1 Flow and Diffusion Models

In the previous section, we formalized generative modeling as sampling from a data distribution p_{data}, and reviewed the fundamentals. In this section[1], we show that sampling could be achieved via the transformation of samples from a simple distribution p_{init}, such as the Gaussian $\mathcal{N}(0, I_d)$, to samples from the target distribution p_{data}, and describe how the desired transformation can be obtained as the simulation of a suitably constructed differential equation. For example, *flow matching* and *diffusion models* involve simulating ordinary differential equations (ODEs) and stochastic differential equations (SDEs), respectively.

The goal of this section is to define and construct these generative models as they will be used throughout the remainder of the notes. Specifically:

1. we define ODEs and SDEs and discuss their simulation;
2. we describe how to parameterize an ODE/SDE using a deep neural network;
3. this leads to the definition of a flow model and diffusion model and the fundamental algorithms to sample from them.

In later sections, we then explore how to train these models.

1.1 Flow Models

We start by defining ordinary differential equations (ODEs). A solution to an ODE is defined by a trajectory, i.e. a function of the form

$$X : [0, 1] \to \mathbb{R}^d, \qquad t \mapsto X_t,$$

that maps from time t to some location in space $\mathbb{R}^d$.

Every ODE is defined by a *vector field* u, i.e. a function

$$u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d, \qquad (x, t) \mapsto u_t(x),$$

so that for every time t and location x we get a vector $u_t(x) \in \mathbb{R}^d$ specifying a velocity in space. An ODE imposes a condition on a trajectory: we want a trajectory X that "follows along the lines" of the vector field u_t, starting at a point x_0. We may formalize such a trajectory as the solution to

$$\frac{d}{dt} X_t = u_t(X_t) \tag{1a}$$

$$X_0 = x_0. \tag{1b}$$

[1]This lecture note largely follows from the MIT course "Introduction to Flow Matching and Diffusion Models".

We may now ask: if we start at $X_0 = x_0$ at $t = 0$, where are we at time t? This question is answered by a function called the *flow*, which is a solution to the ODE

$$\phi : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d, \qquad (x_0, t) \mapsto \phi_t(x_0), \tag{2a}$$

$$\frac{d}{dt}\phi_t(x_0) = u_t(\phi_t(x_0)), \tag{2b}$$

$$\phi_0(x_0) = x_0. \tag{2c}$$

For a given initial condition $X_0 = x_0$, a trajectory of the ODE is recovered via $X_t = \phi_t(X_0)$. Therefore, vector fields, ODEs, and flows are, intuitively, three descriptions of the same object: **vector fields define ODEs whose solutions are flows**.

Theorem 1.1 (Flow existence and uniqueness)

If $u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is continuously differentiable with a bounded derivative, then the ODE in (2a)–(2c) has a unique solution given by a flow ϕ_t. In this case, ϕ_t is a diffeomorphism for all t, i.e. ϕ_t is continuously differentiable with a continuously differentiable inverse ϕ_t^{-1}.

Note that the assumptions required for the existence and uniqueness of a flow are almost always fulfilled in machine learning, as we use neural networks to parameterize $u_t(x)$ and they typically have bounded derivatives under mild architectural/regularity assumptions. Therefore, the theorem should not be a concern but rather good news: flows exist and are unique solutions to ODEs in our cases of interest. A proof can be found, for example, in Perko (2013); Coddington et al. (1956).

Example 1.1 (Linear vector fields)

Consider a vector field $u_t(x)$ that is linear in x, i.e. $u_t(x) = -\theta x$ for $\theta > 0$. Then the function

$$\phi_t(x_0) = \exp(-\theta t)\, x_0 \tag{3}$$

defines a flow solving (2b)–(2c). Indeed, $\phi_0(x_0) = x_0$ and

$$\begin{aligned}
\frac{d}{dt}\phi_t(x_0) &= \frac{d}{dt}\big(\exp(-\theta t)\, x_0\big) \\
&= -\theta \exp(-\theta t)\, x_0 \\
&= -\theta\, \phi_t(x_0) \\
&= u_t(\phi_t(x_0)).
\end{aligned}$$

In Fig. 1, we visualize a flow of this form converging to 0 exponentially in $d = 2$ dimensions.

Simulating an ODE. In general, it is not possible to compute the flow ϕ_t explicitly if u_t is not as simple as a linear function. In these cases, one uses numerical methods to simulate ODEs. One of the simplest methods is the Euler method. We initialize with $X_0 = x_0$ and update via

$$X_{t+h} = X_t + h\, u_t(X_t), \qquad t = 0, h, 2h, \ldots, 1 - h, \qquad h = \frac{1}{n} > 0. \tag{4}$$

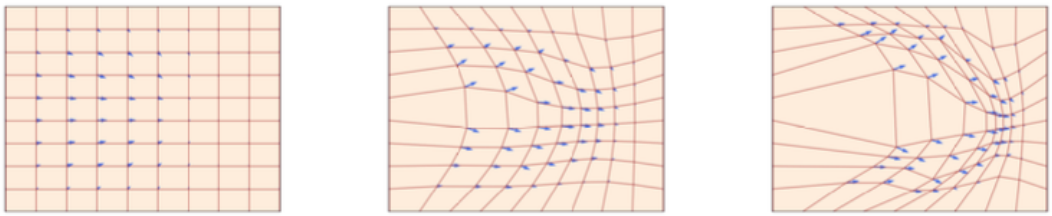

Figure 1: A flow $\phi_t : \mathbb{R}^d \to \mathbb{R}^d$ (red square grid) is defined by a velocity field $u_t : \mathbb{R}^d \to \mathbb{R}^d$ (visualized with blue arrows) that prescribes its instantaneous movements at all locations (here, $d = 2$). We show three different times t. As one can see, a flow is a diffeomorphism that "warps" space.

For this class, the Euler method will be good enough.

To give a taste of a more complex method, consider Heun's method:

$$X'_{t+h} = X_t + h\, u_t(X_t) \qquad \text{(initial guess of new state)}$$

$$X_{t+h} = X_t + \frac{h}{2}\big(u_t(X_t) + u_{t+h}(X'_{t+h})\big). \qquad \text{(update with average slope correction)}$$

Intuitively, the Heun's method is as follows: it takes a first guess X'_{t+h} of what the next step could be but corrects the direction initially taken via an updated guess.

Flow models. We can now construct a generative model via an ODE. Remember that our goal is to convert a simple distribution p_{init} into a complex distribution p_{data}. The simulation of an ODE is thus a natural choice for this transformation.

A *flow model* is described by

$$X_0 \sim p_{\text{init}} \qquad \text{(random initialization)}$$

$$\frac{d}{dt} X_t = u_t^\theta(X_t) \qquad \text{(ODE)}$$

where the vector field u_t^θ is a neural network with parameters θ:

$$u^\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d, \qquad (x,t) \mapsto u_t^\theta(x).$$

Our goal is to make the endpoint X_1 have distribution p_{data}, i.e.

$$X_1 \sim p_{\text{data}} \quad \Leftrightarrow \quad \phi_1^\theta(X_0) \sim p_{\text{data}},$$

where ϕ_t^θ is the flow corresponding to u_t^θ. Note: although it is called a *flow model*, the neural network parameterizes the *vector field*, not the flow. To compute the flow, we simulate the ODE. In Algorithm 1, we summarize the sampling procedure using the Euler method.

1.2 Diffusion Models

Stochastic differential equations (SDEs) extend deterministic ODE trajectories with stochastic trajectories. A stochastic trajectory is commonly called a stochastic process $(X_t)_{0 \le t \le 1}$:

$$X_t \text{ is a random variable for every } 0 \le t \le 1,$$

$$X : [0,1] \to \mathbb{R}^d,\ t \mapsto X_t \text{ is a random trajectory for every draw of } X.$$

In particular, simulating the same stochastic process twice can yield different outcomes.

Algorithm 1: Sampling from a flow model with Euler method.

Input: Neural network vector field u_t^θ, number of steps n
Set $t = 0$;
Set step size $h = \frac{1}{n}$;
Draw a sample $X_0 \sim p_{\text{init}}$;
for $i = 1, \dots, n$ **do**
  $X_{t+h} \leftarrow X_t + h\, u_t^\theta(X_t)$;
  Update $t \leftarrow t + h$;
return X_1;

Brownian motion. SDEs are constructed via Brownian motion (Wiener process): a continuous-time stochastic process with independent, normally distributed increments. It originates from the study of physical diffusion processes. A Brownian motion $W = (W_t)_{0 \le t \le 1}$ is a stochastic process such that $W_0 = 0$, the paths are continuous, and:

1. **Normal increments:** $W_t - W_s \; sim \mathcal{N}\big(0, (t-s) I_d\big)$ for all $0 \le s < t \le 1$.
2. **Independent increments:** For a ny $0 \le t_0 < t_1 < \cdots < t_n = 1$, the increments $W_{t_1} - W_{t_0}, \dots, W_{t_n} - W_{t_{n-1}}$ are independent.

We can simulate Brownian motion approximatel y with step size $h > 0$ by setting $W_0 = 0$ and updating

$$W_{t+h} = W_t + \sqrt{h}\, \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, I_d), \qquad t = 0, h, 2h, \dots, 1-h. \tag{5}$$

From ODEs to SDEs. The idea of an SDE is to extend the deterministic dynamics of an ODE by adding stochastic dynamics driven by a Brownian motion. Because everything is stochastic, we may no longer take the derivative as in (1a). Instead, we use an infinitesimal increment description, that does not use derivatives directly. For this, let us rewrite trajectories $(X_t)_{0 \le t \le 1}$ of an ODE via infinitesimal updates:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t(X_t) && \text{(expression via derivatives)} \\
&\Leftrightarrow \quad \frac{1}{h}\big(X_{t+h} - X_t\big) = u_t(X_t) + R_t(h), \\
&\Leftrightarrow X_{t+h} = X_t + h\, u_t(X_t) + h\, R_t(h), \;\; \lim_{h \to 0} R_t(h) = 0, \\
& && \text{(expression via infinitesitmal updates)}
\end{aligned}$$

where we use the definition of derivatives in (i), and $R_t(h)$ describe a negligible deterministic remainder term for small h, i.e., $\lim_{h \to 0} R_t(h) = 0$.

To make it stochastic, we add a Brownian increment scaled by a diffusion coefficient $\sigma_t \ge 0$:

$$X_{t+h} = X_t + h\, u_t(X_t) + \sigma_t\big(W_{t+h} - W_t\big) + h\, R_t(h), \tag{6}$$

where the term $\sigma_t\big(W_{t+h} - W_t\big)$ adds stochasticity to the dynamics, and $R_t(h)$ describes a stochastic error term such that the standard deviation $\mathbb{E}\big[\|R_t(h)\|^2\big]^{1/2} \to 0$ goes to zero

as $h \to 0$. The above describes a stochastic differential equation (SDE). It is common to denote it in the following symbolic notation:

$$\mathrm{d}X_t = u_t(X_t)\,\mathrm{d}t + \sigma_t\,\mathrm{d}W_t \qquad \blacktriangleright \text{ SDE} \tag{7a}$$

$$X_0 = x_0 \qquad \blacktriangleright \text{ initial condition} \tag{7b}$$

However, always keep in mind that the "$\mathrm{d}X_t$"-notation above is a purely informal notation of eq. (6). Unfortunately, SDEs do not have a flow map ϕ_t anymore. This is because the value X_t is not fully determined by $X_0 \sim p_{\text{init}}$ anymore as the evolution itself is stochastic. Still, in the same way as for ODEs, we have the following existence and uniqueness result for SDEs.

Theorem 1.2 (SDE solution existence and uniqueness)

If $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is continuously differentiable with a bounded derivative and σ_t is continuous, then the SDE in (7a)-(7b) has a solution given by a unique stochastic process $(X_t)_{0 \leq t \leq 1}$ satisfying (6).

If this was a stochastic calculus class, we would spend several lectures proving this theorem and constructing SDEs with full mathematical rigor, i.e. constructing a Brownian motion from first principles and constructing the process X_t via stochastic integration. As we focus on machine learning in this class, we refer to Mao (2007) for a more technical treatment. Finally, note that every ODE is also an SDE - simply with a vanishing diffusion coefficient $\sigma_t = 0$. Therefore, for the remainder of this class, when we speak about SDEs, we consider ODEs as a special case.

Example 1.2 (Ornstein–Uhlenbeck process)

Let $\sigma_t = \sigma \geq 0$ be constant and let the drift be $u_t(x) = -\theta x$ for $\theta > 0$. Then the SDE

$$dX_t = -\theta X_t\,dt + \sigma\,dW_t \tag{8}$$

defines an Ornstein–Uhlenbeck (OU) process. The drift term $-\theta x$ pulls the process toward 0 by always going the inverse direction of where I am, while the diffusion term adds noise. This process converges towards a Gaussian distribution $\mathcal{N}(0, \sigma^2/(2\theta))$ if we simulate it for $t \to \infty$. For $\sigma = 0$ we recover the deterministic flow case (3).

Simulating an SDE. A more intuitive way of thinking about SDEs is given by answering the question: How might we simulate an SDE? The simplest such scheme is known as the Euler-Maruyama method, and is essentially to SDEs what the Euler method is to ODEs. Using the Euler-Maruyama method, we initialize $X_0 = x_0$ and update iteratively via

$$X_{t+h} = X_t + h\,u_t(X_t) + \sigma_t\sqrt{h}\,\varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, I_d), \tag{9}$$

where $h = 1/n > 0$ is a step size hyperparameter for $n \in \mathbb{N}$. In other words, to simulate using the Euler-Maruyama method, we take a small step in the direction of $u_t(X_t)$ as well as add a little bit of Gaussian noise scaled by $\sigma_t\sqrt{h}$. When simulating SDEs in this class (such as in the accompanying labs), we will usually stick to the Euler-Maruyama method.

Algorithm 2: Sampling from a Diffusion Model with Euler–Maruyama method.

Input: Neural network u_t^θ, number of steps n, diffusion coefficient σ_t
Set $t = 0$;
Set step size $h = \frac{1}{n}$;
Draw a sample $X_0 \sim p_{\text{init}}$;
for $i = 1, \dots, n$ **do**
  Draw a sample $\varepsilon \sim \mathcal{N}(0, I_d)$;
  $X_{t+h} \leftarrow X_t + h\, u_t^\theta(X_t) + \sigma_t \sqrt{h}\, \varepsilon$;
  Update $t \leftarrow t + h$;
return X_1;

Diffusion models. We can now construct a generative model via an SDE in the same way as we did for ODEs. Remember that our goal was to convert a simple distribution pinit into a complex distribution pdata. Like for ODEs, the simulation of an SDE randomly initialized with $X_0 \sim p_{\text{init}}$ is a natural choice for this transformation. To parameterize this SDE, we can simply parameterize its central ingredient, the vector field u_t, a neural network u_t^θ. A *diffusion model* is thus given by

$$\begin{aligned} \mathrm{d}X_t &= u_t^\theta(X_t)\,\mathrm{d}t + \sigma_t\,\mathrm{d}W_t && \blacktriangleright \text{ SDE} \\ X_0 &\sim p_{\text{init}}. && \blacktriangleright \text{ random initialization} \end{aligned}$$

In Algorirthm 2, we describe the procedure by which to sample from a diffusion model with the Euler-Maruyama method. We summarize the results of this section as follows.

Summary 1.1 (SDE generative model)

Throughout this document, a diffusion model consists of a neural network vector field u_t^θ and a fixed diffusion coefficient σ_t:

$$\begin{aligned} \textbf{Neural network: } u^\theta &: \mathbb{R}^d \times [0,1] \to \mathbb{R}^d, \qquad (x,t) \mapsto u_t^\theta(x) \text{ with parameter } \theta, \\ \textbf{Fixed: } \sigma &: [0,1] \to [0,\infty), \qquad t \mapsto \sigma_t. \end{aligned}$$

To obtain samples from our SDE model (i.e. generate objects), the procedure is as follows:

$$\begin{aligned} \textbf{Init:} \quad & X_0 \sim p_{\text{init}} && \blacktriangleright \text{ Initialize with simple distribution, e.g. Gaussian} \\ \textbf{Simulate:} \quad & \mathrm{d}X_t = u_t^\theta(X_t)\,\mathrm{d}t + \sigma_t\,\mathrm{d}W_t && \blacktriangleright \text{ Simulate SDE from 0 to 1} \\ \textbf{Goal:} \quad & X_1 \sim p_{\text{data}}. && \blacktriangleright \text{ Goal: make } X_1 \text{ have distribution } p_{\text{data}} \end{aligned}$$

A diffusion model with $\sigma_t \equiv 0$ is a flow model.

2 Constructing the Training Target: The Marginalization Trick

In the previous section, we constructed flow and diffusion models where we obtain trajectories $(X_t)_{0\le t\le 1}$ by simulating

$$X_0 \sim p_{\text{init}}, \qquad dX_t = u_t^\theta(X_t)\,dt \qquad \text{(flow model)}, \tag{10}$$

$$X_0 \sim p_{\text{init}}, \qquad dX_t = u_t^\theta(X_t)\,dt + \sigma_t\,dW_t \qquad \text{(diffusion model)}. \tag{11}$$

If we randomly initialize the parameters θ, simulating will produce nonsense. We need to train the neural network by minimizing a loss such as mean-squared error:

$$L(\theta) = \left\| u_t^\theta(x) - u_t^{\text{target}}(x) \right\|^2,$$

where u_t^{target} is a training target vector field. To derive a training algorithm, we proceed in two steps:

1. derive an equation/formula for the training target u_t^{target};
2. describe a training algorithm that approximates it.

The goal of this chapter is to derive a formula for u_t^{target} such that the corresponding ODE/SDE converts p_{init} into p_{data}. Along the way we will encounter the continuity equation and the Fokker–Planck equation.

Remark 2.1

There are a number of different approaches to deriving a training target for flow and diffusion models. The approach we present here is both general and arguably simple and aligns with many modern formulations, though it may differ from older presentations.

2.1 Conditional and Marginal Probability Path

The first step is to specify a probability path: an interpolation between noise p_{init} and data p_{data}. For a data point $z \in \mathbb{R}^d$, denote by δ_z the Dirac delta distribution at z.

A *conditional (interpolating) probability path* is a family of distributions $p_t(x \mid z)$ such that

$$p_0(\cdot \mid z) = p_{\text{init}}, \qquad p_1(\cdot \mid z) = \delta_z \quad \text{for all } z \in \mathbb{R}^d. \tag{12}$$

Every conditional path induces a *marginal* probability path $p_t(x)$ defined by: sample $z \sim p_{\text{data}}$ and then $x \sim p_t(\cdot \mid z)$:

$$z \sim p_{\text{data}}, \quad x \sim p_t(\cdot \mid z) \quad \Longrightarrow \quad x \sim p_t, \qquad \blacktriangleright \text{ sampling from marginal path} \tag{13}$$

$$p_t(x) = \int p_t(x \mid z)\, p_{\text{data}}(z)\, dz. \qquad \blacktriangleright \text{ density of marginal path} \tag{14}$$

Note that we know how to sample from p_t but we do not know the density values $p_t(x)$ as the integral is intractable. Because of (12), the marginal path interpolates:

$$p_0 = p_{\text{init}}, \qquad p_1 = p_{\text{data}}. \tag{15}$$

Example 2.1 (Gaussian conditional probability path)

A popular choice (used in denoising diffusion models or DDM for short) is the Gaussian probability path. Let α_t, β_t be noise schedulers: continuously differentiable, monotonic functions such that

$$\alpha_0 = 0,\ \alpha_1 = 1, \qquad \beta_0 = 1,\ \beta_1 = 0.$$

Define the conditional probability path

$$p_t(\cdot \mid z) = \mathcal{N}\big(\alpha_t z,\ \beta_t^2 I_d\big). \qquad \blacktriangleright \text{ Gaussian conditional path} \tag{16}$$

which by the conditions we imposed on α_t and β_t, fulfills

$$p_0(\cdot \mid z) = \mathcal{N}(0, I_d) = p_{\text{init}}, \quad p_1(\cdot \mid z) = \delta_z,$$

where we have used the fact that a normal distribution with zero variance and mean z is just δ_z. Therefore, this choice of $p_t(x|z)$ fulfills eq. (12) for $p_{\text{init}} = \mathcal{N}(0, I_d)$ and is therefore a valid conditional interpolating path. The Gaussian conditional probability path has several useful properties which makes it especially amenable to our goals, and because of this we will use it as our prototypical example of a conditional probability path for the rest of the section. Moreover, sampling from the marginal path can be written as

$$z \sim p_{\text{data}},\ \varepsilon \sim \mathcal{N}(0, I_d) \ \Rightarrow\ x = \alpha_t z + \beta_t \varepsilon \sim p_t. \quad \blacktriangleright \text{ sampling from marginal Gaussian path} \tag{17}$$

2.2 Conditional and Marginal Vector Fields

We now construct a training target u_t^{target} for a flow model using a probability path p_t. The idea is to construct u_t^{target} from simple components that we can derive analytically by hand.

Theorem 2.1 (Marginalization trick)

For every data point $z \in \mathbb{R}^d$, let $u_t^{\text{target}}(\cdot \mid z)$ be a conditional vector field such that the corresponding ODE yields the conditional probability path:

$$X_0 \sim p_{\text{init}}, \quad \frac{d}{dt} X_t = u_t^{\text{target}}(X_t \mid z) \quad \Longrightarrow \quad X_t \sim p_t(\cdot \mid z), \quad 0 \le t \le 1. \tag{18}$$

Define the marginal vector field by

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x \mid z)\, \frac{p_t(x \mid z)\, p_{\text{data}}(z)}{p_t(x)}\, dz. \tag{19}$$

Then the ODE with drift u_t^{target} follows the marginal probability path:

$$X_0 \sim p_{\text{init}}, \quad \frac{d}{dt} X_t = u_t^{\text{target}}(X_t) \quad \Longrightarrow \quad X_t \sim p_t, \quad 0 \le t \le 1. \tag{20}$$

In particular, $X_1 \sim p_{\text{data}}$. Thus we might say "u_t^{target} converts noise p_{init} into data p_{data} through the marginal probability path p_t".

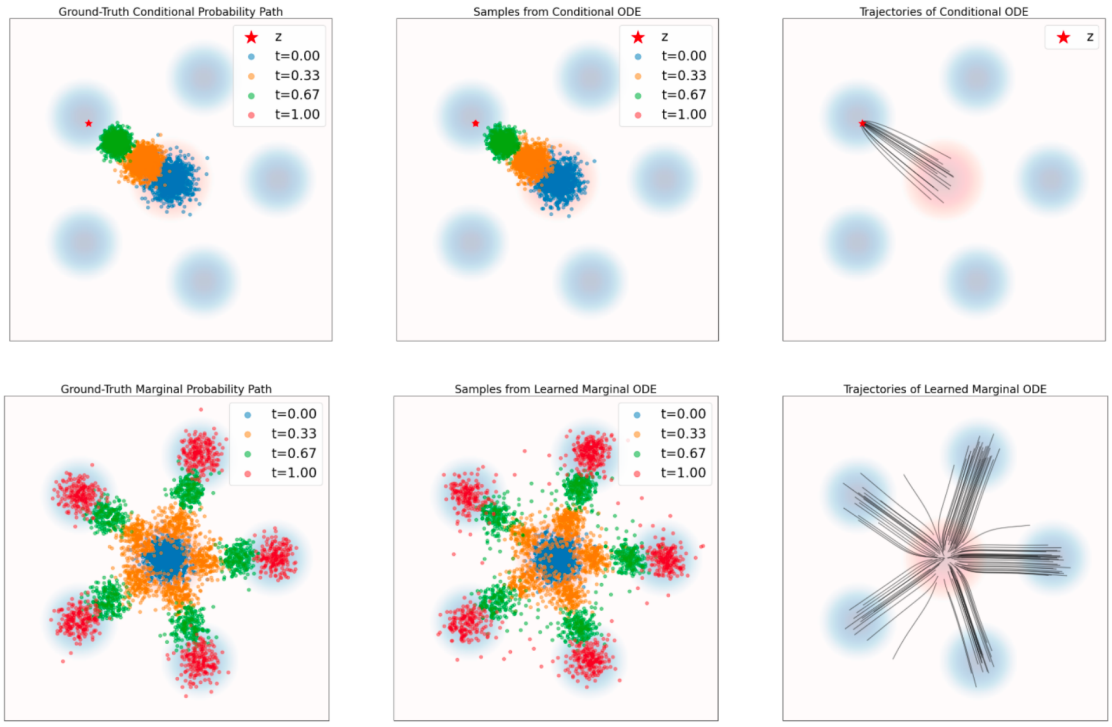

Figure 2: Illustration of Theorem 2.1. Simulating a probability path with ODEs. Data distribution p_{data} in blue background. Gaussian p_{init} in red background. Top row: Conditional probability path. Left: Ground truth samples from conditional path $p_t(\cdot|z)$. Middle: ODE samples over time. Right: Trajectories by simulating ODE with $u_t^{\text{target}}(x|z)$ in eq. (21). Bottom row: Simulating a marginal probability path. Left: Ground truth samples t from p_t. Middle: ODE samples over time. Right: Trajectories by simulating ODE with marginal vector field $u_t^{\text{target}}(x)$. As one can see, the conditional vector field follows the conditional probability path and the marginal vector field follows the marginal probability path.

See fig. 2 for an illustration. Before we prove the marginalization trick, let us first explain why it is useful: The marginalization trick from Theorem 2.1 allows us to construct the marginal vector field from a conditional vector field. This simplifies the problem of finding a formula for a training target significantly as we can often find a conditional vector field $u_t^{\text{target}}(\cdot \mid z)$ satisfying eq. (18) analytically by hand (i.e. by just doing some algebra ourselves). Let us illustrate this by deriving a conditional vector field $u_t^{\text{target}}(x \mid z)$ for our running example of a Gaussian probability path.

Example 2.2 (Target ODE for Gaussian probability paths)

Let $p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$ as in (16) for noise schedulers α_t, β_t. Let $\dot{\alpha}_t = \partial_t \alpha_t$ and $\dot{\beta}_t = \partial_t \beta_t$. Then the conditional vector field

$$u_t^{\text{target}}(x \mid z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x \tag{21}$$

is valid in the sense of (18) in Theorem 2.1. In fig. 2, we confirm this visually by

comparing samples from the conditional probability path (ground truth) to samples from simulated ODE trajectories of this flow. As you can see, the distribution match. We will now prove this.

Proof. Define the conditional flow

$$\phi_t^{\text{target}}(x \mid z) = \alpha_t z + \beta_t x. \tag{22}$$

If X_t is the ODE trajectory of $\phi_t^{\text{target}}(\cdot \mid z)$ with $X_0 \sim p_{\text{init}} = \mathcal{N}(0, I_d)$, then by definition

$$X_t = \phi_t^{\text{target}}(X_0 \mid z) = \alpha_t z + \beta_t X_0 \sim \mathcal{N}(\alpha_t z, \beta_t^2 I_d) = p_t(\cdot \mid z).$$

We conclude that the trajectories are distributed like the conditional probability path (i.e, eq. (18) is fulfilled).
It remains to extract the vector field $u_t^{\text{target}}(x|z)$ from $\phi_t^{\text{target}}(x|z)$. By the definition of a flow, it holds that

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t^{\text{target}}(x \mid z) = u_t^{\text{target}}\big(\phi_t^{\text{target}}(x \mid z) \mid z\big), \qquad \forall x, z \in \mathbb{R}^d.$$

Thus,

$$\begin{aligned}
\dot{\alpha}_t z + \dot{\beta}_t x &= u_t^{\text{target}}(\alpha_t z + \beta_t x \mid z) && \text{(i)}\\
\iff \quad \dot{\alpha}_t z + \dot{\beta}_t \left(\frac{x - \alpha_t z}{\beta_t}\right) &= u_t^{\text{target}}(x \mid z) && \text{(ii)}\\
\iff \quad u_t^{\text{target}}(x \mid z) &= \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x, && \text{(iii)}
\end{aligned}$$

where where in (i) we used the definition of $\phi_t^{\text{target}}(x|z)$, in (ii) we reparameterized $x \mapsto (x - \alpha_t z)/\beta_t$, and in (iii) we just did some algebra. Note that the last equation is the conditional Gaussian vector field as we defined in eq. 21. This proves the statement. □

The remainder of this section will now prove Theorem 2.1 via the continuity equation, a fundamental result in mathematics and physics. We require the divergence operator, which is defined as:

$$\operatorname{div}(v_t)(x) = \sum_{i=1}^{d} \frac{\partial}{\partial x_i} (v_t(x))_i. \tag{23}$$

Theorem 2.2 (Continuity Equation)

Consider a flow model with vector field u_t^{target} and $X_0 \sim p_{\text{init}}$. Then $X_t \sim p_t$ for all $0 \le t \le 1$ if and only if

$$\partial_t p_t(x) = -\operatorname{div}\left(p_t u_t^{\text{target}}\right)(x), \qquad x \in \mathbb{R}^d, \quad 0 \le t \le 1, \tag{24}$$

where $\partial_t p_t(x) = \frac{d}{dt} p_t(x)$ denotes the time-derivative of $p_t(x)$. Equation (24) is known as the *continuity equation*.

Intuition. Before we move on, let us try and understand intuitively the continuity equation. The left-hand side $\partial_t p_t(x)$ describes how much the probability $p_t(x)$ at x changes over time. Intuitively, the change should correspond to the net inflow of probability mass. For a flow model, a particle X follows along the vector field u_t^{target}. As you might recall from physics, the divergence measures a sort of net outflow from the vector field. Therefore, the negative divergence measures the net inflow. Scaling this by the total probability mass currently residing at x, we get that the net $-\operatorname{div}(p_t u_t)$ measures the total inflow of probability mass. Since probability mass is conserved, the left-hand and right-hand side of the equation should be the same! We now proceed with a proof of the marginalization trick from theorem 2.1.

Proof of Theorem 2.1. By Theorem 2.2, we have to show that the marginal vector field u_t^{target}, as defined as in eq. (19), satisfies the continuity equation. We can do this by direct calculation:

$$\begin{aligned}
\partial_t p_t(x) &\overset{(i)}{=} \partial_t \int p_t(x \mid z)\, p_{\text{data}}(z)\, dz \\
&= \int \partial_t p_t(x \mid z)\, p_{\text{data}}(z)\, dz \\
&\overset{(ii)}{=} \int -\operatorname{div}\Big(p_t(\cdot \mid z)\, u_t^{\text{target}}(\cdot \mid z)\Big)(x)\, p_{\text{data}}(z)\, dz \\
&\overset{(iii)}{=} -\operatorname{div}\left(\int p_t(x \mid z)\, u_t^{\text{target}}(x \mid z)\, p_{\text{data}}(z)\, dz\right)(x) \\
&\overset{(iv)}{=} -\operatorname{div}\left(p_t(x) \int u_t^{\text{target}}(x \mid z)\, \frac{p_t(x \mid z)\, p_{\text{data}}(z)}{p_t(x)}\, dz\right)(x) \\
&\overset{(v)}{=} -\operatorname{div}\big(p_t\, u_t^{\text{target}}\big)(x). \qquad (2.1)
\end{aligned}$$

where in (i) we used the definition of $p_t(x)$ in eq. (13), in (ii) we used the continuity equation for the conditional probability path $p_t(\cdot \mid z)$, in (iii) we swapped the integral and divergence operator, in (iv) we multiplied and divided by $p_t(x)$, and in (v) we used eq. (19). The beginning and end of the above chain of equations show that the continuity equation is fulfilled for u_t^{target}. By Theorem 2.2, this is enough to imply eq. (20), and we are done. □

2.3 Conditional and Marginal Score Functions

We have just successfully constructed a training target for a flow model. We now extend the reasoning to SDEs. Define the marginal score function of p_t as $\nabla \log p_t(x)$. We can use this to extend the ODE from the previous section to an SDE, as the following result demonstrates.

Theorem 2.3 (SDE extension trick)

Let $u_t^{\text{target}}(x \mid z)$ and $u_t^{\text{target}}(x)$ be as before, and let $\sigma_t \geq 0$ be a diffusion coefficient. Then the SDE

$$X_0 \sim p_{\text{init}}, \qquad dX_t = \left(u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t)\right) dt + \sigma_t\, dW_t \qquad (25)$$

follows the same marginal probability path:

$$X_t \sim p_t, \qquad 0 \leq t \leq 1. \qquad (26)$$

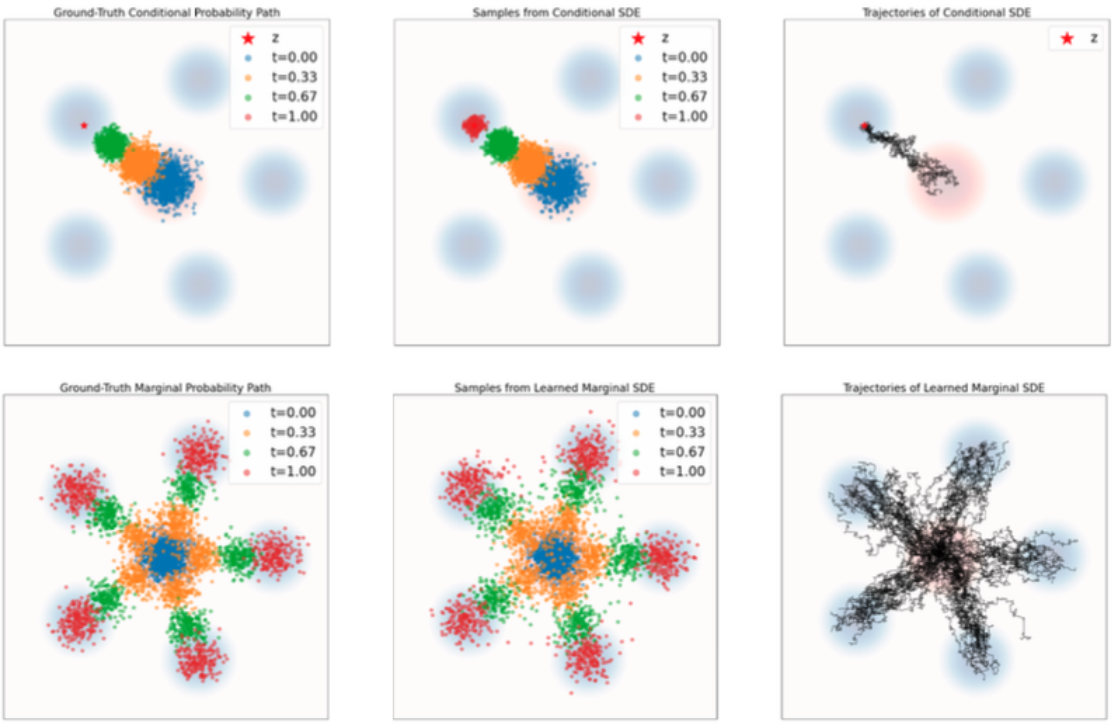

Figure 3: Illustration of Theorem 2.3. Simulating a probability path with SDEs. This repeats the plots from fig. 2 with SDE sampling using eq. (25). Data distribution p_{data} in blue background. Gaussian p_{init} in red background. Top row: Conditional path. Bottom row: Marginal probability path. As one can see, the SDE transports samples from p_{init} into samples from δ_z (for the conditional path) and to p_{data} (for the marginal path).

In particular, $X_1 \sim p_{\text{data}}$. The same identity holds if we replace the marginal $p_t(x)$ and $u_t^{\text{target}}(x)$ with the conditional $p_t(x \mid z)$ and $u_t^{\text{target}}(x \mid z)$.

We illustrate the above theorem in fig. 3. The formula in Theorem 2.3 is useful because, siimlar to before, the marginal score can be expressed as an integral of the conditional score function $\nabla \log p_t(x \mid z)$:

$$\nabla \log p_t(x) = \frac{\nabla \int p_t(x \mid z) p_{\text{data}}(z)\, dz}{p_t(x)} = \int \nabla \log p_t(x \mid z) \frac{p_t(x \mid z)\, p_{\text{data}}(z)}{p_t(x)}\, dz, \tag{27}$$

and the conditional score function $\nabla \log p_t(x|z)$ is usually available in closed form, as illustrated by the following example.

Example 2.3 (Score function for Gaussian probability paths)

For the Gaussian path $p_t(x|z) = \mathcal{N}(x; \alpha_t z, \beta_t^2 I_d)$,

$$\nabla \log p_t(x|z) = \nabla \log \mathcal{N}(x; \alpha_t z, \beta_t^2 I_d) = -\frac{x - \alpha_t z}{\beta_t^2}. \tag{28}$$

Note that the score is a linear function of x. This is a unique feature of Gaussian distributions.

In the remainder of this section, we will prove Theorem 2.3 via the Fokker–Planck equation, a fundamental result in stochastic analysis that extends the continuity equation from ODEs to SDEs. To state the Fokker–Planck equation, we first define the Laplacian operator Δ via

$$\Delta w_t(x) = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2} w_t(x) = \operatorname{div}(\nabla w_t)(x). \tag{29}$$

Theorem 2.4 (Fokker–Planck equation)

Let p_t be a probability path and consider the SDE

$$X_0 \sim p_{\text{init}}, \qquad dX_t = u_t(X_t)\, dt + \sigma_t\, dW_t.$$

Then $X_t \sim p_t$ for all $0 \le t \le 1$ if and only if the Fokker–Planck equation holds:

$$\partial_t p_t(x) = -\operatorname{div}\big(p_t u_t\big)(x) + \frac{\sigma_t^2}{2}\, \Delta p_t(x), \qquad x \in \mathbb{R}^d, \quad 0 \le t \le 1. \tag{30}$$

The Fokker-Planck equation can be proved by using test functions, which is not required in this class. Note that the continuity equation is recovered from the Fokker-Planck equation when $\sigma_t = 0$. The additional Laplacian term Δp_t might be hard to rationalize at first. Those familiar with physics will note that the same term also appears in the heat equation (which is in fact a special case of the Fokker-Planck equation). Heat diffuses through a medium. We also add a diffusion process (not a physical but a mathematical one) and hence we add this additional Laplacian term. Let us now use the Fokker-Planck equation to help us prove Theorem 2.3.

Proof of Theorem 2.3. By Theorem 2.4, we need to show that the SDE defined in eq. (25) satisfies the Fokker-Planck equation for p_t. We can do this by direct calculation:

$$\begin{aligned}
\partial_t p_t(x) &\overset{(i)}{=} -\operatorname{div}\big(p_t u_t^{\text{target}}\big)(x) \\
&\overset{(ii)}{=} -\operatorname{div}\big(p_t u_t^{\text{target}}\big)(x) - \frac{\sigma_t^2}{2}\, \Delta p_t(x) + \frac{\sigma_t^2}{2}\, \Delta p_t(x) \\
&\overset{(iii)}{=} -\operatorname{div}\big(p_t u_t^{\text{target}}\big)(x) - \operatorname{div}\left(\frac{\sigma_t^2}{2}\, \nabla p_t\right)(x) + \frac{\sigma_t^2}{2}\, \Delta p_t(x) \\
&\overset{(iv)}{=} -\operatorname{div}\big(p_t u_t^{\text{target}}\big)(x) - \operatorname{div}\left(p_t \left[\frac{\sigma_t^2}{2}\, \nabla \log p_t\right]\right)(x) + \frac{\sigma_t^2}{2}\, \Delta p_t(x) \\
&\overset{(v)}{=} -\operatorname{div}\left(p_t \left[u_t^{\text{target}} + \frac{\sigma_t^2}{2}\, \nabla \log p_t\right]\right)(x) + \frac{\sigma_t^2}{2}\, \Delta p_t(x).
\end{aligned} \tag{2.2}$$

where in (i) we used the Continuity Equation, in (ii) we added and subtracted the same term, in (iii) we used the definition of the Laplacian (eq. (29)), in (iv) we used that $\nabla \log p_t = \frac{\nabla p_t}{p_t}$, and in (v) we used the linearity of the divergence operator. The above derivation shows that the SDE defined in eq. (25) satisfies the Fokker–Planck equation for p_t. By theorem 15, this implies $X_t \sim p_t$ for $0 \le t \le 1$, as desired.

□

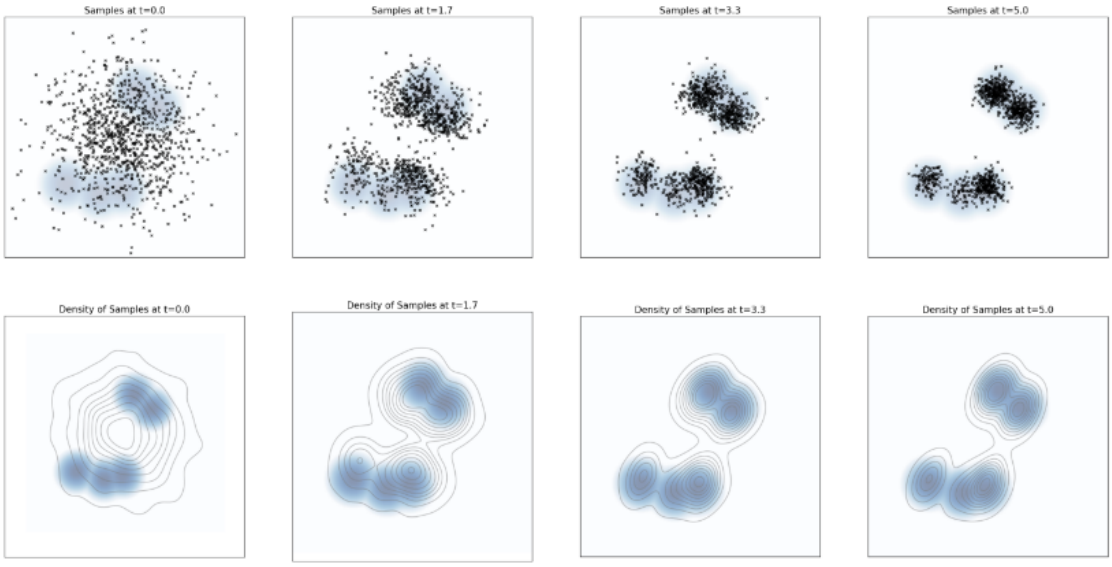

Figure 4: Top row: Particles evolving under the Langevin dynamics given by eq. (31), with $p(x)$ taken to be a Gaussian mixture with 5 modes. Bottom row: A kernel density estimate of the same samples shown in the top row. As one can see, the distribution of samples converges to the equilibrium distribution p (blue background colour).

Remark 2.2 (Langevin dynamics)

The above construction has a famous special case when the probability path is static, i.e., $p_t = p$ for a fixed distribution. In this case, we set $u_t^{\text{target}} = 0$ in (25) to obtain

$$dX_t = \frac{\sigma_t^2}{2} \nabla \log p(X_t)\, dt + \sigma_t\, dW_t, \tag{31}$$

which is commonly known as Langevin dynamics.
Since p_t is time-independent, we have $\partial_t p_t(x) = 0$. It follows immediately from Theorem 2.3 that the dynamics satisfy the Fokker–Planck equation, and that the stationary solution is given by the static path $p_t = p$. Consequently, p is a stationary distribution of Langevin dynamics, i.e.,

$$X_0 \sim p \quad \Longrightarrow \quad X_t \sim p \qquad (t \geq 0).$$

As with many Markov chains, these dynamics converge to the stationary distribution p under fairly general conditions, with an example given in fig. 4. In particular, if $X_0 \sim p' \neq p$, then under mild assumptions $X_t \to p$ as $t \to \infty$. This makes Langevin dynamics extremely useful, and it forms the basis of, e.g., molecular dynamics simulations, as well as many Markov chain Monte Carlo (MCMC) methods in Bayesian statistics and the natural sciences.

Summary of results. Let us summarize the results of this section.

Summary 2.1 (Derivation of the training target)

The flow training target is the marginal field u_t^{target}. To construct it, we choose a conditional probability path $p_t(x \mid z)$ with $p_0(\cdot \mid z) = p_{\text{init}}$ and $p_1(\cdot \mid z) = \delta_z$. Next, we find a conditional vector field $u_t^{\text{target}}(x \mid z)$ that realizes this path:

$$X_0 \sim p_{\text{init}}, \quad dX_t = u_t^{\text{target}}(X_t \mid z)\, dt \implies X_t = \phi_t^{\text{target}}(X_0|z) \sim p_t(\cdot \mid z).$$

Then the marginal vector field

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x \mid z) \frac{p_t(x \mid z) p_{\text{data}}(z)}{p_t(x)}\, dz \tag{32}$$

follows the marginal probability path:

$$X_0 \sim p_{\text{init}}, \quad dX_t = u_t^{\text{target}}(X_t)\, dt \implies X_t \sim p_t, \quad 0 \le t \le 1, \tag{33}$$

so in particular $X_1 \sim p_{\text{data}}$. Thus u_t^{target} converts noise into data, as desired.

Extending to SDEs. For a time-dependent diffusion coefficient $\sigma_t \ge 0$, we can extend the above ODE to an SDE with the same marginal probability path:

$$X_0 \sim p_{\text{init}}, \quad dX_t = \left(u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right) dt + \sigma_t\, dW_t, \tag{34}$$

which yields

$$X_t \sim p_t, \quad 0 \le t \le 1, \qquad \text{hence } X_1 \sim p_{\text{data}}. \tag{35}$$

The score function $\nabla \log p_t(x)$ admits the marginalization identity

$$\nabla \log p_t(x) = \int \nabla \log p_t(x \mid z) \frac{p_t(x \mid z) p_{\text{data}}(z)}{p_t(x)}\, dz. \tag{36}$$

For the trajectories X_t of the above SDE, it holds that $X_1 \sim p_{\text{data}}$, so that the SDE "converts noise into data", as desired.

Gaussian path (an important example). An important example for both ODEs and SDEs is given by the Gaussian probability path. Specifically, we have

$$p_t(x \mid z) = \mathcal{N}(x; \alpha_t z, \beta_t^2 I_d), \tag{37}$$

$$u_t^{\text{target}}(x \mid z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x, \tag{38}$$

$$\nabla \log p_t(x \mid z) = -\frac{x - \alpha_t z}{\beta_t^2}, \tag{39}$$

for noise schedulers $\alpha_t, \beta_t \in \mathbb{R}$: continusouly differentiable, monotonic functions such that $\alpha_0 = 0, \alpha_1 = 1, \beta_0 = 1, \beta_1 = 0$:

References

Coddington, E. A., Levinson, N., and Teichmann, T. (1956). Theory of ordinary differential equations.

Mao, X. (2007). *Stochastic differential equations and applications*. Elsevier.

Perko, L. (2013). *Differential equations and dynamical systems*, volume 7. Springer Science & Business Media.