{t - t_k}{\Delta t_k},$$

where $\widetilde{X}_{t_{k+1}}$ is a predictor of the endpoint state. This uses *two* score evaluations per step and typically yields a second-order correction, in the same spirit as Heun/trapezoidal updates [3].

More generally, one can build higher-order schemes by:

- using midpoint or multi-stage evaluations (Runge-Kutta type ideas),
- interpolating the score term with higher-degree polynomials in time,
- or reusing past score evaluations in multistep formulas.

The continuous-time view clarifies what these methods are doing: they are not changing the learned model $\widehat{s}$, but rather improving how we *integrate* the reverse-time dynamics by inserting higher-order information about the score within each interval. Diffusion-specialized solvers (e.g., DPM-Solver / DPM-Solver++ and related variants) can be understood as carefully engineered instances of this template, achieving high accuracy with very few network calls by exploiting integral formulations of diffusion dynamics [4, 5].

6 Statistical and Sampling Theory

The goal of theory in this chapter is conceptually simple: we want to bound the discrepancy between the *generated distribution* $\widehat{P}$ (the output law of our sampler using a learned score $\widehat{s}$) and the *ground-truth data distribution* P_{data}. To do so, it is helpful to separate two fundamentally different sources of errors.

1. **Statistical error** (learning): $\widehat{s}$ is trained from finite data and is not equal to the true score $\nabla \log p_t(x)$.
2. **Sampling error** (simulation): even if we had $\nabla \log p_t(x)$, we still approximate the reverse-time dynamics by a numerical solver with finitely many steps.

This decomposition clarifies what "better training" can fix (statistical error) and what "better samplers" can fix (sampling error).

Statistical error. From the regression viewpoint, statistical error aims to bound an integrated generalization error:

$$\mathcal{E}_{\text{stat}}^2 = \int_0^T \mathbb{E}\Big[\|\widehat{s}(X_t, t) - \nabla \log p_t(X_t)\|_2^2\Big] \mathrm{d}t,$$

in terms of sample size, function class complexity, and regularity of p_t. This is the part of the theory that looks like standard learning theory: generalization bounds, bias-variance tradeoffs, and (in structured settings) rates that depend on intrinsic dimension rather than ambient dimension.

Sampling error. Sampling error arises because we do not simulate the exact reverse-time dynamics.

1. We replace the true score $\nabla \log p_t$ by an approximate score $\widehat{s}$ in the drift.
2. We discretize time and simulate using a finite-step numerical solver.

Both create a *perturbed* stochastic process. A natural mathematical tool to compare such processes is *Girsanov's theorem*, which relates the laws of two diffusions that share the same diffusion coefficient but have different drifts.

References

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

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

[3] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[4] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

[5] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, pages 1–22, 2025.

[6] Wenpin Tang and Hanyang Zhao. Score-based diffusion models via stochastic differential equations–a technical tutorial. *arXiv preprint arXiv:2402.07487*, 2024.

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