Faster Sampling: Advanced Solvers

Lecturer: Qiang Sun Lecture 08

Contents

1 Faster Sampling: Advanced Solvers

1.1 Motivation and conventions

Flow and diffusion models generate samples by *simulating a differential equation* that transports a simple distribution (noise) to the data distribution. In continuous time:

- **ODE sampling** follows a deterministic trajectory (often fewer steps);
- **SDE sampling** injects randomness (more diversity, usually more steps).

In this lecture we focus on *training-free acceleration*: the model is fixed after training and we only change the *numerical solver* used at inference. The dominant cost is the *number of function evaluations (*NFE*)*, i.e., the number of neural network calls performed by the solver. The definition of NFE is made precise below.

Definition 1 (Number of function evaluations (NFE))

Suppose a sampler uses N time steps and calls the neural network m times per step. Then

$$\mathrm{NFE} \triangleq mN.$$

Multistep methods reuse past evaluations and often have average $m \approx 1$ after a short warm-up. Classifier-free guidance typically doubles NFE because it evaluates the model twice (conditional/unconditional).

Remark 1 (Time convention)

We follow the diffusion/solver literature: $t = 0$ corresponds to *clean data* and $t = 1$ corresponds to *pure noise*. Hence sampling integrates the reverse-time dynamics from $t = 1$ down to $t = 0$. To reduce to the time convention in our previous lectures (noise $\to$ data with increasing time), we can set $t \to 1 - t$. In this section, we stick to the solver convention for consistency with the literature and to avoid notational clutter.

1.2 Forward process, backward process, and connection to flow models introduced in earlier lectures

Consider a forward (noising) SDE on $[0, 1]$:

$$\mathrm{d}X_t = f_t(X_t)\,\mathrm{d}t + g_t\,\mathrm{d}W_t, \qquad X_0 \sim p_{\mathrm{data}}. \tag{1.1}$$

Let p_t denote the density of X_t. Under standard regularity assumptions, the *reverse-time* SDE (integrated from $t = 1$ to $t = 0$) is

$$\mathrm{d}X_t = \Big(f_t(X_t) - g_t^2\,\nabla_x \log p_t(X_t)\Big)\,\mathrm{d}t + g_t\,\mathrm{d}\bar{W}_t, \qquad X_1 \sim p_{\mathrm{init}}, \tag{1.2}$$

and the associated *probability flow ODE (PF-ODE)* with the *same marginals* $\{p_t\}$ is

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = f_t(x_t) - \tfrac{1}{2} g_t^2\,\nabla_x \log p_t(x_t), \qquad x_1 \sim p_{\mathrm{init}}. \tag{1.3}$$

In practice the score $\nabla_x \log p_t(x)$ is unknown and replaced by a learned model. Here $\bar{W}_t$ denotes a Brownian motion under the reverse-time filtration; one convenient choice is $\bar{W}_t \triangleq W_1 - W_{1-t}$.

Connection to the course convention (noise $\to$ data with increasing time). In earlier lectures we often parameterize the *generative* dynamics forward in time, starting from noise and moving toward data. To translate the solver convention (integrate from $t = 1 \to 0$) into that convention, set

$$\tau \triangleq 1 - t, \qquad Y_\tau \triangleq X_{1-\tau}, \qquad q_\tau \triangleq p_{1-\tau}. \tag{1.4}$$

Then τ increases from 0 to 1 during generation, with $Y_0 \sim p_{\text{init}}$ and $Y_1 \sim p_{\text{data}}$.

Under this reparameterization, the PF-ODE (1.3) becomes

$$\frac{\mathrm{d}Y_\tau}{\mathrm{d}\tau} = -f_{1-\tau}(Y_\tau) + \tfrac{1}{2} g_{1-\tau}^2 \, \nabla_y \log q_\tau(Y_\tau), \tag{1.5}$$

and the reverse-time SDE (1.2) becomes the forward-in-τ generative SDE

$$\mathrm{d}Y_\tau = \Big(-f_{1-\tau}(Y_\tau) + g_{1-\tau}^2 \, \nabla_y \log q_\tau(Y_\tau)\Big)\,\mathrm{d}\tau + g_{1-\tau}\,\mathrm{d}W_\tau. \tag{1.6}$$

If we define the course-style generative drift

$$u_\tau(y) \triangleq -f_{1-\tau}(y) + \tfrac{1}{2} g_{1-\tau}^2 \, \nabla_y \log q_\tau(y), \tag{1.7}$$

then (1.5) is simply

$$\dot{Y}_\tau = u_\tau(Y_\tau),$$

and (1.6) is its stochastic counterpart

$$\mathrm{d}Y_\tau = \big(u_\tau(Y_\tau) + \tfrac{1}{2} g_{1-\tau}^2 \nabla \log q_\tau(Y_\tau)\big)\mathrm{d}\tau + g_{1-\tau}\mathrm{d}W_\tau,$$

matching the structure used in our earlier notes.

Remark 2 (Flow models as a special case)

If $g_t \equiv 0$, then (1.3) reduces to a deterministic flow ODE. In τ-time, (1.5) becomes $\dot{Y}_\tau = -f_{1-\tau}(Y_\tau)$, matching the pure flow setup from earlier lectures.

1.3 Gaussian perturbation schedules

Many diffusion models admit a Gaussian probability path (Gaussian perturbation kernel)

$$x_t \mid x_0 \sim \mathcal{N}(\alpha_t x_0,\, \beta_t^2 I_d), \qquad \alpha_0 = 1,\ \beta_0 = 0,\ \alpha_1 = 0,\ \beta_1 = 1. \tag{1.8}$$

The schedule (α_t, β_t) determines how signal/noise mix over time. When the forward drift is linear, $f_t(x) = \gamma_t x$, one can convert between the SDE coefficients (γ_t, g_t) and the kernel (α_t, β_t).

Proposition 2 (Relating (γ_t, g_t) and (α_t, β_t))

Assume the linear SDE $\mathrm{d}x_t = \gamma_t x_t \,\mathrm{d}t + g_t \,\mathrm{d}W_t$ has conditional law (1.8). Then, for any $\delta > 0$, we have the following relations for all $t \in [0, 1-\delta]$:

$$\gamma_t = \frac{\dot{\alpha}_t}{\alpha_t}, \qquad g_t^2 = \frac{\mathrm{d}}{\mathrm{d}t}\,\beta_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\,\beta_t^2. \tag{1.9}$$

Proof of Proposition 2. We assume throughout that W_t is a standard d-dimensional Brownian motion, that $\gamma : [0, 1-\delta] \to \mathbb{R}$ and $g : [0, 1-\delta] \to [0, \infty)$ are continuous for any $\delta > 0$, and that $\int_0^{1-\delta} g_s^2 \,\mathrm{d}s < \infty$ so the Itô integral below is well-defined. Write $X_t = x_t$.

Step 1: Explicit solution via an integrating factor (Itô product rule). Define the (deterministic) integrating factor

$$\Phi_t \triangleq \exp\Big(\int_0^t \gamma_u \,\mathrm{d}u\Big), \qquad \text{so that} \qquad \dot{\Phi}_t = \gamma_t \Phi_t, \quad \Phi_0 = 1.$$

We use the Itô product rule for semimartingales: if U_t, V_t are (scalar) semimartingales, then

$$\mathrm{d}(U_t V_t) = U_t \,\mathrm{d}V_t + V_t \,\mathrm{d}U_t + \mathrm{d}\langle U, V\rangle_t,$$

see, e.g., Øksendal (2003) (Ch. 4) or Karatzas and Shreve (1991) (Ch. 3).

Let $U_t = \Phi_t^{-1}$ which has finite variation, hence $\langle U, \cdot\rangle \equiv 0$, aka the quadratic covariation term vanishes because Φ_t^{-1} is a smooth, deterministic function. Apply the Itô product rule with $U_t = \Phi_t^{-1}$ and $V_t = X_t$. Since $\mathrm{d}(\Phi_t^{-1}) = -\gamma_t \Phi_t^{-1}\,\mathrm{d}t$, we obtain

$$\begin{aligned}
\mathrm{d}(\Phi_t^{-1} X_t) &= \Phi_t^{-1}\,\mathrm{d}X_t + X_t\,\mathrm{d}(\Phi_t^{-1}) \\
&= \Phi_t^{-1}\big(\gamma_t X_t\,\mathrm{d}t + g_t\,\mathrm{d}W_t\big) \;+\; X_t\big(-\gamma_t \Phi_t^{-1}\,\mathrm{d}t\big) \\
&= \Phi_t^{-1} g_t\,\mathrm{d}W_t.
\end{aligned}$$

Integrating from 0 to t yields

$$\Phi_t^{-1} X_t = X_0 + \int_0^t \Phi_s^{-1} g_s\,\mathrm{d}W_s,$$

and multiplying by Φ_t gives the explicit solution

$$X_t = \Phi_t X_0 + \int_0^t \frac{\Phi_t}{\Phi_s}\, g_s\,\mathrm{d}W_s. \tag{1.10}$$

Step 2: Matching the conditional mean. Taking conditional expectation given X_0 in (1.10) and using that an Itô integral has mean zero (again a standard fact; see, e.g., Øksendal, Ch. 3), we get

$$\mathbb{E}[X_t \mid X_0] = \Phi_t X_0.$$

By the assumed conditional law (1.8), $\mathbb{E}[X_t \mid X_0] = \alpha_t X_0$, hence

$$\alpha_t = \Phi_t.$$

Differentiating $\alpha_t = \exp(\int_0^t \gamma_u\,\mathrm{d}u)$ gives

$$\gamma_t = \frac{\dot{\alpha}_t}{\alpha_t}.$$

Step 3: Matching the conditional variance. Conditioned on X_0, the random part of X_t is the Itô integral

$$Z_t \triangleq \int_0^t \frac{\Phi_t}{\Phi_s}\, g_s\,\mathrm{d}W_s.$$

Since the integrand is deterministic (depends only on time), $Z_t \mid X_0$ is Gaussian with mean 0. Moreover, by Itô isometry (see, e.g., Øksendal (2003), Ch. 3),

$$\mathrm{Cov}(Z_t \mid X_0) = \left(\int_0^t \Big(\frac{\Phi_t}{\Phi_s}\Big)^2 g_s^2\,\mathrm{d}s\right) I_d.$$

Therefore,

$$\mathrm{Cov}(X_t \mid X_0) = \mathrm{Cov}(Z_t \mid X_0) = \left(\Phi_t^2 \int_0^t \frac{g_s^2}{\Phi_s^2}\,\mathrm{d}s\right) I_d.$$

Matching with (1.8), i.e. $\mathrm{Cov}(X_t \mid X_0) = \beta_t^2 I_d$, and using $\Phi_t = \alpha_t$, we obtain

$$\beta_t^2 \;=\; \alpha_t^2 \int_0^t \frac{g_s^2}{\alpha_s^2}\,\mathrm{d}s. \tag{1.11}$$

Step 4: Solving for g_t^2. Differentiate (1.11) with respect to t:

$$\frac{\mathrm{d}}{\mathrm{d}t}\beta_t^2 = 2\alpha_t\dot{\alpha}_t \int_0^t \frac{g_s^2}{\alpha_s^2}\,\mathrm{d}s + \alpha_t^2 \cdot \frac{g_t^2}{\alpha_t^2} = 2\frac{\dot{\alpha}_t}{\alpha_t}\,\beta_t^2 + g_t^2.$$

Rearranging yields

$$g_t^2 = \frac{\mathrm{d}}{\mathrm{d}t}\,\beta_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\,\beta_t^2,$$

which is exactly (1.9). □

1.4 Prediction targets: score, noise, data, velocity.

Recall that we assume the Gaussian probability path (1.8), i.e.

$$X_t \;=\; \alpha_t X_0 + \beta_t \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I_d) \text{ independent of } X_0, \tag{1.12}$$

so that $p_t(\cdot \mid X_0 = x_0) = \mathcal{N}(\alpha_t x_0, \beta_t^2 I_d)$.

Under this Gaussian path, there are several equivalent prediction targets. The most common ones are the score $\nabla_x \log p_t(x)$, the noise ε, the clean data x_0, and the velocity v_t. It is helpful to introduce them in that order, because each target can be derived from the previous one.

From score prediction to noise prediction. For (1.8), the conditional score is $\nabla_x \log p_t(x \mid x_0) = -(x-\alpha_t x_0)/\beta_t^2$. Writing $x_t = \alpha_t x_0 + \beta_t \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I_d)$, we obtain the marginal identity

$$\varepsilon^*(x_t, t) \triangleq \mathbb{E}[\varepsilon \mid x_t], \qquad \nabla_x \log p_t(x_t) = -\frac{1}{\beta_t}\,\varepsilon^*(x_t, t). \tag{1.13}$$

This explains why *ε-prediction* is often numerically more stable than direct score regression: as $t \to 0$, $\beta_t \to 0$ so the score can blow up like β_t^{-1}, whereas ε^* typically remains $\mathcal{O}(1)$.

From noise prediction to data prediction. The Gaussian relation also links the posterior-mean noise predictor to the posterior-mean data predictor. Indeed,

$$\varepsilon^*(x_t, t) = \mathbb{E}[\varepsilon \mid x_t] = \mathbb{E}\Big[\frac{x_t - \alpha_t x_0}{\beta_t} \;\Big|\; x_t\Big] = \frac{x_t - \alpha_t x_0^*(x_t, t)}{\beta_t}, \qquad x_0^*(x_t, t) \triangleq \mathbb{E}[x_0 \mid x_t].$$

Thus, ε-prediction and x_0-prediction are equivalent up to a known affine transformation determined by the schedule. In practice, one may train either target and recover the other immediately.

Velocity prediction. Another widely used target is the velocity

$$v_t \triangleq \dot{\alpha}_t x_0 + \dot{\beta}_t \varepsilon,$$

which combines the data and noise components along the Gaussian path. The corresponding oracle velocity predictor is

$$\begin{aligned} v^*(x_t, t) &\triangleq \mathbb{E}[\dot{\alpha}_t x_0 + \dot{\beta}_t \varepsilon \mid x_t] && \blacktriangleright \text{ velocity / } v\text{-prediction} \\ &= \dot{\alpha}_t \, x_0^*(x_t, t) + \dot{\beta}_t \, \varepsilon^*(x_t, t). && (1.14) \end{aligned}$$

Remark 3 (Relation to the common v-prediction target $\alpha_t\varepsilon - \beta_t x_0$)

If the schedule is normalized so that $\alpha_t^2 + \beta_t^2 = 1$, we may write $\alpha_t = \cos\theta_t$ and $\beta_t = \sin\theta_t$ for some angle θ_t. Then

$$\dot{\alpha}_t x_0 + \dot{\beta}_t \varepsilon = \dot{\theta}_t\big(\alpha_t \varepsilon - \beta_t x_0\big). \tag{1.15}$$

Thus predicting $\alpha_t\varepsilon - \beta_t x_0$ differs from predicting $\dot{\alpha}_t x_0 + \dot{\beta}_t\varepsilon$ only by the known scalar factor $\dot{\theta}_t$.

Why $v_t = \dot{\alpha}_t x_0 + \dot{\beta}_t\varepsilon$ is called the velocity. The quantity $\dot{\alpha}_t x_0 + \dot{\beta}_t\varepsilon$ is literally the time-derivative of the noising trajectory (1.12) when (x_0, ε) are held fixed. We derive this below.

Fix $x_0 \in \mathbb{R}^d$ and $\varepsilon \in \mathbb{R}^d$ and consider the deterministic curve

$$x_t \;\triangleq\; \phi_t(\varepsilon \mid x_0) \;\triangleq\; \alpha_t x_0 + \beta_t \varepsilon. \tag{1.16}$$

If $\varepsilon \sim \mathcal{N}(0, I_d)$, then x_t has conditional law $x_t \mid x_0 \sim \mathcal{N}(\alpha_t x_0, \beta_t^2 I_d)$, i.e. it realizes the Gaussian kernel.

Differentiating (1.16) in time gives the instantaneous tangent (velocity) along this curve:

$$\frac{\mathrm{d}}{\mathrm{d}t} x_t = \dot{\alpha}_t x_0 + \dot{\beta}_t \varepsilon. \tag{1.17}$$

Equivalently, one may express this velocity as a vector field in the current state x and the conditioning x_0. When $\beta_t \neq 0$, we can solve for the noise as $\varepsilon = (x - \alpha_t x_0)/\beta_t$ and substitute into (1.17) to obtain

$$u_t(x \mid x_0) \triangleq \dot{\alpha}_t x_0 + \dot{\beta}_t \frac{x - \alpha_t x_0}{\beta_t} = \frac{\dot{\beta}_t}{\beta_t}\, x + \Big(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\Big) x_0. \tag{1.18}$$

Evaluating at $x = x_t = \alpha_t x_0 + \beta_t\varepsilon$ recovers $u_t(x_t \mid x_0) = \dot{\alpha}_t x_0 + \dot{\beta}_t\varepsilon$, so this quantity is exactly the (conditional) velocity of the Gaussian noising trajectory. □

1.5 Semilinear ODE and exponential integrators

Assume the forward drift is linear, $f_t(x) = \gamma_t x$. Substituting (1.13) into the PF-ODE (1.3):

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = \gamma_t x_t - \tfrac{1}{2} g_t^2 \, \nabla_x \log p_t(x_t)$$

yields the *semilinear* ODE

$$\begin{aligned}\frac{\mathrm{d}x_t}{\mathrm{d}t} &= \gamma_t x_t + \frac{g_t^2}{2\beta_t}\,\varepsilon^*(x_t,t)\\ &= \gamma_t x_t + \frac{\frac{\mathrm{d}}{\mathrm{d}t}\beta_t^2 - 2\frac{\dot\alpha_t}{\alpha_t}\beta_t^2}{2\beta_t}\,\varepsilon^*(x_t,t)\\ &= \gamma_t x_t + \left(\dot\beta_t - \frac{\dot\alpha_t}{\alpha_t}\beta_t\right)\varepsilon^*(x_t,t)\\ &= \gamma_t x_t + b_t\,\varepsilon^*(x_t,t), \qquad \text{where}\ \ b_t \triangleq \alpha_t\,\frac{\mathrm{d}}{\mathrm{d}t}\frac{\beta_t}{\alpha_t}. \end{aligned} \tag{1.19}$$

The coefficient b_t is known from the schedule; the only learned term is ε_θ. We need the following lemma to derive exponential integrators for (1.19).

Lemma 3 (Variation of constants / Duhamel formula)

Let $\gamma : [0,1] \to \mathbb{R}$ be continuous and define the (scalar) propagator

$$E(s \to t) \;\triangleq\; \exp\Big(\int_s^t \gamma_u\,\mathrm{d}u\Big), \qquad s,t \in [0,1].$$

Consider the possibly nonlinear ODE

$$\dot x_t \;=\; \gamma_t x_t + F(x_t,t), \tag{1.20}$$

where $F : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is continuous in t and locally Lipschitz in x so that solutions exist and are unique. Then for any $s,t \in [0,1]$ the solution satisfies

$$x_t = E(s \to t)\,x_s + \int_s^t E(u \to t)\,F(x_u,u)\,\mathrm{d}u. \tag{1.21}$$

In particular, when $s > t$ (integrating backward in time), the same identity holds with an integral over a reversed interval, equivalently

$$x_t = E(s \to t)\,x_s - \int_t^s E(u \to t)\,F(x_u,u)\,\mathrm{d}u.$$

Proof of Lemma 3. We use an integrating factor argument.

Step 1: basic properties of $E(s \to t)$. By definition,

$$E(s \to s) = 1, \qquad \frac{\partial}{\partial t}E(s \to t) = \gamma_t\,E(s \to t), \qquad \frac{\partial}{\partial s}E(s \to t) = -\gamma_s\,E(s \to t).$$

Moreover, E has the semigroup (composition) property: for any r,s,t,

$$E(r \to t) = E(s \to t)\,E(r \to s), \tag{1.22}$$

since both sides equal $\exp(\int_r^t \gamma_u\,\mathrm{d}u)$.

Step 2: integrating factor. Fix $s \in [0,1]$ and define the integrating-factor transform

$$y_t \triangleq E(t \to s)\, x_t.$$

Differentiate using the product rule (here everything is deterministic, so ordinary calculus applies):

$$\begin{aligned}\dot{y}_t &= \frac{\mathrm{d}}{\mathrm{d}t} E(t \to s) \cdot x_t + E(t \to s) \cdot \dot{x}_t \\ &= \big(-\gamma_t E(t \to s)\big) x_t + E(t \to s)\big(\gamma_t x_t + F(x_t, t)\big) \\ &= E(t \to s)\, F(x_t, t).\end{aligned}$$

Integrating from s to t gives

$$y_t - y_s = \int_s^t E(u \to s)\, F(x_u, u)\, \mathrm{d}u.$$

Since $y_s = E(s \to s) x_s = x_s$, we have

$$E(t \to s)\, x_t = x_s + \int_s^t E(u \to s)\, F(x_u, u)\, \mathrm{d}u.$$

Step 3: return to x_t. Multiply both sides by $E(s \to t)$, noting that $E(s \to t) E(t \to s) = 1$, to obtain

$$x_t = E(s \to t)\, x_s + \int_s^t E(s \to t)\, E(u \to s)\, F(x_u, u)\, \mathrm{d}u.$$

Using the semigroup property (1.22) with $(r, s, t) = (u, s, t)$ yields $E(s \to t)\, E(u \to s) = E(u \to t)$, hence

$$x_t = E(s \to t)\, x_s + \int_s^t E(u \to t)\, F(x_u, u)\, \mathrm{d}u,$$

which is (1.21). The case $s > t$ is identical; the integral $\int_s^t$ is simply a negative-oriented integral, equivalently $-\int_t^s$. □

Taking $F(x_u, u) = b_u\, \varepsilon^*(x_u, u)$ in Lemma 3 gives the variation of constants formula for (1.19):

$$x_t = E(s \to t)\, x_s + \int_s^t E(u \to t)\, b_u\, \varepsilon^*(x_u, u)\, \mathrm{d}u. \tag{1.23}$$

Exponential-integrator samplers handle the linear multiplier E exactly and approximate only the integral.

Remark 4 (Why exponential integrators help)

For large steps, a plain Euler update approximates the linear multiplier $E(s \to t)$ by a first-order Taylor expansion, introducing a purely linear distortion. Exponential integrators apply the exact multiplier, so the dominant error comes from approximating the nonlinear integral only.

1.6 DDIM as exponential Euler

We work with a *Gaussian probability path*

$$p_t(\cdot \mid x_0) = \mathcal{N}(\alpha_t x_0, \beta_t^2 I_d),$$

and consider the probability-flow ODE written in the ε-prediction semilinear form

$$\dot{x}_t = \gamma_t x_t + b_t\, \varepsilon_\theta(x_t, t), \qquad t \in [0,1]. \tag{1.24}$$

For Gaussian schedules, the linear coefficient satisfies $\gamma_t = \dot{\alpha}_t/\alpha_t$, and it is convenient to parameterize the nonlinear coefficient as

$$b_t \;=\; \alpha_t \,\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\beta_t}{\alpha_t}\Big) \;=\; \dot{\beta}_t - \frac{\dot{\alpha}_t}{\alpha_t}\,\beta_t. \tag{1.25}$$

Propagator. Let $E(s \to t)$ be the propagator of the homogeneous linear ODE $\dot{z}_t = \gamma_t z_t$, i.e.

$$E(s \to t) = \exp\Big(\int_s^t \gamma_u \,\mathrm{d}u\Big).$$

When $\gamma_t = \dot{\alpha}_t/\alpha_t$, we obtain the closed form

$$E(s \to t) = \exp\Big(\int_s^t \frac{\dot{\alpha}_u}{\alpha_u}\,\mathrm{d}u\Big) = \frac{\alpha_t}{\alpha_s}. \tag{1.26}$$

Variation of constants in (α, β)-coordinates. For a denoising step $s > t$ (sampling runs backward from noise to data), the variation-of-constants formula (1.23) applied to (1.24) gives

$$x_t = E(s \to t)\, x_s + \int_s^t E(u \to t)\, b_u\, \varepsilon^*(x_u, u)\,\mathrm{d}u, \tag{1.27}$$

where ε^* denotes the oracle (posterior-mean) noise predictor. Substituting (1.26) and (1.25) into (1.27) yields

$$\begin{aligned} x_t &= \frac{\alpha_t}{\alpha_s} x_s + \int_s^t \frac{\alpha_t}{\alpha_u}\Big(\alpha_u \,\frac{\mathrm{d}}{\mathrm{d}u}\frac{\beta_u}{\alpha_u}\Big)\varepsilon^*(x_u, u)\,\mathrm{d}u \\ &= \frac{\alpha_t}{\alpha_s} x_s + \alpha_t \int_s^t \frac{\mathrm{d}}{\mathrm{d}u}\Big(\frac{\beta_u}{\alpha_u}\Big)\,\varepsilon^*(x_u, u)\,\mathrm{d}u. \end{aligned} \tag{1.28}$$

Exponential Euler $\Rightarrow$ DDIM. The *exponential Euler* approximation freezes the learned term at the *start of the step* (time s),

$$\varepsilon^*(x_u, u) \;\approx\; \varepsilon^*(x_s, s), \qquad u \in [t, s],$$

and integrates the scalar weight exactly. Applying this to (1.28) gives

$$\begin{aligned} x_t &\approx \frac{\alpha_t}{\alpha_s} x_s + \alpha_t\Big(\int_s^t \frac{\mathrm{d}}{\mathrm{d}u}\Big(\frac{\beta_u}{\alpha_u}\Big)\,\mathrm{d}u\Big)\varepsilon^*(x_s, s) \\ &= \frac{\alpha_t}{\alpha_s} x_s + \alpha_t\Big(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s}\Big)\varepsilon^*(x_s, s). \end{aligned} \tag{1.29}$$

Algorithm 1: DDIM sampler (PF-ODE; one model call per step)

Require: time grid $1 = t_0 > t_1 > \cdots > t_M = 0$; schedule (α_t, β_t); model $\varepsilon_\theta(x, t)$

Initialize $\widetilde{x}_{t_0} \sim p_{\text{init}}$ // e.g. $\mathcal{N}(0, I_d)$

for $i = 0, 1, \ldots, M-1$ **do**

$s \leftarrow t_i$, $t \leftarrow t_{i+1}$

$\widehat{\varepsilon} \leftarrow \varepsilon_\theta(\widetilde{x}_s, s)$

$\widetilde{x}_t \leftarrow \frac{\alpha_t}{\alpha_s}\widetilde{x}_s + \alpha_t\Big(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s}\Big)\widehat{\varepsilon}$

Output: $\widetilde{x}_0$ (approximate sample from p_{data})

Replacing the oracle predictor ε^* with a learned network ε_θ yields the deterministic DDIM update (Song et al., 2022):

$$\widetilde{x}_t = \frac{\alpha_t}{\alpha_s}\widetilde{x}_s + \alpha_t\Big(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s}\Big)\varepsilon_\theta(\widetilde{x}_s, s). \tag{1.30}$$

Thus, DDIM is a one-model-call-per-step exponential Euler method applied to the semilinear PF-ODE (1.24).

Corollary 4 (DDIM update in different parameterizations)

Fix a denoising step $s > t$ and let $\widetilde{x}_s$ be the current state. Define either a noise prediction $\widehat{\varepsilon} \approx \varepsilon^*(\widetilde{x}_s, s)$ or a clean prediction $\widehat{x}_0 \approx \mathbb{E}[x_0 \mid x_s = \widetilde{x}_s]$ and let them be related by the consistency relations

$$\widehat{x}_0 = \frac{\widetilde{x}_s - \beta_s\widehat{\varepsilon}}{\alpha_s}, \qquad \widehat{\varepsilon} = \frac{\widetilde{x}_s - \alpha_s\widehat{x}_0}{\beta_s}. \tag{1.31}$$

Then the deterministic DDIM update $\widetilde{x}_s \mapsto \widetilde{x}_t$ can be written equivalently as

$$\widetilde{x}_t = \frac{\alpha_t}{\alpha_s}\widetilde{x}_s + \alpha_t\Big(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s}\Big)\widehat{\varepsilon} \qquad (\varepsilon\text{-prediction}), \tag{1.32}$$

$$\widetilde{x}_t = \frac{\beta_t}{\beta_s}\widetilde{x}_s + \beta_t\Big(\frac{\alpha_t}{\beta_t} - \frac{\alpha_s}{\beta_s}\Big)\widehat{x}_0 \qquad (x_0\text{-prediction}), \tag{1.33}$$

$$\widetilde{x}_t = \alpha_t\,\widehat{x}_0 + \beta_t\,\widehat{\varepsilon} \qquad (\text{reconstruction / } v\text{-view}). \tag{1.34}$$

Proof of Corollary 4. We start from the ε-prediction DDIM update obtained in (1.30) above via exponential Euler:

$$\widetilde{x}_t = \frac{\alpha_t}{\alpha_s}\widetilde{x}_s + \alpha_t\Big(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s}\Big)\widehat{\varepsilon}, \tag{1.35}$$

which is exactly (1.32).

Assume $\alpha_s \neq 0$ and $\beta_s \neq 0$ (true for any interior step under standard noise schedules; not applying to endpoints that creates singularities).

Step 1: consistency relations. We have $x_t = \alpha_t x_0 + \beta_t \varepsilon$. Taking conditional expectation $\mathbb{E}[\cdot|x_t]$ on both sides gives

$$x_t = \alpha_t \mathbb{E}[X_0|x_t] + \beta_t \mathbb{E}[\varepsilon|x_t] = \alpha_t x_0^*(x_t, t) + \beta_t \varepsilon^*(x_t, t),$$

where $x_0^*(x_t, t) = \mathbb{E}[X_0|x_t]$ and $\varepsilon^*(x_t, t) = \mathbb{E}[\varepsilon|x_t]$ are the oracle predictors. Solving this linear relation for either predictor gives the identities

$$x_0^*(x_t, t) = \frac{x_t - \beta_t \varepsilon^*(x_t, t)}{\alpha_t}, \qquad \varepsilon^*(x_t, t) = \frac{x_t - \alpha_t x_0^*(x_t, t)}{\beta_t}. \tag{1.36}$$

In practice, if one has either a noise prediction $\widehat{\varepsilon}$ or a clean prediction $\widehat{x}_0$, one enforces the same consistency relations by defining the other via (1.31), which is exactly (1.36) with $\widehat{x}_0$ and $\widehat{\varepsilon}$ in place of the oracle predictors. Specifically, if a noise prediction $\widehat{\varepsilon}$ is given, then $\widehat{x}_0$ can be computed using:

$$\widetilde{x}_s = \alpha_s \widehat{x}_0 + \beta_s \widehat{\varepsilon},$$

which gives

$$\widehat{x}_0 = \frac{\widetilde{x}_s - \beta_s \widehat{\varepsilon}}{\alpha_s}.$$

Conversely, if $\widehat{x}_0$ is given, solving the same identity for $\widehat{\varepsilon}$ yields the second relation in (1.31):

$$\widehat{\varepsilon} = \frac{\widetilde{x}_s - \alpha_s \widehat{x}_0}{\beta_s}.$$

Step 2: ε-prediction $\Rightarrow$ reconstruction form. Using the definition of $\widehat{x}_0$ above,

$$\begin{aligned}
\alpha_t \widehat{x}_0 + \beta_t \widehat{\varepsilon} &= \alpha_t \Big(\frac{\widetilde{x}_s - \beta_s \widehat{\varepsilon}}{\alpha_s} \Big) + \beta_t \widehat{\varepsilon} \\
&= \frac{\alpha_t}{\alpha_s} \widetilde{x}_s + \Big(\beta_t - \frac{\alpha_t}{\alpha_s} \beta_s \Big) \widehat{\varepsilon} \\
&= \frac{\alpha_t}{\alpha_s} \widetilde{x}_s + \alpha_t \Big(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s} \Big) \widehat{\varepsilon}.
\end{aligned}$$

The right-hand side matches (1.35), hence $\widetilde{x}_t = \alpha_t \widehat{x}_0 + \beta_t \widehat{\varepsilon}$, which is (1.34).

Step 3: reconstruction $\Rightarrow$ x_0-prediction form. Starting from (1.34) and eliminating $\widehat{\varepsilon}$ using $\widehat{\varepsilon} = (\widetilde{x}_s - \alpha_s \widehat{x}_0)/\beta_s$ (from (1.31)), we obtain

$$\begin{aligned}
\widetilde{x}_t &= \alpha_t \widehat{x}_0 + \beta_t \Big(\frac{\widetilde{x}_s - \alpha_s \widehat{x}_0}{\beta_s} \Big) \\
&= \frac{\beta_t}{\beta_s} \widetilde{x}_s + \Big(\alpha_t - \frac{\beta_t}{\beta_s} \alpha_s \Big) \widehat{x}_0 \\
&= \frac{\beta_t}{\beta_s} \widetilde{x}_s + \beta_t \Big(\frac{\alpha_t}{\beta_t} - \frac{\alpha_s}{\beta_s} \Big) \widehat{x}_0,
\end{aligned}$$

which is (1.33).

Therefore (1.32), (1.33), and (1.34) are equivalent representations of the same deterministic DDIM step, with the predictors related by (1.31). □

Remark 5 (Euler vs. exponential Euler)

DDIM is an *exponential Euler* step for the semilinear PF-ODE in ε-, x_0-, or score-prediction form: it integrates the linear "shrink/scale" factor $E(s \to t) = \alpha_t/\alpha_s$ *exactly* and only approximates the learned nonlinear term. A plain Euler step would also discretize this linear multiplier, which typically degrades stability and accuracy when taking large denoising steps.

In contrast, in a v-prediction parameterization one can often rewrite the PF-ODE so that the drift has no isolated linear part, orequivalently, $E(s \to t) = 1$; see details below. In that case, exponential Euler reduces to ordinary Euler, and DDIM scoincides with a plain Euler update on the corresponding v-ODE.

Why v-prediction makes exponential Euler look like Euler? In ε-prediction, the probability-flow ODE (PF-ODE) is naturally written in a *semilinear* form

$$\dot{x}_t = \underbrace{\frac{\dot{\alpha}_t}{\alpha_t} x_t}_{\text{isolated linear part}} + \underbrace{\Big(\dot{\beta}_t - \frac{\dot{\alpha}_t}{\alpha_t}\beta_t\Big)\, \varepsilon^*(x_t, t)}_{\text{learned/nonlinear part}}, \tag{1.37}$$

so the linear propagator is

$$E(s \to t) = \exp\Big(\int_s^t \frac{\dot{\alpha}_u}{\alpha_u}\, \mathrm{d}u\Big) = \frac{\alpha_t}{\alpha_s}.$$

Freezing $\varepsilon^*(x_u, u) \approx \varepsilon^*(x_s, s)$ and integrating the scalar kernel exactly produces the DDIM *exponential-Euler* update, when using $\varepsilon_\theta(x_s, s)$ in place of $\varepsilon^*(x_s, s)$.

By contrast, in the *reconstruction / v-view* one keeps track of the clean/noise decomposition

$$x \approx \alpha_t\, x_0(x, t) + \beta_t\, \varepsilon_\theta(x, t), \qquad \text{where} \qquad x_0(x, t) \triangleq \frac{x - \beta_t \varepsilon_\theta(x, t)}{\alpha_t}.$$

Using the identity $x_t = \alpha_t x_0^*(x_t, t) + \beta_t \varepsilon^*(x_t, t)$, one can rewrite (1.37) as

$$\dot{x}_t = \dot{\alpha}_t\, x_0^*(x_t, t) + \dot{\beta}_t\, \varepsilon^*(x_t, t). \tag{1.38}$$

This representation has *no standalone term proportional to x_t*. Equivalently, if we split the ODE as $\dot{x}_t = 0 \cdot x_t + F(t, x_t)$, then the linear part is zero and therefore the propagator is the identity:

$$E(s \to t) = \exp\Big(\int_s^t 0\, \mathrm{d}u\Big) = 1.$$

Hence, exponential Euler collapses to an Euler-type *freeze the state dependence* rule. Indeed, freezing $x_0^*(x_u, u)$ as $x_0^*(x_s, s) \approx x_0^*(x_u, u)$ and $\varepsilon^*(x_u, u)$ as $\varepsilon^*(x_s, s) \approx \varepsilon^*(x_u, u)$ in (1.38) gives

$$\begin{aligned} x_t &\approx x_s + \int_s^t \dot{\alpha}_u\, \mathrm{d}u\; x_0^*(x_s, s) + \int_s^t \dot{\beta}_u\, \mathrm{d}u\; \varepsilon^*(x_s, s) \\ &= x_s + (\alpha_t - \alpha_s) x_0^*(x_s, s) + (\beta_t - \beta_s)\varepsilon^*(x_s, s) \\ &= \alpha_t\, x_0^*(x_s, s) + \beta_t\, \varepsilon^*(x_s, s) + \big(x_s - \alpha_s x_0^*(x_s, s) - \beta_s \varepsilon^*(x_s, s)\big). \end{aligned}$$

Due to the consistency relation $x_s = \alpha_s x_0^*(x_s, s) + \beta_s \varepsilon^*(x_s, s)$, the residual term vanishes and we obtain

$$x_t = \alpha_t x_0^*(x_s, s) + \beta_t \varepsilon^*(x_s, s),$$

which is exactly the DDIM update in reconstruction form, if we replace the oracle predictors with the learned ones.

A common special case: variance-preserving schedules with $\alpha_t^2+\beta_t^2 \equiv 1$. If $\alpha_t^2+\beta_t^2 \equiv 1$, then the PF-ODE in v-prediction form can be rewritten as

$$\begin{aligned}\dot{x}_t &= \gamma_t x_t + \left(\dot{\beta}_t - \frac{\dot{\alpha}_t}{\alpha_t}\beta_t\right)\varepsilon^*(x_t,t) \\ &= \dot{\alpha}_t x_0^*(x_t,t) + \dot{\beta}_t \varepsilon^*(x_t,t) = v^*(x_t,t).\end{aligned}$$

This makes the *no isolated linear part* (and thus $E(s\to t)=1$) completely explicit: the drift is exactly equal to v^*.

1.7 DEIS: multistep exponential integrators

DDIM is a one-step (first-order) exponential integrator. The Diffusion Exponential Integrator Sampler (DEIS) by Zhang and Chen, 2023 improves accuracy by approximating the integrand in (1.28) with a degree-$(K-1)$ polynomial fitted to the last K model evaluations, then integrating this polynomial *exactly* against the known kernel weight.

Lagrange interpolation. Fix a step index $i \geq K$ and let $\mathcal{T}_i \triangleq \{t_{i-1}, t_{i-2}, \ldots, t_{i-K}\}$. Along the numerical trajectory we store $\widehat{\varepsilon}_{t_j} \triangleq \varepsilon_\theta(\widetilde{x}_{t_j}, t_j)$. Define Lagrange basis polynomials on these nodes,

$$\ell_{i,j}(u) \triangleq \prod_{\substack{m=0 \\ m\neq j}}^{K-1} \frac{u - t_{i-1-m}}{t_{i-1-j} - t_{i-1-m}}, \qquad j = 0, 1, \ldots, K-1, \tag{1.39}$$

such that $\ell_{i,j}(t_{i-1-k}) = \delta_{j,k}$ for $k = 0, \ldots, K-1$, and the interpolant

$$\widehat{\varepsilon}_i(u) \triangleq \sum_{j=0}^{K-1} \ell_{i,j}(u)\, \widehat{\varepsilon}_{t_{i-1-j}} \qquad \text{hence} \qquad \widehat{\varepsilon}_i(t_{i-1-j}) = \widehat{\varepsilon}_{t_{i-1-j}}. \tag{1.40}$$

Specifically, $u \mapsto \widehat{\varepsilon}_i(u)$ is the unique polynomial of degree at most $K-1$ on $[t_i, t_{i-1}]$ that matches the stored values at past times:

$$\widehat{\varepsilon}_i(t_{i-1}) = \widehat{\varepsilon}_{t_{i-1}},\ \widehat{\varepsilon}_i(t_{i-2}) = \widehat{\varepsilon}_{t_{i-2}},\ \ldots,\ \widehat{\varepsilon}_i(t_{i-K}) = \widehat{\varepsilon}_{t_{i-K}}.$$

The Lagrange basis polynomials thus act as *selector polynomials*: the interpolant is a weighted sum of stored values with weights that are polynomials in u.

DEIS update. DEIS replaces $\varepsilon_\theta(x_u, u)$ by $\widehat{\varepsilon}_i(u)$ in (1.28):

$$\widetilde{x}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\widetilde{x}_{t_{i-1}} + \sum_{j=0}^{K-1} C_{i,j}\, \widehat{\varepsilon}_{t_{i-1-j}}, \tag{1.41}$$

with coefficients

$$C_{i,j} \triangleq \alpha_{t_i} \int_{t_{i-1}}^{t_i} \frac{\mathrm{d}}{\mathrm{d}u}\left(\frac{\beta_u}{\alpha_u}\right) \ell_{i,j}(u)\, \mathrm{d}u. \tag{1.42}$$

The $C_{i,j}$ depend only on the grid and schedule, so they can be precomputed.

Algorithm 2: DEIS-K: a multistep exponential integrator sampler.

Require: time grid $1 = t_0 > t_1 > \cdots > t_M = 0$; history length K; schedule (α_t, β_t); noise model ε_θ.

Initialize $\widetilde{x}_{t_0} \sim p_{\text{init}}$

Warm up for $K-1$ steps using DDIM to obtain $\widetilde{x}_{t_1}, \ldots, \widetilde{x}_{t_{K-1}}$

Store $\widehat{\varepsilon}_{t_j} \leftarrow \varepsilon_\theta(\widetilde{x}_{t_j}, t_j)$ for $j = 0, \ldots, K-1$

for $i = K, \ldots, M$ **do** `// main multistep loop`

  Compute/look up DEIS coefficients $\{C_{i,j}\}_{j=0}^{K-1}$ from (1.42)

  `// Integrate the Lagrange interpolant against the known kernel weight`

  $\widetilde{x}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \widetilde{x}_{t_{i-1}} + \sum_{j=0}^{K-1} C_{i,j}\, \widehat{\varepsilon}_{t_{i-1-j}}$

  $\widehat{\varepsilon}_{t_i} \leftarrow \varepsilon_\theta(\widetilde{x}_{t_i}, t_i)$

DDIM as DEIS with $K = 1$. When $K = 1$, the Lagrange basis satisfies $\ell_{i,0} \equiv 1$, so (1.41) reduces to DDIM. The coefficient simplifies as:

$$C_{i,0} = \alpha_{t_i} \int_{t_{i-1}}^{t_i} \frac{\mathrm{d}}{\mathrm{d}u}\Big(\frac{\beta_u}{\alpha_u}\Big)\, \mathrm{d}u = \alpha_{t_i} \left(\frac{\beta_{t_i}}{\alpha_{t_i}} - \frac{\beta_{t_{i-1}}}{\alpha_{t_{i-1}}} \right).$$

Why DEIS is efficient. Because DEIS reuses the past $K-1$ stored values $\{\widehat{\varepsilon}_{t_{i-1}}, \ldots, \widehat{\varepsilon}_{t_{i-K}}\}$, each step after warm-up requires only *one* new model evaluation $\widehat{\varepsilon}_{t_i}$. This is the hallmark of a *multistep* method. In contrast, single-step k-th order methods typically require multiple model calls per step (e.g., multiple stages in a Runge–Kutta method), whereas DEIS achieves high-order accuracy with just one new evaluation per step (after a short warm-up). Thus the average cost per step remains $m \approx 1$, making it highly efficient.

1.8 The DPM-Solver family via log-SNR reparameterization

The core idea of the DPM-Solver family (Lu et al., 2022, 2023; Zheng et al., 2023) is to reparameterize time by the *half-log signal-to-noise ratio (SNR)*.

Half-log SNR. For (1.8), define

$$\lambda_t \triangleq \tfrac{1}{2} \log \frac{\alpha_t^2}{\beta_t^2} = \log \frac{\alpha_t}{\beta_t}. \tag{1.43}$$

Then $\mathrm{SNR}(t) \triangleq \alpha_t^2/\beta_t^2 = e^{2\lambda_t}$ and $\beta_t/\alpha_t = e^{-\lambda_t}$. Typical schedules make $t \mapsto \lambda_t$ strictly decreasing.

Observation 1 (Why a uniform grid in λ helps?)

A step size $h \triangleq \lambda_t - \lambda_s$ induces a *multiplicative* SNR change:

$$\frac{\mathrm{SNR}(t)}{\mathrm{SNR}(s)} = \exp\big(2(\lambda_t - \lambda_s)\big) = e^{2h}. \tag{1.44}$$

Hence, equal increments in λ correspond to equal *relative* changes in SNR. Because denoising difficulty is driven more by SNR than by raw time t, a uniform λ-grid typically yields more balanced steps. By contrast, a uniform grid in t can produce uneven SNR changes, over-resolving some regions and under-resolving others, which leads to redundant steps in some intervals and insufficient resolution in others.

1.8.1 DPM-Solver

Exact integral equation in λ-time. Let $t_\lambda(\cdot)$ be the inverse map and define $\widehat{x}_\lambda \triangleq x_{t_\lambda(\lambda)}$ and $\widehat{\varepsilon}_\theta(\widehat{x}_\lambda, \lambda) \triangleq \varepsilon_\theta(\widehat{x}_\lambda, t_\lambda(\lambda))$. Then we have the following exact integral representation in λ-time.

Proposition 5 (Exact integral equation in λ-time)

Assume for simplicity that $\varepsilon_\theta = \varepsilon^*$ so that the same PF-ODE holds for the learned model. Then, in λ-time, the following holds:

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\Big(\frac{\widehat{x}_\lambda}{\widehat{\alpha}_\lambda}\Big) = -e^{-\lambda}\,\widehat{\varepsilon}_\theta(\widehat{x}_\lambda, \lambda), \qquad \widehat{\alpha}_\lambda \triangleq \alpha_{t_\lambda(\lambda)}, \tag{1.45}$$

$$x_t = \frac{\alpha_t}{\alpha_s} x_s - \beta_t \int_{\lambda_s}^{\lambda_t} e^{\lambda_t - \lambda}\,\widehat{\varepsilon}_\theta(\widehat{x}_\lambda, \lambda)\,\mathrm{d}\lambda, \qquad h \triangleq \lambda_t - \lambda_s \geq 0. \tag{1.46}$$

Approximating the above exponentially weighted integral better with a small number of model evaluations yields fast solvers.

Proof of Proposition 5. Starting from the semilinear form (1.19):

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = \gamma_t x_t + b_t\,\varepsilon_\theta(x_t, t), \qquad b_t = \alpha_t\,\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\beta_t}{\alpha_t}\Big),$$

define the scaled state $y_t \triangleq x_t/\alpha_t$. Then

$$\frac{\mathrm{d}y_t}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\beta_t}{\alpha_t}\Big)\,\varepsilon_\theta(x_t, t). \tag{1.47}$$

Using $\beta_t/\alpha_t = e^{-\lambda_t}$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\beta_t}{\alpha_t}\Big)\,\mathrm{d}t = \mathrm{d}\big(e^{-\lambda_t}\big) = -e^{-\lambda}\,\mathrm{d}\lambda.$$

Hence

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\Big(\frac{\widehat{x}_\lambda}{\widehat{\alpha}_\lambda}\Big) = -e^{-\lambda}\,\widehat{\varepsilon}_\theta(\widehat{x}_\lambda, \lambda), \qquad \widehat{\alpha}_\lambda \triangleq \alpha_{t_\lambda(\lambda)}.$$

Integrating from λ_s to λ_t gives (1.46). □

Taylor expansions and ϕ-functions. Write the exact integral in the shifted variable $r = \lambda - \lambda_s \in [0, h]$, where $h \triangleq \lambda_t - \lambda_s \geq 0$:

$$x_t = \frac{\alpha_t}{\alpha_s} x_s - \beta_t \int_0^h e^{h-r}\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s+r}, \lambda_s + r)\, \mathrm{d}r. \tag{1.48}$$

If we expand the integrand in a Taylor series in r,

$$\widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s+r}, \lambda_s + r) \approx \sum_{k=0}^{p-1} \frac{r^k}{k!}\, \partial_\lambda^k\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s),$$

then (1.48) yields coefficients that are *explicit* functions of h:

$$\int_0^h e^{h-r}\, \frac{r^k}{k!}\, \mathrm{d}r \;=\; \phi_{k+1}(h), \qquad \phi_k(h) \triangleq \int_0^h e^{h-r}\, \frac{r^{k-1}}{(k-1)!}\, \mathrm{d}r. \tag{1.49}$$

In particular,

$$\phi_1(h) = e^h - 1, \qquad \phi_2(h) = e^h - 1 - h, \qquad \phi_3(h) = e^h - 1 - h - \tfrac{1}{2}h^2. \tag{1.50}$$

DPM-Solver methods can be viewed as using a small number of model evaluations to approximate these Taylor terms (without explicitly computing derivatives), resulting in accurate updates even for relatively large h, hence few steps.

DPM-Solver-1 = DDIM. For any $(\widetilde{x}_s, \lambda_s)$ and $(\widetilde{x}_t, \lambda_t)$, freezing the integrand in (1.46) at $(\widetilde{x}_s, \lambda_s)$ gives the following update:

$$\begin{aligned} \widetilde{x}_t &\leftarrow \frac{\alpha_t}{\alpha_s}\widetilde{x}_s - \beta_t\big(e^h - 1\big)\, \varepsilon_\theta(\widetilde{x}_s, s) \\ &= \frac{\alpha_t}{\alpha_s}\widetilde{x}_s + \alpha_t \left(\frac{\beta_t}{\alpha_t} - \frac{\beta_s}{\alpha_s}\right) \varepsilon_\theta(\widetilde{x}_s, s), \qquad\qquad \frac{\beta}{\alpha} = \exp(-\lambda), \end{aligned} \tag{1.51}$$

which is exactly the DDIM update (1.30).

DPM-Solver-p with $p \geq 2$ formally involves derivatives $\partial_\lambda^k \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s)$. Because explicitly computing high-order derivatives is expensive, Lu et al. (2022) approximates these terms efficiently using finite-difference schemes (including midpoint formulas). We illustrate this for $p = 2$, which leads to DPM-Solver-2.

DPM-Solver-2. Let $\lambda_{\mathrm{mid}} \triangleq \lambda_s + \frac{1}{2}h$ and $t_{\mathrm{mid}} \triangleq t_\lambda(\lambda_{\mathrm{mid}})$. We first compute an intermediate state x_{mid} using the initial noise prediction, then evaluate the model at the midpoint and use that evaluation to correct the full step:

$$\widetilde{x}_{\mathrm{mid}} \leftarrow \frac{\alpha_{t_{\mathrm{mid}}}}{\alpha_s}\, \widetilde{x}_s - \beta_{t_{\mathrm{mid}}}\big(e^{h/2} - 1\big)\, \varepsilon_\theta(\widetilde{x}_s, s), \quad \text{(forecast to midpoint)} \tag{1.52}$$

$$\widetilde{x}_t \leftarrow \frac{\alpha_t}{\alpha_s}\, \widetilde{x}_s - \beta_t\big(e^h - 1\big)\, \varepsilon_\theta(\widetilde{x}_{\mathrm{mid}}, t_{\mathrm{mid}}). \quad \text{(correction using midpoint)} \tag{1.53}$$

This is precisely the exponential midpoint (Heun-type) correction underlying DPM-Solver-2. Algorithm 3 summarizes the full method.

Derivation of DPM-Solver-2 via Taylor expansions. The midpoint method is second-order because it matches the first two terms of the Taylor expansion of the integrand in (1.48) with a second-order error. We make this explicit below.

Starting from (1.48), expand the noise predictor:

$$\widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s+r}, \lambda_s + r) = \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) + r\, \partial_\lambda \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) + O(r^2). \tag{1.54}$$

Substituting into the integral and integrating term by term yields

$$x_t = \frac{\alpha_t}{\alpha_s} x_s - \beta_t \Big[\phi_1(h)\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) + \phi_2(h)\, \partial_\lambda \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) + O(h^3) \Big]. \tag{1.55}$$

To build a practical two-stage solver that *explicitly estimates* the derivative $\partial_\lambda \widehat{\varepsilon}_\theta$, use the midpoint finite-difference formula:

$$\partial_\lambda \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) \;=\; \frac{\widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) - \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s)}{h/2} + O(h). \tag{1.56}$$

Substituting this approximation into (1.55) gives

$$\begin{aligned}
x_t &= \frac{\alpha_t}{\alpha_s} x_s - \beta_t \phi_1(h)\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) - \beta_t \phi_2(h) \left(\frac{\widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) - \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s)}{h/2} + O(h) \right) + O(h^3) \\
&= \frac{\alpha_t}{\alpha_s} x_s - \beta_t \left(\phi_1(h) - \frac{\phi_2(h)}{h/2} \right) \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) - \beta_t \frac{\phi_2(h)}{h/2} \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) + O(h^3) \\
&= \frac{\alpha_t}{\alpha_s} x_s - \beta_t \phi_1(h)\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) \\
&\qquad - \beta_t \left(\phi_1(h) - \frac{\phi_2(h)}{h/2} \right) \Big(\widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) - \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) \Big) + O(h^3) \\
&= \frac{\alpha_t}{\alpha_s} x_s - \beta_t \phi_1(h)\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) + O(h^3) \\
&\approx \frac{\alpha_t}{\alpha_s} x_s - \beta_t \big(e^h - 1 \big)\, \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}),
\end{aligned}$$

where the penultimate step uses

$$\phi_1(h) - \frac{\phi_2(h)}{h/2} = O(h^2) \qquad \text{and} \qquad \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_s}, \lambda_s) - \widehat{\varepsilon}_\theta(\widehat{x}_{\lambda_{\mathrm{mid}}}, \lambda_{\mathrm{mid}}) = O(h),$$

so the extra term is $O(h^3)$. This is exactly the DPM-Solver-2 update. The one-step truncation error is $O(h^3)$, which yields second-order global accuracy. □

Remark 6 (Classical ODE methods hiding underneath)

The representation (1.46) is an exponentially weighted integral equation in λ. DPM-Solver-1 corresponds to the first-order exponential Euler; DPM-Solver-2 is a second-order exponential midpoint / Heun-type correction. Higher-order variants use additional intermediate evaluations or multistep differences to approximate the λ-integral more accurately.

Algorithm 3: DPM-Solver-2 (midpoint; ε-prediction)

Require: time grid $t_0 > t_1 > \cdots > t_M$ (e.g. $t_0 = 1, t_M = 0$); $\lambda_t = \log(\alpha_t/\beta_t)$; inverse $t_\lambda(\cdot)$; model ε_θ.

Initialize $\widetilde{x}_{t_0} \sim p_{\text{init}}$

for $i = 1, \dots, M$ **do** `// steps in decreasing t / increasing λ`

- $s \leftarrow t_{i-1}$, $t \leftarrow t_i$, $h \leftarrow \lambda_t - \lambda_s$
- $\lambda_{\text{mid}} \leftarrow \lambda_s + \frac{1}{2}h$, $t_{\text{mid}} \leftarrow t_\lambda(\lambda_{\text{mid}})$
- $\widehat{\varepsilon}_s \leftarrow \varepsilon_\theta(\widetilde{x}_s, s)$
- `// forecast to the midpoint (1st-order / DDIM)`
- $\widetilde{x}_{\text{mid}} \leftarrow \frac{\alpha_{t_{\text{mid}}}}{\alpha_s}\widetilde{x}_s - \beta_{t_{\text{mid}}}\big(e^{h/2} - 1\big)\widehat{\varepsilon}_s$
- $\widehat{\varepsilon}_{\text{mid}} \leftarrow \varepsilon_\theta(\widetilde{x}_{\text{mid}}, t_{\text{mid}})$
- `// 2nd-order correction using the midpoint evaluation`
- $\widetilde{x}_t \leftarrow \frac{\alpha_t}{\alpha_s}\widetilde{x}_s - \beta_t\big(e^h - 1\big)\widehat{\varepsilon}_{\text{mid}}$

1.8.2 DPM-Solver++: stability via x_0-prediction

Classifier-free guidance can greatly increase the magnitude of the model output. In ε-prediction this may lead to unstable updates when using large steps. DPM-Solver++ mitigates this by switching to *x_0-prediction* and (optionally) clipping/thresholding the predicted $\widehat{x}_0$.

Relating x_0- and ε-prediction. Assume that $x_{0,\theta}(x_t, t) = x_0^*(x_t, t)$ and $\varepsilon_\theta(x_t, t) = \varepsilon^*(x_t, t)$. From Section (1.4), the two parameterizations are related by

$$\varepsilon_\theta(x_t, t) = \frac{x_t - \alpha_t\, x_{0,\theta}(x_t, t)}{\beta_t}, \qquad x_{0,\theta}(x_t, t) = \frac{x_t - \beta_t\, \varepsilon_\theta(x_t, t)}{\alpha_t}. \tag{1.57}$$

Thus one can rewrite the same PF-ODE either in ε-prediction form or in x_0-prediction form.

A key simplification in λ-time. As above, view the trajectory in λ-time via $\widehat{x}_\lambda \triangleq x_{t_\lambda(\lambda)}$. Define the corresponding clean predictor $\widehat{x}_{0,\theta}(\widehat{x}_\lambda, \lambda) \triangleq x_{0,\theta}(\widehat{x}_\lambda, t_\lambda(\lambda))$. Starting from (1.45) and using $e^{-\lambda} = \beta/\alpha$ together with (1.57) gives

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\Big(\frac{\widehat{x}_\lambda}{\widehat{\alpha}_\lambda}\Big) = -\Big(\frac{\widehat{x}_\lambda}{\widehat{\alpha}_\lambda} - \widehat{x}_{0,\theta}(\widehat{x}_\lambda, \lambda)\Big). \tag{1.58}$$

Let $\widehat{y}_\lambda \triangleq \widehat{x}_\lambda/\widehat{\alpha}_\lambda$. Then

$$\frac{\mathrm{d}\widehat{y}_\lambda}{\mathrm{d}\lambda} = -\widehat{y}_\lambda + \widehat{x}_{0,\theta}(\widehat{x}_\lambda, \lambda).$$

This is an exponential relaxation toward the clean prediction on a unit λ-timescale. The key point is that, after switching to x_0-prediction, the kernel becomes decaying rather than growing.

Proposition 6 (Exact integral form in x_0-prediction)

Let $s > t$ and $h \triangleq \lambda_t - \lambda_s \geq 0$. The PF-ODE written as (1.58) implies

$$x_t = \frac{\beta_t}{\beta_s}x_s + \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-(\lambda_t - \lambda)}\, \widehat{x}_{0,\theta}(\widehat{x}_\lambda, \lambda)\, \mathrm{d}\lambda. \tag{1.59}$$

Proof. Equation (1.58) is linear in $\widehat{y}_\lambda$, so solving it with an integrating factor yields

$$\widehat{y}_{\lambda_t} = e^{-h}\widehat{y}_{\lambda_s} + \int_{\lambda_s}^{\lambda_t} e^{-(\lambda_t-\lambda)}\widehat{x}_{0,\theta}(\widehat{x}_\lambda, \lambda)\,\mathrm{d}\lambda.$$

Multiplying by α_t and using

$$\frac{\alpha_t e^{-h}}{\alpha_s} = \frac{\alpha_t}{\alpha_s}e^{-(\lambda_t-\lambda_s)} = \frac{\alpha_t}{\alpha_s}\frac{\beta_t/\alpha_t}{\beta_s/\alpha_s} = \frac{\beta_t}{\beta_s}$$

gives (1.59). □

First-order DPM-Solver++ update. Freezing the integrand in (1.59) at the start of the step gives the one-step update

$$x_t \approx \frac{\beta_t}{\beta_s}x_s + \alpha_t\big(1-e^{-h}\big)\,x_{0,\theta}(x_s, s), \qquad h = \lambda_t - \lambda_s \geq 0. \tag{1.60}$$

This is the exact analogue of DPM-Solver-1, but written in x_0-prediction form. Now the coefficients satisfy $\beta_t/\beta_s \leq 1$ and $1-e^{-h} \in (0,1)$, so they remain bounded even for coarse steps.

Why this helps under strong guidance. Compare the first-order updates in ε-prediction versus x_0-prediction:

$$\text{DDIM / DPM-Solver-1 } (\varepsilon\text{-pred}): \qquad x_t \approx \frac{\alpha_t}{\alpha_s}x_s - \beta_t\big(e^h-1\big)\,\varepsilon_\theta(x_s, s), \tag{1.61}$$

$$\text{DPM-Solver++-1 } (x_0\text{-pred}): \qquad x_t \approx \frac{\beta_t}{\beta_s}x_s + \alpha_t\big(1-e^{-h}\big)\,x_{0,\theta}(x_s, s). \tag{1.62}$$

When CFG amplifies the model output, the ε-update can be destabilized by the growing factor e^h-1. By contrast, the x_0-update uses bounded coefficients and also permits clipping or dynamic thresholding directly in data space, where these operations are more natural.

DPM-Solver++-2 (midpoint; 2nd order). Let $\lambda_{\mathrm{mid}} = \lambda_s + \frac{1}{2}h$ and $t_{\mathrm{mid}} = t_\lambda(\lambda_{\mathrm{mid}})$. A second-order midpoint variant mirrors DPM-Solver-2: first take a first-order forecast to the midpoint, then evaluate the clean predictor there, and finally use that midpoint value to update the full step:

$$x_{\mathrm{mid}} \leftarrow \frac{\beta_{t_{\mathrm{mid}}}}{\beta_s}x_s + \alpha_{t_{\mathrm{mid}}}\big(1-e^{-h/2}\big)\,x_{0,\theta}(x_s, s), \tag{1.63}$$

$$x_t \leftarrow \frac{\beta_t}{\beta_s}x_s + \alpha_t\big(1-e^{-h}\big)\,x_{0,\theta}(x_{\mathrm{mid}}, t_{\mathrm{mid}}). \tag{1.64}$$

In practice one often applies dynamic thresholding or simple clipping to the midpoint prediction $x_{0,\theta}(x_{\mathrm{mid}}, t_{\mathrm{mid}})$ before using it.

Algorithm 4: DPM-Solver++-2 (midpoint; x_0-prediction)

Require: time grid $t_0 > t_1 > \cdots > t_M$; log-SNR $\lambda_t = \log(\alpha_t/\beta_t)$ and inverse $t_\lambda(\cdot)$; schedule (α_t, β_t); x_0-prediction model $\widehat{x}_{0,\theta}$.

Initialize $\widetilde{x}_{t_0} \sim p_{\text{init}}$

$\widehat{x}_{0,s} \leftarrow \widehat{x}_{0,\theta}(\widetilde{x}_{t_0}, t_0)$

for $i = 1, \ldots, M$ **do** `// main loop`

- $s \leftarrow t_{i-1}$, $t \leftarrow t_i$, $h \leftarrow \lambda_t - \lambda_s$
- $\lambda_{\text{mid}} \leftarrow \lambda_s + \frac{1}{2}h$, $t_{\text{mid}} \leftarrow t_\lambda(\lambda_{\text{mid}})$
- `// 1st-order forecast to the midpoint using cached` $\widehat{x}_{0,s}$
- $\widetilde{x}_{\text{mid}} \leftarrow \frac{\beta_{t_{\text{mid}}}}{\beta_s}\widetilde{x}_s + \alpha_{t_{\text{mid}}}\big(1 - e^{-h/2}\big)\widehat{x}_{0,s}$
- $\widehat{x}_{0,\text{mid}} \leftarrow \widehat{x}_{0,\theta}(\widetilde{x}_{\text{mid}}, t_{\text{mid}})$
- `// (optional) stabilize: threshold/clip` $\widehat{x}_{0,\text{mid}}$
- `// 2nd-order update using the midpoint evaluation`
- $\widetilde{x}_t \leftarrow \frac{\beta_t}{\beta_s}\widetilde{x}_s + \alpha_t\big(1 - e^{-h}\big)\widehat{x}_{0,\text{mid}}$
- **if** $i < M$ **then** `// cache for next step; may skip at` $i = M$
  - $\widehat{x}_{0,s} \leftarrow \widehat{x}_{0,\theta}(\widetilde{x}_t, t)$

Multistep variants. DPM-Solver++ also admits multistep versions (2M, 3M, ...) that reuse a short history so that, after warm-up, each step needs only one new model evaluation, similar in spirit to DEIS.

1.8.3 DPM-Solver-v3

DPM-Solver-v3 (Zheng et al., 2023) makes explicit a design choice that was implicit in the previous sections: before discretizing the PF-ODE, one must decide which prediction target to use (ε, x_0, v, score, or hybrids) and which one-step coefficients to attach to that parameterization. Rather than fixing thßßßßßese choices by hand, DPM-Solver-v3 chooses them by minimizing a proxy for the local truncation error. The main idea is simple: treat sampler design itself as a small regression problem.

A simple residual-minimization view. Write the PF-ODE in semilinear form

$$\dot{x}_t = \gamma_t x_t + N_\theta(x_t, t), \tag{1.65}$$

where γ_t is schedule-dependent and N_θ depends on the chosen parameterization. For a step $s \to t$, the exact variation-of-constants identity is

$$x_t = E(s \to t)\, x_s + \int_s^t E(u \to t)\, N_\theta(x_u, u)\, \mathrm{d}u. \tag{1.66}$$

Thus the design problem is really: how should we approximate the integral term in (1.66)?

[More details - it might be hard to understand for students.]

A broad class of one-step solvers replaces that integral by a weighted combination of a few

model evaluations:

$$\widetilde{x}_t(\beta) \triangleq E(s \to t)\, x_s + \sum_{m=1}^{q} \beta_m\, \Phi_m(s,t)\, N_\theta(\bar{x}_{s_m}, s_m), \tag{1.67}$$

where $\{s_m\} \subset [t,s]$ are evaluation times and $\Phi_m(s,t)$ are known scalar kernel factors (ϕ-functions, integrals of E, etc.), while $\bar{x}_{s_m}$ are stage states taken from a cheap surrogate trajectory such as a first-order forecast. So the unknowns are the coefficients $\beta = (\beta_1, \ldots, \beta_q)$, and the question is how to choose them well.

To answer that, DPM-Solver-v3 first fixes a surrogate trajectory $\bar{x}_u$ and measures how well a candidate coefficient vector β reproduces the integral along that surrogate path. This gives the residual

$$\mathcal{R}(\beta) \triangleq \int_s^t E(u \to t)\, N_\theta(\bar{x}_u, u)\, \mathrm{d}u \;\; - \sum_{m=1}^{q} \beta_m\, \Phi_m(s,t)\, N_\theta(\bar{x}_{s_m}, s_m). \tag{1.68}$$

The first term is the surrogate version of the exact integral, and the second term is the one-step approximation. The coefficients are then chosen by a least-squares criterion

$$\beta^* \in \arg\min_{\beta}\; \mathbb{E}\big[\|\mathcal{R}(\beta)\|_2^2\big], \tag{1.69}$$

and one repeats this procedure across candidate parameterizations N_θ. The final choice is the target and coefficient vector whose optimized proxy error is smallest.

In this sense, DPM-Solver-v3 is a meta-design principle rather than a single hand-derived update rule: it uses a cheap surrogate trajectory to score candidate parameterizations and quadrature weights, then keeps the combination predicted to have the smallest local error.

1.9 A numerical-analysis viewpoint: What are these solvers?

Many diffusion samplers are classical ODE methods applied to either the PF-ODE itself or an exponential-integrator reformulation (often after a time reparameterization). The table below lists common correspondences.

View / parameterization	**Classical method**	**Sampler name (typical)**
v-prediction	Euler	DDIM (often coincides)
ε/ x_0/ score prediction	exponential Euler	DDIM
log-SNR (λ) + ε-pred	exponential midpoint / Heun	DPM-Solver-2
semilinear, multistep	Adams–Bashforth-type EI	DEIS-K
log-SNR + tailored Taylor / ϕ-weights	specialized EI / RK	DPM-Solver family

1.10 Closing remarks

Training-free solvers can drastically reduce sampling cost:

- **DDIM**: first-order exponential Euler for the PF-ODE;
- **DEIS**: multistep exponential integrators (higher order with NFE near 1 per step);

- **DPM-Solver family**: log-SNR reparameterization and tailored exponential-integrator updates;

These methods make diffusion models far more practical, but they remain iterative. An important next direction is to *learn* a fast solver (distillation / one-step generative models).

References

Karatzas, I. and Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus*, volume 113 of *Graduate Texts in Mathematics*. Springer, New York, 2 edition.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022). DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2023). DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Universitext. Springer, Berlin, Heidelberg, 6 edition.

Song, J., Meng, C., and Ermon, S. (2022). Denoising Diffusion Implicit Models.

Zhang, Q. and Chen, Y. (2023). Fast Sampling of Diffusion Models with Exponential Integrator.

Zheng, K., Lu, C., Chen, J., and Zhu, J. (2023). DPM-Solver-v3: Improved Diffusion ODE Solver with Empirical Model Statistics.