

Ch.9 Sophisticated Solvers for Fast Sampling

The Principles of Diffusion Models

Jikwang Kim

Seoul National University

August 24, 2026

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models
- 3 DEIS : Diffusion Exponential Integrator Sampler
- 4 DPM-Solver : Diffusion Probabilistic Model Solver
- 5 DPM-Solver++
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models
- 3 DEIS : Diffusion Exponential Integrator Sampler
- 4 DPM-Solver : Diffusion Probabilistic Model Solver
- 5 DPM-Solver++
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps

For the forward stochastic process $d\mathbf{x}(t) = f(\mathbf{x}(t), t) dt + g(t) dw(t)$, we know that the sampling procedure follows the PF-ODE:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}}\log p_t(\mathbf{x}(t)),$$

and we target the score $\nabla_{\mathbf{x}}\log p_t(\mathbf{x}(t)) \approx s_{\phi\times}(\mathbf{x}, t)$.

Let $\Psi_{s \rightarrow t}$ be a flow map of the oracle PF-ODE, then the exact solution of that is

$$\begin{aligned}\Psi_{T \rightarrow 0}(x(T)) &= x(T) + \int_T^0 \left[f(\mathbf{x}(\tau), \tau) - \frac{1}{2} g^2(\tau) \nabla_{\mathbf{x}} \log p_{\tau}(\mathbf{x}(\tau)) \right] d\tau \\ &\approx x(T) + \int_T^0 \left[f(\mathbf{x}(\tau), \tau) - \frac{1}{2} g^2(\tau) s_{\phi \times}(\mathbf{x}(\tau), \tau) \right] d\tau \\ &=: \tilde{\Psi}_{T \rightarrow 0}(\mathbf{x}(T)).\end{aligned}$$

In practice, for $s > t$,

$$\begin{aligned}[\text{Euler}] \quad \frac{d\mathbf{x}}{d\tau} &\approx \frac{\mathbf{x}(\tau) - \mathbf{x}(s)}{\tau - s} \quad \forall \tau \in [t, s]. \\ \Rightarrow \underbrace{\tilde{\Psi}_{s \rightarrow t}(\mathbf{x}_s)}_{\mathbf{x}_t^{\text{Euler}}} &\approx \mathbf{x}_s - (s - t) \left[f(\mathbf{x}_s, s) - \frac{1}{2} g^2(s) s_{\phi \times}(\mathbf{x}_s, s) \right].\end{aligned}$$

That is, the Euler solver produces a discrete trajectory $\{\mathbf{x}_{t_i}^{\text{Euler}}\}_{i=0}^m$ that approximates the continuous.

1 Semilinear Structure.

$$\frac{d\mathbf{x}(t)}{dt} = \underbrace{f(t)\mathbf{x}(t)}_{\text{linear}} - \underbrace{\frac{1}{2}g^2(t)s_{\phi^\times}(\mathbf{x}(t), t)}_{\text{nonlinear}}. \quad (1)$$

2 Parametrizations Beyond the Score.

For the standard perturbation kernel $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 I)$, it is preferable to use ϵ -prediction (or $\mathbf{x}$ -, or $\mathbf{v}$ -prediction) with the closed-form relation

$$s(\mathbf{x}, t) = -\frac{1}{\sigma_t} \epsilon(\mathbf{x}, t).$$

Reason. Since σ_t is an increasing function, the score can blow up at the rate $1/\sigma_t^2$, while ϵ stays bounded.

8 Exponential Integrators for semilinear PF-ODEs.

Let the nonlinear part of (1) be

$$\mathbf{N}(\mathbf{x}(t), t) := \frac{1}{2} \frac{g^2(t)}{\sigma_t} \epsilon_{\phi \times}(\mathbf{x}(t), t),$$

and introduce the exponential integrator

$$\mathcal{E}(s \rightarrow t) := \exp\left(\int_s^t f(u) du\right).$$

Then, we can induce

$$\tilde{\Psi}_{s \rightarrow t}(\mathbf{x}_s) = \mathcal{E}(s \rightarrow t) \mathbf{x}_s + \frac{1}{2} \int_s^t \frac{g^2(\tau)}{\sigma_\tau} \mathcal{E}(\tau \rightarrow t) \epsilon_{\phi \times}(\mathbf{x}_\tau, \tau) d\tau,$$

since

$$\mathcal{E}^{-1}(s \rightarrow t) \left(\frac{d\mathbf{x}(t)}{dt} - f(t)\mathbf{x}(t) \right) = \frac{d}{dt} [\mathcal{E}^{-1}(s \rightarrow t) \mathbf{x}(t)].$$

It implies the update rule

$$[\text{Exponential-Euler}] \quad \mathbf{N}(\mathbf{x}(\tau), \tau) \approx \mathbf{N}(\mathbf{x}(s), s), \quad \forall \tau \in [s - \Delta s, s].$$

$$\begin{aligned} \Rightarrow \quad \mathbf{x}_{s-\Delta s}^{\text{Exp-Euler}} &= e^{-f(s)\Delta s} \mathbf{x}_s + \frac{e^{-f(s)\Delta s} - 1}{f(s)} \mathbf{N}(\mathbf{x}_s, s) \\ &\approx e^{-f(s)\Delta s} \mathbf{x}_s - \Delta s \mathbf{N}(\mathbf{x}_s, s), \quad \text{as } f \rightarrow 0. \end{aligned}$$

Revisit:

$$[\text{Euler}] \quad f(\tau) \mathbf{x}_\tau + \mathbf{N}(\mathbf{x}_\tau, \tau) \approx f(s) \mathbf{x}_s + \mathbf{N}(\mathbf{x}_s, s), \quad \forall \tau \in [s - \Delta s, s].$$

$$\Rightarrow \quad \mathbf{x}_{s-\Delta s}^{\text{Euler}} = (1 - f(s)\Delta s) \mathbf{x}_s - \Delta s \mathbf{N}(\mathbf{x}_s, s).$$

The difference of the linear factors is

$$e^{-f(s)\Delta s} - (1 - f(s)\Delta s) = \frac{(-f(s)\Delta s)^2}{2} + \mathcal{O}(\Delta s^3),$$

which is critical, especially when Δs is not small.

More precisely,

$$\begin{aligned}\mathbf{x}_{s-\Delta s}^{\text{exact}} &= \mathbf{x}_s - \Delta s (f\mathbf{x}_s + \mathbf{N}_s) + \frac{\Delta s^2}{2} (f^2\mathbf{x}_s + f\mathbf{N}_s + \dot{\mathbf{N}}_s) + \mathcal{O}(\Delta s^3), \\ \mathbf{x}_{s-\Delta s}^{\text{Euler}} &= \mathbf{x}_s - \Delta s (f\mathbf{x}_s + \mathbf{N}_s), \\ \mathbf{x}_{s-\Delta s}^{\text{Exp-Euler}} &= e^{-f\Delta s} \mathbf{x}_s + \frac{e^{-f\Delta s} - 1}{f} \mathbf{N}_s \\ &= \mathbf{x}_s - \Delta s (f\mathbf{x}_s + \mathbf{N}_s) + \frac{\Delta s^2}{2} (f^2\mathbf{x}_s + f\mathbf{N}_s) + \mathcal{O}(\Delta s^3).\end{aligned}$$

- Time Stepping Methods
 - DDIM
 - DEIS
 - DPM-Solver family
- Time Parallel Methods
 - ParaDiGMs

True Computational Cost (NFEs). The wall-clock cost of a solver is determined

- **not** by the number of steps,
- but by how many times we need to call the network.

If the solver calls the network m times per step over N steps, the number of function evaluations (NFEs) is

$$\text{NFE} = mN,$$

which is the significant measure whether the solver is fast or not.

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models**
- 3 DEIS : Diffusion Exponential Integrator Sampler
- 4 DPM-Solver : Diffusion Probabilistic Model Solver
- 5 DPM-Solver++
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps

Original Motivation (Variational View)

In the training step of DDPM, our important objective is $q(\mathbf{x}_t \mid \mathbf{x}_0)$, not $q(\mathbf{x}_t \mid \mathbf{x}_{t-\Delta t})$, even if we assume the Markov chain process.

In the reverse sampling, however, the reverse posterior depends on the specific one-step forward kernel:

$$q(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t \mid \mathbf{x}_{t-\Delta t}) q(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)}.$$

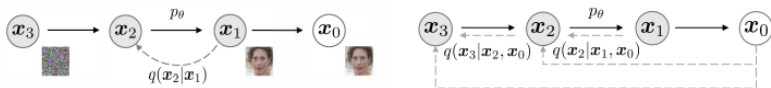


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Then, we can focus only on the marginals $p_t(\mathbf{x}_t \mid \mathbf{x}_0)$. Can we reconstruct the reverse conditionals $\pi(\mathbf{x}_t \mid \mathbf{x}_s, \mathbf{x}_0)$, for $t < s$, which preserve the marginals?:

$$\int \pi(\mathbf{x}_t \mid \mathbf{x}_s, \mathbf{x}_0) p_s(\mathbf{x}_s \mid \mathbf{x}_0) d\mathbf{x}_s = p_t(\mathbf{x}_t \mid \mathbf{x}_0).$$

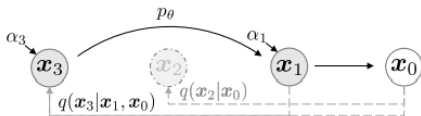


Figure 2: Graphical model for accelerated generation, where $\tau = [1, 3]$.

Lemma 9.2.2. Marginal-Consistent Reverse Conditionals

For the standard perturbation kernel $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 I)$, and for **any** positive $c_{t,s} \leq \sigma_t$, if

$$\pi(\mathbf{x}_t \mid \mathbf{x}_s, \mathbf{x}_0) = \mathcal{N}\left(\alpha_t \mathbf{x}_0 + \frac{\sqrt{\sigma_t^2 - c_{t,s}^2}}{\sigma_s}(\mathbf{x}_s - \alpha_s \mathbf{x}_0), c_{t,s}^2 I\right),$$

then this process satisfies the marginal-consistency condition.

Proof.

$$\begin{aligned}\mathbf{x}_t &= \alpha_t \mathbf{x}_0 + \frac{\sqrt{\sigma_t^2 - c_{t,s}^2}}{\sigma_s}(\mathbf{x}_s - \alpha_s \mathbf{x}_0) + c_{t,s} \epsilon \\ &= \alpha_t \mathbf{x}_0 + \sqrt{\sigma_t^2 - c_{t,s}^2} \epsilon' + c_{t,s} \epsilon \\ &= \alpha_t \mathbf{x}_0 + \sigma_t \epsilon'', \quad (\epsilon \perp \epsilon'),\end{aligned}$$

which preserves the marginals.

Remark. By replacing $\mathbf{x}_0$ with an $\mathbf{x}$ -prediction $\mathbf{x}_{\phi \times}(\mathbf{x}_s, s) = \frac{\mathbf{x}_s - \sigma_s \epsilon_{\phi \times}(\mathbf{x}_s, s)}{\alpha_s}$, we obtain

$$\mathbf{x}_t = \frac{\alpha_t}{\alpha_s} \mathbf{x}_s + \left(\sqrt{\sigma_t^2 - c_{t,s}^2} - \frac{\alpha_t}{\alpha_s} \sigma_s \right) \epsilon_{\phi \times}(\mathbf{x}_s, s) + c_{t,s} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I).$$

Special choices of $c_{t,s}$ include

- **DDPM:**

$$c_{t,s} = \frac{\sigma_s}{\sigma_t} \left(\sigma_s^2 - \frac{\alpha_s^2}{\alpha_t^2} \sigma_t^2 \right).$$

- **Deterministic:**

$$c_{t,s} = 0.$$

- **Interpolation:**

$$c_{t,s} = \eta \frac{\sigma_s}{\sigma_t} \left(\sigma_s^2 - \frac{\alpha_s^2}{\alpha_t^2} \sigma_t^2 \right).$$

Q. Compare the DDIM with the DDPM sampling in terms of the marginal distribution.

A. Both DDIM and DDPM preserve the same marginals $p_t(\mathbf{x}_t \mid \mathbf{x}_0)$, but they differ in their transition kernels: Markovian kernel v.s. non-Markovian kernel.

Why? Let's start with the transition kernel of DDPM: For $s > t$, and the forward step $\mathbf{x}_k = \alpha_k \mathbf{x}_0 + \sigma_k \epsilon$ where $\alpha_k^2 + \sigma_k^2 = 1$,

$$q(\mathbf{x}_s | \mathbf{x}_t) = N\left(\sqrt{\frac{\alpha_s^2}{\alpha_t^2}} \mathbf{x}_t, \left(1 - \frac{\alpha_s}{\alpha_t}\right) I\right) = N\left(\frac{\alpha_s}{\alpha_t} \mathbf{x}_t, \underbrace{\left(\sigma_s^2 - \frac{\alpha_s^2}{\alpha_t^2} \sigma_t^2\right)}_{\sigma_{s|t}^2} I\right).$$

Then, to compare the kernel,

$$\begin{aligned} q(\mathbf{x}_t | \mathbf{x}_s, \mathbf{x}_0) &\propto q(\mathbf{x}_t | \mathbf{x}_0) q(\mathbf{x}_s | \mathbf{x}_t, \mathbf{x}_0) \\ &= N(\alpha_t \mathbf{x}_0, \sigma_t^2 I) N\left(\frac{\alpha_s}{\alpha_t} \mathbf{x}_t, \sigma_{s|t}^2 I\right) \\ &\propto N\left(\frac{\alpha_t}{\sigma_s^2} \sigma_{s|t} \mathbf{x}_0 + \frac{\alpha_s \sigma_t^2}{\alpha_t \sigma_s^2} \mathbf{x}_s, \frac{\sigma_t^2}{\sigma_s^2} \sigma_{s|t}^2 I\right). \end{aligned}$$

Hence, the transition kernel of DDPM is equivalent to DDIM with the specific $c_{t,s}$, since

$$\sqrt{\sigma_t^2 - c_{t,s}^2} = \sqrt{\sigma_t^2 - \frac{\sigma_t^2}{\sigma_s^2} \left(\sigma_s^2 - \frac{\alpha_s^2}{\alpha_t^2} \sigma_t^2\right)} = \frac{\alpha_s \sigma_t^2}{\alpha_t \sigma_s^2}.$$

DDPM Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(0, I)$ 
2: for  $t = T, T-1, \dots, 1$  do
3:    $\hat{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$ 
4:    $\hat{\mathbf{x}}_0 \leftarrow \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}$ 
5:    $z \sim \mathcal{N}(0, I)$ 
6:    $\mathbf{x}_{t-1} \leftarrow \frac{\sqrt{1 - \beta_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \hat{\mathbf{x}}_0 + \frac{\sqrt{\bar{\alpha}_t} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_t + \sigma_t z$ 
7: end for
8: return  $\mathbf{x}_0$ 
```

DDIM Sampling

```
1:  $\mathbf{x}_{t_S} \sim \mathcal{N}(0, I)$ 
2: for  $0 = t_0 < t_1 < \dots < t_S = T$  do
3:    $\hat{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_{t_i}, t_i)$ 
4:    $\hat{\mathbf{x}}_0 \leftarrow \frac{\mathbf{x}_{t_i} - \sqrt{1 - \bar{\alpha}_{t_i}} \hat{\epsilon}}{\sqrt{\bar{\alpha}_{t_i}}}$ 
5:    $\sigma_i \leftarrow \eta \sqrt{\frac{1 - \bar{\alpha}_{t_{i-1}}}{1 - \bar{\alpha}_{t_i}}} \sqrt{1 - \frac{\bar{\alpha}_{t_i}}{\bar{\alpha}_{t_{i-1}}}}$ 
6:    $z_i \sim \mathcal{N}(0, I)$ 
7:    $\mathbf{x}_{t_{i-1}} \leftarrow \sqrt{\bar{\alpha}_{t_{i-1}}} \hat{\mathbf{x}}_0 + \sqrt{1 - \bar{\alpha}_{t_{i-1}} - \sigma_i^2} \hat{\epsilon} + \sigma_i z_i$ 
8: end for
9: return  $\mathbf{x}_0$ 
```

Interpreting DDIM as an ODE Solver

Recall the Exponential–Euler sampler:

$$\tilde{\mathbf{x}}_t = \mathcal{E}(s \rightarrow t) \mathbf{x}_s + \frac{1}{2} \int_s^t \frac{g^2(\tau)}{\sigma_\tau} \mathcal{E}(\tau \rightarrow t) \epsilon_{\phi \times}(\mathbf{x}_\tau, \tau) d\tau.$$

Now, applying the Euler rule to the intractable part of the integral term:

$$[\text{Euler}] \quad \epsilon_{\phi \times}(\mathbf{x}_\tau, \tau) \approx \epsilon_{\phi \times}(\mathbf{x}_s, s), \quad \forall \tau \in [t, s].$$

$$\begin{aligned} \Rightarrow \quad \tilde{\mathbf{x}}_t &= \mathcal{E}(s \rightarrow t) \tilde{\mathbf{x}}_s + \frac{1}{2} \int_s^t \frac{g^2(\tau)}{\sigma_\tau} \mathcal{E}(\tau \rightarrow t) d\tau \epsilon_{\phi \times}(\tilde{\mathbf{x}}_s, s) \\ &= \frac{\alpha_t}{\alpha_s} \tilde{\mathbf{x}}_s - \alpha_t \left(\frac{\sigma_s}{\alpha_s} - \frac{\sigma_t}{\alpha_t} \right) \epsilon_{\phi \times}(\tilde{\mathbf{x}}_s, s). \quad (\text{Deterministic DDIM}) \end{aligned}$$

using the relations

$$f(t) = \frac{\alpha'_t}{\alpha_t}, \quad g^2(t) = \frac{d}{dt}(\sigma_t^2) - 2 \frac{\alpha'_t}{\alpha_t} \sigma_t^2.$$

Corollary 9.2.1.

$$\begin{aligned}
 \tilde{\mathbf{x}}_t &= \frac{\alpha_t}{\alpha_s} \tilde{\mathbf{x}}_s + \alpha_t \left(\frac{\sigma_t}{\alpha_t} - \frac{\sigma_s}{\alpha_s} \right) \boldsymbol{\epsilon}^*(\tilde{\mathbf{x}}_s, s) \\
 &= \frac{\sigma_t}{\sigma_s} \tilde{\mathbf{x}}_s + \alpha_s \left(\frac{\alpha_t}{\alpha_s} - \frac{\sigma_t}{\sigma_s} \right) \mathbf{x}^*(\tilde{\mathbf{x}}_s, s) \\
 &= \frac{\alpha_t}{\alpha_s} \tilde{\mathbf{x}}_s + \sigma_s^2 \left(\frac{\alpha_t}{\alpha_s} - \frac{\sigma_t}{\sigma_s} \right) \mathbf{s}^*(\tilde{\mathbf{x}}_s, s) \\
 &= \alpha_t \mathbf{x}^*(\tilde{\mathbf{x}}_s, s) + \sigma_t \boldsymbol{\epsilon}^*(\tilde{\mathbf{x}}_s, s).
 \end{aligned}$$

Proof. For $\mathbf{x}_s = \alpha_s \mathbf{x}_0 + \sigma_s \boldsymbol{\epsilon}$, and the optimal predictors $\boldsymbol{\epsilon}^* = \mathbb{E}[\boldsymbol{\epsilon} \mid \mathbf{x}_s]$, $\mathbf{x}^* = \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_s]$, use the relations

$$\boldsymbol{\epsilon}^* = \frac{\tilde{\mathbf{x}}_s - \alpha_s \mathbf{x}^*}{\sigma_s}, \quad \mathbf{s}^* = -\frac{\boldsymbol{\epsilon}^*}{\sigma_s}.$$

□

The last equation means that DDIM does not inject noise in the denoising process, unlike DDPM.

Euler method for v-prediction

The natural form of v-prediction in PF-ODE is

$$\frac{d\mathbf{x}}{d\tau} = \alpha'_\tau \mathbf{x}^* + \sigma'_\tau \boldsymbol{\epsilon}^*.$$

Then,

$$[\text{Euler}] \quad \mathbf{x}^*(\mathbf{x}(\tau), \tau) \approx \mathbf{x}^*(\mathbf{x}_s, s), \quad \boldsymbol{\epsilon}^*(\mathbf{x}(\tau), \tau) \approx \boldsymbol{\epsilon}^*(\mathbf{x}_s, s), \quad \forall \tau \in [s, t].$$

$$\begin{aligned} \Rightarrow \quad \tilde{\mathbf{x}}_t &= \tilde{\mathbf{x}}_s + \int_s^t (\alpha'_\tau \mathbf{x}^* + \sigma'_\tau \boldsymbol{\epsilon}^*) d\tau \\ &\approx (\alpha_s \mathbf{x}^* + \sigma_s \boldsymbol{\epsilon}^*) + (\alpha_t - \alpha_s) \mathbf{x}^*(\tilde{\mathbf{x}}_s, s) + (\sigma_t - \sigma_s) \boldsymbol{\epsilon}^*(\tilde{\mathbf{x}}_s, s) \\ &= \alpha_t \mathbf{x}^* + \sigma_t \boldsymbol{\epsilon}^*. \quad (\text{Question 1\&2}) \end{aligned}$$

Hence, we can consider DDIM as the plain Euler sampling of v-prediction.

Cf. ϵ - (or x -, s -) prediction is not, since their linear parts are explicit:

$$\frac{d\mathbf{x}}{dt} = \underbrace{\frac{\alpha'}{\alpha} \mathbf{x}}_{\text{linear}} + \left(\sigma' - \frac{\alpha'}{\alpha} \sigma \right) \boldsymbol{\epsilon}^*.$$

Thus, the plain Euler method approximates the linear part as well as the nonlinear part.

DDIM's First Order Error

v-prediction (i.e., plain Euler of $\frac{d\mathbf{x}}{d\tau} = \mathbf{v}(\mathbf{x}(\tau), \tau)$)

$$[\text{Euler}] \quad \mathbf{v}(\mathbf{x}_\tau, \tau) = \mathbf{v}(\mathbf{x}_s, s), \quad \forall \tau \in [t, s].$$

$$\Rightarrow \quad \tilde{\mathbf{x}}_t^{\text{Euler}} = \mathbf{x}_s - h \mathbf{v}(\mathbf{x}_s, s).$$

$$\Rightarrow \quad \mathbf{x}(t) - \tilde{\mathbf{x}}_t^{\text{Euler}} = \int_t^s \left[\mathbf{v}(\mathbf{x}_s, s) - \mathbf{v}(\mathbf{x}(\tau), \tau) \right] d\tau = \int_t^s (\tau - t) \frac{d}{d\tau} \mathbf{v}(\mathbf{x}(\tau), \tau) d\tau.$$

$$\therefore \quad \left\| \mathbf{x}(t) - \tilde{\mathbf{x}}_t^{\text{Euler}} \right\| \leq \frac{h^2}{2} \sup_{\tau \in [t, s]} \underbrace{\left\| \frac{d}{d\tau} \mathbf{v}(\mathbf{x}(\tau), \tau) \right\|}_{\text{acceleration}}.$$

ϵ -prediction (i.e., exponential-Euler of $\frac{d\mathbf{x}}{d\tau} = f(\tau)\mathbf{x}(\tau) + \mathcal{N}(\mathbf{x}(\tau), \tau)$)

$$[\text{Exp-Euler}] \quad \mathcal{N}(\mathbf{x}_\tau, \tau) = \mathcal{N}(\mathbf{x}_s, s), \quad \forall \tau \in [t, s].$$

$$\Rightarrow \quad \tilde{\mathbf{x}}_t^{\text{Exp-Euler}} = \mathcal{E}(s \rightarrow t) \mathbf{x}_s + \left(\int_s^t \mathcal{E}(\tau \rightarrow t) d\tau \right) \mathcal{N}(\mathbf{x}_s, s).$$

$$\Rightarrow \quad \mathbf{x}(t) - \tilde{\mathbf{x}}_t^{\text{Exp-Euler}} = \int_s^t \mathcal{E}(\tau \rightarrow t) \left[\mathcal{N}(\mathbf{x}(\tau), \tau) - \mathcal{N}(\mathbf{x}_s, s) \right] d\tau.$$

$$\therefore \quad \left\| \mathbf{x}(t) - \tilde{\mathbf{x}}_t^{\text{Exp-Euler}} \right\| \leq \left(\sup_{\tau \in [t, s]} |\mathcal{E}(\tau \rightarrow t)| \right) \frac{h^2}{2} \sup_{\tau \in [t, s]} \underbrace{\left\| \frac{d}{d\tau} \mathcal{N}(\mathbf{x}(\tau), \tau) \right\|}_{\text{velocity of nonlinear part}}.$$

DDIM's target = CFM target

Under the linear Gaussian path $x_\tau = \alpha_\tau x_0 + \sigma_\tau \epsilon$, the DDIM conditional map $((\Psi_{s \rightarrow t})_\# p_s(\cdot | x_0) = p_t(\cdot | x_0))$ is

$$\Psi_{s \rightarrow t}(x_s | x_0) = \frac{\sigma_t}{\sigma_s} x_s + \left(\alpha_t - \alpha_s \frac{\sigma_t}{\sigma_s} \right) x_0,$$

and its (conditional) velocity is

$$\begin{aligned} v_t^*(x | x_0) &= \frac{d}{dh} \Psi_{t \rightarrow t+h}(x | x_0) \Big|_{h=0} \\ &= \frac{\sigma'_t}{\sigma_t} x + \left(\alpha'_t - \alpha_t \frac{\sigma'_t}{\sigma_t} \right) x_0 \\ &= \alpha'_t x_0 + \sigma'_t \epsilon. \end{aligned}$$

which is the target of Conditional Flow Matching (CFM):

$$\mathcal{L}_{\text{CFM}} = \mathbb{E} \left[\|v_\phi(x_t, t) - v_t^*(x_t | x_0)\|^2 \right].$$

Also, marginalizing $v_t^*(x | x_0)$ given $x_t = x$ yields

$$\underbrace{\mathbb{E}[v_t^*(x | x_0) | x_t = x]}_{v^*(x, t)} = \underbrace{\alpha'_t \mathbb{E}[x_0 | x_t = x]}_{x^*(x, t)} + \underbrace{\sigma'_t \mathbb{E}[\epsilon | x_t = x]}_{\epsilon^*(x, t)}.$$

which is the v -parameterized PF-ODE drift.

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models
- 3 DEIS : Diffusion Exponential Integrator Sampler**
- 4 DPM-Solver : Diffusion Probabilistic Model Solver
- 5 DPM-Solver++
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps

The strongest assumption in DDIM sampler is

$$[\text{Euler}] \quad \epsilon_{\phi^x}(x_\tau, \tau) \approx \epsilon_{\phi^x}(x_s, s), \quad \forall \tau \in [t, s].$$

$$\Rightarrow \quad \int_s^t \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t) \epsilon_{\phi^x}(x_\tau, \tau) d\tau \approx \left(\int_s^t \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t) d\tau \right) \epsilon_{\phi^x}(x_s, s).$$

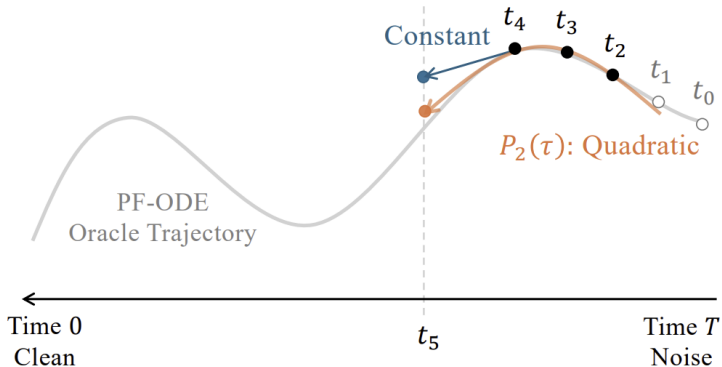
But in the sequential sampling process, we may prefer to reuse previous outputs (anchors) to consider the trend:

$$[n\text{-th Adams-Bashforth}] \quad \epsilon_{\phi^x}(x_\tau, \tau) \approx P_{n-1}(\tau), \quad \forall \tau \in [t_i, t_{i-1}].$$

where $P_{n-1}(t_{i-1}) = \epsilon_{\phi^x}(x_{t_{i-1}}, t_{i-1})$, $\dots$, $P_{n-1}(t_{i-n}) = \epsilon_{\phi^x}(x_{t_{i-n}}, t_{i-n})$.

$$\Rightarrow \quad \int_{t_{i-1}}^{t_i} \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t_i) \epsilon_{\phi^x}(x_\tau, \tau) d\tau \approx \int_{t_{i-1}}^{t_i} \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t_i) P_{n-1}(\tau) d\tau.$$

Diffusion Exponential Integrator Sampler (DEIS)



Lagrangian Polynomial

Assume we know some points

$$(\tau_0, Y_0), (\tau_1, Y_1), \dots, (\tau_n, Y_n), \quad \tau_0 < \tau_1 < \dots < \tau_n.$$

The n -th Lagrangian polynomial in $\tau \geq \tau_n$ is

- $n = 0$ (constant)

$$Y(\tau) = Y_n.$$

- $n = 1$ (linear)

$$Y(\tau) = \frac{\tau - \tau_n}{\tau_{n-1} - \tau_n} Y_{n-1} + \frac{\tau - \tau_{n-1}}{\tau_n - \tau_{n-1}} Y_n.$$

- n (general)

$$Y(\tau) = \sum_{j=0}^n \ell_j(\tau) Y_j, \quad \text{where} \quad \ell_j(\tau_k) = \delta_{jk}, \quad \sum_{j=0}^n \ell_j(\tau) = 1.$$

That is,

$$\ell_j(\tau) = \frac{\tau - \tau_0}{\tau_j - \tau_0} \dots \frac{\tau - \tau_{j-1}}{\tau_j - \tau_{j-1}} \frac{\tau - \tau_{j+1}}{\tau_j - \tau_{j+1}} \dots \frac{\tau - \tau_n}{\tau_j - \tau_n} = \prod_{\substack{0 \leq k \leq n \\ k \neq j}} \frac{\tau - \tau_k}{\tau_j - \tau_k}.$$

Warm-up and Main Evaluation of AB-DEIS- n

Let's start AB-DEIS- n . In i -th evaluation step,

- For the initial step, $i = 1, \dots, n$, since we can access only i evaluations,

$$(\tau_j, Y_j) := (t_{i-1-j}, \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j})), \quad j = 0, 1, \dots, i-1.$$

We set the degree to $i-1$ and define

$$P_{i-1}(\tau) = \sum_{j=0}^{i-1} \left(\prod_{\substack{0 \leq k \leq i-1 \\ k \neq j}} \frac{\tau - t_{i-1-k}}{t_{i-1-j} - t_{i-1-k}} \right) \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j}).$$

- For the main step, $i = n+1, \dots, M$, since all n past observations are available,

$$(\tau_j, Y_j) := (t_{i-1-j}, \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j})), \quad j = 0, 1, \dots, n,$$

then

$$P_{n-1}(\tau) = \sum_{j=0}^n \left(\prod_{\substack{0 \leq k \leq n \\ k \neq j}} \frac{\tau - t_{i-1-k}}{t_{i-1-j} - t_{i-1-k}} \right) \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j}).$$

Warm-up and Main Evaluation of AB-DEIS- n

$$\begin{aligned}
 & \therefore \int_{t_{i-1}}^{t_i} \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t_i) \epsilon_{\phi \times}(x_\tau, \tau) d\tau \\
 & \approx \int_{t_{i-1}}^{t_i} \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t_i) \sum_{j=0}^{n \wedge (i-1)} \left(\prod_{\substack{0 \leq k \leq n \wedge (i-1) \\ k \neq j}} \frac{\tau - t_{i-1-k}}{t_{i-1-j} - t_{i-1-k}} \right) \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j}) d\tau \\
 & = \sum_{j=0}^{n \wedge (i-1)} \underbrace{\left(\int_{t_{i-1}}^{t_i} \frac{g^2(\tau)}{2\sigma_\tau} \mathcal{E}(\tau \rightarrow t_i) \left(\prod_{\substack{0 \leq k \leq n \wedge (i-1) \\ k \neq j}} \frac{\tau - t_{i-1-k}}{t_{i-1-j} - t_{i-1-k}} \right) d\tau \right)}_{C_{ij}} \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j}).
 \end{aligned}$$

AB-DEIS- n Update Rule

$$\tilde{x}_{t_i} = \mathcal{E}(t_{i-1} \rightarrow t_i) \tilde{x}_{t_{i-1}} + \sum_{j=0}^{n \wedge (i-1)} C_{ij} \epsilon_{\phi \times}(\tilde{x}_{t_{i-1-j}}, t_{i-1-j})$$

(As we know, DDIM = AB-DEIS-0).

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models
- 3 DEIS : Diffusion Exponential Integrator Sampler
- 4 DPM-Solver : Diffusion Probabilistic Model Solver**
- 5 DPM-Solver++
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps

1. Time Reparameterization via Log-SNR

Recall the semilinear ϵ -prediction form of the PF-ODE:

$$\frac{dx_t}{dt} = \frac{\alpha'_t}{\alpha_t} x_t - \sigma_t \left(\frac{\alpha'_t}{\alpha_t} - \frac{\sigma'_t}{\sigma_t} \right) \epsilon_{\phi \times}(x_t, t). \quad (2)$$

Now consider the time reparameterization using the half-log SNR

$$\lambda_t := \frac{1}{2} \log \frac{\alpha_t^2}{\sigma_t^2} = \log \frac{\alpha_t}{\sigma_t},$$

which is strictly decreasing in t for common noise schedules.

Let a hat ($\hat{\cdot}$) denote the quantity in λ -time. For the mapping $t = t_\lambda(\lambda)$, define

$$\hat{x}_\lambda := x_{t_\lambda(\lambda)}, \quad \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda) := \epsilon_{\phi \times}(x_{t_\lambda(\lambda)}, t_\lambda(\lambda)).$$

Proposition 9.4.1: Exponentially Weighted Exact Solution

Using log-SNR, the PF-ODE can be reparameterized as

$$\frac{d\hat{x}_\lambda}{d\lambda} = \frac{\alpha'_\lambda}{\alpha_\lambda} \hat{x}_\lambda - \sigma_\lambda \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda).$$

That is, given x_s , the exact solution is

$$\tilde{\Psi}_{s \rightarrow t}(x_s) = \frac{\alpha_t}{\alpha_s} x_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda) d\lambda.$$

proof. From (2),

$$\frac{dx_t}{dt} = \frac{d \log \alpha_t}{dt} x_t - \sigma_t \frac{d\lambda_t}{dt} \epsilon_{\phi \times}(x_t, t).$$

Using the chain rule,

$$\begin{aligned} \frac{d\hat{x}_\lambda}{d\lambda} &= \frac{dx_t}{dt} \frac{dt}{d\lambda} = \left(\frac{d\lambda_t}{dt} \right)^{-1} \frac{dx_t}{dt} \\ &= \left(\frac{d\lambda_t}{dt} \right)^{-1} \left[\frac{d \log \alpha_t}{dt} x_t - \sigma_t \frac{d\lambda_t}{dt} \epsilon_{\phi \times}(x_t, t) \right] \\ &= \left(\frac{d\lambda_t}{dt} \right)^{-1} \left[\frac{d \log \alpha_\lambda}{d\lambda} \frac{d\lambda_t}{dt} \hat{x}_\lambda - \sigma_\lambda \frac{d\lambda_t}{dt} \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda) \right] \\ &= \frac{d \log \alpha_\lambda}{d\lambda} \hat{x}_\lambda - \sigma_\lambda \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda), \end{aligned}$$

and we can derive the flow map using Exponential Integrator.

Intuition of Reparameterization

The λ -schedule allows us to explore more information as the process gets closer to the complicated (data) distribution.

- For strictly monotone $\lambda(t)$,

$$\Delta t \approx \frac{\Delta \lambda}{|\lambda'(t)|}.$$

- This reparameterization changes the speed, not the path, of the solution:

$$\frac{d\hat{x}_\lambda}{d\lambda} = \frac{1}{\lambda'(t)} \frac{dx_t}{dt}.$$

Example: OT-FM scheduler. For $(\alpha_t, \sigma_t) = (1 - t, t)$, we have

$$\lambda(t) = \log \frac{1-t}{t}, \quad \lambda'(t) = -\frac{1}{t(1-t)}.$$

2. Taylor Expansion with High-Order Approximation

To approximate the intractable integrand in

$$\tilde{x}_t = \frac{\alpha_t}{\alpha_s} \tilde{x}_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda) d\lambda,$$

we introduce a Taylor expansion about λ_s :

$$\hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda) = \sum_{k=0}^{n-1} \frac{(\lambda - \lambda_s)^k}{k!} \hat{\epsilon}_{\phi \times}^{(k)}(\hat{x}_{\lambda_s}, \lambda_s) + \mathcal{O}((\lambda - \lambda_s)^n).$$

Hence, for $h := \lambda_t - \lambda_s$,

$$\tilde{x}_t = \frac{\alpha_t}{\alpha_s} \tilde{x}_s - \sigma_t \sum_{k=0}^{n-1} \underbrace{\left(\int_0^h e^{-\lambda+h} \frac{\lambda^k}{k!} d\lambda \right)}_{=: C_k} \hat{\epsilon}_{\phi \times}^{(k)}(\hat{x}_{\lambda_s}, \lambda_s) + \mathcal{O}(h^{n+1}),$$

which can precompute C_k using integration by parts:

$$C_k = e^h - 1 - h - \frac{h^2}{2!} - \dots - \frac{h^k}{k!}.$$

Preliminary: Finite Difference Approximation

To approximate the k -th derivative of $\hat{\epsilon}_{\phi \times}(\hat{x}_s, s)$, we use the Finite Difference Method (FDM), which is a standard approach for constructing single-step ODE solvers.

- **Forward (Backward) difference**

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h),$$

or analogously using $x - h$.

- **Central difference**

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2).$$

- **Second-order derivative**

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2).$$

Preliminary: Runge–Kutta Method

It is a discretization method for approximating the solution of an ODE $\frac{dx}{dt} = f(x, t)$, with step size h .

- **RK1: Euler method**

$$t_{n+1} = t_n + h, \quad x_{n+1} = x_n + h f(x_n, t_n).$$

- **RK2: Midpoint method**

$$\begin{aligned} t_{\text{mid}} &= t_n + \frac{h}{2}, & x_{\text{mid}} &= x_n + \frac{h}{2} f(x_n, t_n), \\ \Rightarrow t_{n+1} &= t_n + h, & x_{n+1} &= x_n + h f(x_{\text{mid}}, t_{\text{mid}}). \end{aligned}$$

- **RK2: Heun method**

$$\begin{aligned} x_{\text{pred}} &= x_n + h f(x_n, t_n), \\ \Rightarrow t_{n+1} &= t_n + h, & x_{n+1} &= x_n + h \frac{f(x_n, t_n) + f(x_{\text{pred}}, t_{n+1})}{2}. \end{aligned}$$

- **RK4: Classical Runge–Kutta**

$$\begin{aligned} t_{\text{mid}} &= t_n + \frac{h}{2}, & x_{\text{mid1}} &= x_n + \frac{h}{2} f(x_n, t_n), & x_{\text{mid2}} &= x_n + \frac{h}{2} f(x_{\text{mid1}}, t_{\text{mid}}), \\ \Rightarrow x_{\text{pred}} &= x_n + h f(x_{\text{mid2}}, t_{\text{mid}}), \\ \Rightarrow t_{n+1} &= t_n + h, & x_{n+1} &= x_n + h \frac{f(x_n, t_n) + 2f(x_{\text{mid1}}, t_{\text{mid}}) + 2f(x_{\text{mid2}}, t_{\text{mid}}) + f(x_{\text{pred}}, t_{n+1})}{6}. \end{aligned}$$

High-Order Derivative Approximation

We combine FDM with the viewpoint of RK2: midpoint.

- **DPM-Solver-1**

$$\begin{aligned}\tilde{x}_t &= \frac{\alpha_t}{\alpha_s} \tilde{x}_s - \sigma_t \left(e^h - 1 \right) \hat{\epsilon}_{\phi \times} (\hat{x}_{\lambda_s}, \lambda_s) + \mathcal{O}(h^2) \\ &= \frac{\alpha_t}{\alpha_s} \tilde{x}_s - \alpha_t \left(\frac{\sigma_s}{\alpha_s} - \frac{\sigma_t}{\alpha_t} \right) \epsilon_{\phi \times} (\tilde{x}_s, s) + \mathcal{O}(h^2).\end{aligned}\quad (\text{DDIM})$$

- **DPM-Solver-2**

For a midpoint $s^{\text{mid}} = t_\lambda(\lambda_s + \gamma h) \in (s, t]$ with $\gamma \in (0, 1]$, define

$$x^{\text{mid}} = \frac{\alpha_{s^{\text{mid}}}}{\alpha_s} \tilde{x}_s - \sigma_{s^{\text{mid}}} \left(e^{\gamma h} - 1 \right) \epsilon_{\phi \times} (\tilde{x}_s, s).$$

Then it is used for the evaluation of $\epsilon_{\phi \times}^{(1)}$ (i.e., $x^{\text{mid}} \approx x_s + \gamma h$):

$$\begin{aligned}\tilde{x}_t &= \frac{\alpha_t}{\alpha_s} \tilde{x}_s - \sigma_t (e^h - 1) \epsilon_{\phi \times} (\tilde{x}_s, s) \\ &\quad - \sigma_t (e^h - 1 - h) \frac{\epsilon_{\phi \times} (x^{\text{mid}}, s^{\text{mid}}) - \epsilon_{\phi \times} (\tilde{x}_s, s)}{\gamma h} + \mathcal{O}(h^3).\end{aligned}$$

Q. Why is equivalent following algorithm to the above DPM-Solver-2 up to $\mathcal{O}(h^3)$?

Algorithm 5 DPM-Solver-2 (with $\gamma = \frac{1}{2}$).

Input: initial value $\mathbf{x}_T$, time steps $\{t_i\}_{i=0}^M$, model $\epsilon_{\phi \times}$

```

1:  $\tilde{\mathbf{x}}_{t_0} \leftarrow \mathbf{x}_T$ 
2: for  $i \leftarrow 1$  to  $M$  do
3:    $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$ 
4:    $s_i^{\text{mid}} \leftarrow t_\lambda \left( \frac{\lambda_{t_{i-1}} + \lambda_{t_i}}{2} \right)$ 
5:    $\mathbf{x}_i^{\text{mid}} \leftarrow \frac{\alpha_{s_i^{\text{mid}}}}{\alpha_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} - \sigma_{s_i^{\text{mid}}} \left( e^{\frac{h_i}{2}} - 1 \right) \epsilon_{\phi \times}(\tilde{\mathbf{x}}_{t_{i-1}}, t_{i-1})$ 
6:    $\tilde{\mathbf{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} - \sigma_{t_i} \left( e^{h_i} - 1 \right) \epsilon_{\phi \times}(\mathbf{x}_i^{\text{mid}}, s_i^{\text{mid}})$ 
7: end for
8: return  $\tilde{\mathbf{x}}_{t_M}$ 

```

1 Error

Consider DPM-Solver-2: we use an $\mathcal{O}(h)$ -scale approximation of the first-order total λ -derivative, but we can reach global $\mathcal{O}(h^2)$ accuracy.

$\gamma \in (0, 1]$ affects only the error constant.

2 Timestep

As mentioned above, in practice, uniform spacing in log-SNR time λ_t achieves high-quality samples even with very few steps.

$$\lambda_{t_i} = \lambda_T + \frac{i}{M}(\lambda_0 - \lambda_T), \quad i = 0, 1, \dots, M.$$

Since all updates are expressed in t , we have to convert λ_{t_i} to t_i using the inverse mapping $t_\lambda(\lambda)$.

- **EDM Power Schedule** (Karras et al., 2022) : uniform spacing in $\sigma^{1/\rho}$ -time

$$\sigma_i = \left[\sigma_0^{1/\rho} + \frac{i}{M} (\sigma_T^{1/\rho} - \sigma_0^{1/\rho}) \right]^\rho$$

- **Data-scale-aware Angle Schedule** (Sabour et al., 2024) : uniform spacing in θ -time
For the VE SDE $x_t = x_0 + \sigma_t \epsilon$, $x_0 \sim N(0, c^2 I)$,

$$\arctan\left(\frac{\sigma_i}{c}\right) = \arctan\left(\frac{\sigma_0}{c}\right) + \frac{i}{M} \left(\arctan\left(\frac{\sigma_T}{c}\right) - \arctan\left(\frac{\sigma_0}{c}\right) \right)$$

(Author's note: it is the optimal grid for Euler/DDIM sampler.)

- **log-SNR Schedule** is the most natural choice for DPM-Solver, since it makes us easily handle the integrand (including high-order derivatives) in the exact solution of the PF-ODE.

In the v -prediction form, the PF-ODE has no linear part. More generally, in log-SNR time λ , write the PF-ODE as

$$\frac{d\hat{x}(\lambda)}{d\lambda} = \underbrace{L(\lambda)\hat{x}(\lambda) + \mathbf{N}(\hat{x}(\lambda), \lambda)}_{\mathbf{F}(\hat{x}(\lambda), \lambda)}.$$

For v -prediction, $L(\lambda) \equiv 0$, whereas this is not generally the case for other parameterizations.

Consider the RK2 Heun method for this PF-ODE. For $\{\lambda_i\}_{i=0}^M$,

$$\begin{aligned}\hat{x}_i^{\text{pred}} &= \hat{x}_{i-1} + h \mathbf{F}(\hat{x}_{i-1}, \lambda_{i-1}), \\ \hat{x}_i &= \hat{x}_{i-1} + h \frac{\mathbf{F}(\hat{x}_{i-1}, \lambda_{i-1}) + \mathbf{F}(\hat{x}_i^{\text{pred}}, \lambda_i)}{2}.\end{aligned}$$

As in exponential Euler, introduce

$$\mathcal{E} := \int_{\lambda_{i-1}}^{\lambda_i} L(\tau) d\tau.$$

Then the Exponential-Heun update defines by

$$\begin{aligned}\hat{x}_{\lambda_i}^{\text{pred}} &= e^{\mathcal{E}} \hat{x}_{\lambda_{i-1}} + h c_1(\mathcal{E}) \mathbf{N}(\hat{x}_{\lambda_{i-1}}, \lambda_{i-1}), \\ \hat{x}_{\lambda_i} &= e^{\mathcal{E}} \hat{x}_{\lambda_{i-1}} + h c_1(\mathcal{E}) \mathbf{N}(\hat{x}_{\lambda_{i-1}}, \lambda_{i-1}) \\ &\quad + h c_2(\mathcal{E}) \left[\mathbf{N}(\hat{x}_{\lambda_i}^{\text{pred}}, \lambda_i) - \mathbf{N}(\hat{x}_{\lambda_{i-1}}, \lambda_{i-1}) \right].\end{aligned}$$

The continuous coefficient functions are $c_1(\mathcal{E}) = \frac{e^{\mathcal{E}} - 1}{\mathcal{E}}$, $c_2(\mathcal{E}) = \frac{e^{\mathcal{E}} - 1 - \mathcal{E}}{\mathcal{E}^2}$. Note that when $L \equiv 0$, i.e., $\mathcal{E} = 0$, this reduces to the standard Heun method.

Heun Updates equivalent to DPM-Solver-2

- **v -prediction:**

$$\frac{d\hat{x}_\lambda}{d\lambda} = \underbrace{\alpha'_\lambda \hat{x}^*(\hat{x}_\lambda, \lambda) + \sigma'_\lambda \hat{\epsilon}^*(\hat{x}_\lambda, \lambda)}_{= \hat{v}_{\phi \times}(\hat{x}_\lambda, \lambda) =: N(\hat{x}_\lambda, \lambda)}.$$

Hence, the Heun update for this equation is identical to DPM-Solver-2.

- **ϵ -, x -, and s -prediction:**

$$\frac{d\hat{x}_\lambda}{d\lambda} = \underbrace{\frac{\alpha'_\lambda}{\alpha_\lambda}}_{=: L(\lambda)} \hat{x}_\lambda + \underbrace{\left(-\sigma_\lambda \hat{\epsilon}_{\phi \times}(\hat{x}_\lambda, \lambda)\right)}_{=: N(\hat{x}_\lambda, \lambda)}.$$

Hence, the Exponential-Heun update for this equation is identical to DPM-Solver-2 (up to $\mathcal{O}(h^2)$).

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models
- 3 DEIS : Diffusion Exponential Integrator Sampler
- 4 DPM-Solver : Diffusion Probabilistic Model Solver
- 5 DPM-Solver++**
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps

The limitations of high-order solvers become important in guidance sampling, especially with **classifier-free guidance (CFG)**.

Since

$$\epsilon_{\text{CFG}}(x_t, t, c) = \epsilon_{\phi \times}(x_t, t, \emptyset) + w [\epsilon_{\phi \times}(x_t, t, c) - \epsilon_{\phi \times}(x_t, t, \emptyset)],$$

where the guidance scale $w \geq 0$ is freely chosen, both the model output and its derivatives can be amplified, making the trajectory unstable.

DPM-Solver++

- 1 Use **x -prediction** instead of ϵ -prediction.

This prevents excessive amplification caused by a large guidance scale.

- 2 Apply thresholding methods, e.g., dynamic thresholding.

Clip/rescale large values appropriately to keep the sample values within the range of the training data.

- **x -prediction**

Using the known relation, the exact solution $\tilde{\Psi}_{s \rightarrow t}(x_s)$ of the empirical PF-ODE is

$$\tilde{\Psi}_{s \rightarrow t}(x_s) = \frac{\sigma_t}{\sigma_s} x_s + \sigma_t \int_{\lambda_s}^{\lambda_t} e^{\lambda} \hat{x}_{\phi \times}(\hat{x}_{\lambda}, \lambda) d\lambda,$$

from

$$\tilde{\Psi}_{s \rightarrow t}(x_s) = \frac{\alpha_t}{\alpha_s} x_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_{\phi \times}(\hat{x}_{\lambda}, \lambda) d\lambda.$$

- **Solver**

- ① Higher-Order Single-Step Solver: **DPM-Solver++(2S)**
- ② Multistep (Two-Step) Solver: **DPM-Solver++(2M)**

x -prediction + DPM-Solver-2

Algorithm 6 DPM-Solver++(2S): a midpoint special case.

Input: initial value $\mathbf{x}_T$, time steps $\{t_i\}_{i=0}^M$, data-prediction model $\hat{\mathbf{x}}_{\phi^\times}$

- 1: $\tilde{\mathbf{x}}_{t_0} \leftarrow \mathbf{x}_T$; $\lambda_{t_i} \leftarrow \log(\alpha_{t_i}/\sigma_{t_i})$ ▷ log-SNR at the grid
 - 2: $\hat{\mathbf{x}}_0 \leftarrow \hat{\mathbf{x}}_{\phi^\times}(\tilde{\mathbf{x}}_{t_0}, t_0)$ ▷ cache at start
 - 3: **for** $i \leftarrow 1$ **to** M **do**
 - 4: $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$; $s_i^{\text{mid}} \leftarrow t_\lambda\left(\frac{\lambda_{t_{i-1}} + \lambda_{t_i}}{2}\right)$
 - 5: $\mathbf{u}_i \leftarrow \frac{\sigma_{s_i^{\text{mid}}}}{\sigma_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} + \alpha_{s_i^{\text{mid}}}(1 - e^{-h_i/2}) \hat{\mathbf{x}}_{i-1}$ ▷ forecast to midpoint
 - 6: $\mathbf{D}_i^{\text{mid}} \leftarrow \hat{\mathbf{x}}_{\phi^\times}(\mathbf{u}_i, s_i^{\text{mid}})$ ▷ one new model call at the midpoint
 - 7: $\tilde{\mathbf{x}}_{t_i} \leftarrow \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} - \alpha_{t_i}(e^{-h_i} - 1)\mathbf{D}_i^{\text{mid}}$
 - 8: $\hat{\mathbf{x}}_i \leftarrow \hat{\mathbf{x}}_{\phi^\times}(\tilde{\mathbf{x}}_{t_i}, t_i)$ ▷ cache for next step
 - 9: **end for**
 - 10: **return** $\tilde{\mathbf{x}}_{t_M}$
-

x -prediction + **DEIS-2**, but we can simplify the update rule.

Originally, we approximate the model evaluation on the interval $[\lambda_{i-1}, \lambda_i]$ by an affine model (using two points):

$$\hat{x}_{\phi \times}(\hat{x}_\lambda, \lambda) \approx \mathbf{a}_0 + \mathbf{a}_1(\lambda - \lambda_{i-1}), \quad \lambda \in [\lambda_{i-1}, \lambda_i],$$

where $\mathbf{a}_0 = \hat{x}_{\phi \times}(\tilde{x}_{t_{i-1}}, t_{i-1})$, $\mathbf{a}_1 = \frac{\hat{x}_{\phi \times}(\tilde{x}_{t_{i-1}}, t_{i-1}) - \hat{x}_{\phi \times}(\tilde{x}_{t_{i-2}}, t_{i-2})}{h_{i-1}}$.

Then,

$$\begin{aligned} \sigma_{t_i} \int_{\lambda_{i-1}}^{\lambda_i} e^\lambda \hat{x}_{\phi \times}(\hat{x}_\lambda, \lambda) d\lambda &\approx \left(\sigma_{t_i} \int_{\lambda_{i-1}}^{\lambda_i} e^\lambda d\lambda \right) \mathbf{a}_0 \\ &\quad + \left(\sigma_{t_i} \int_{\lambda_{i-1}}^{\lambda_i} e^\lambda (\lambda - \lambda_{i-1}) d\lambda \right) \mathbf{a}_1 \\ &= \alpha_{t_i} (1 - e^{-h_i}) \mathbf{a}_0 + \alpha_{t_i} (h_i - 1 + e^{-h_i}) \mathbf{a}_1 \\ &= \alpha_{t_i} (1 - e^{-h_i}) (\mathbf{a}_0 + \beta(h_i) \mathbf{a}_1), \end{aligned}$$

where $\beta(h) = \frac{h-1+e^{-h}}{1-e^{-h}}$.

$$\therefore \tilde{x}_{t_i} = \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}} \tilde{x}_{t_{i-1}} + \alpha_{t_i} (1 - e^{-h_i}) \mathbf{D}_i, \quad \mathbf{D}_i = \mathbf{a}_0 + \beta(h_i) \mathbf{a}_1.$$

For the step ratio $r_i = \frac{h_i}{h_{i-1}}$,

$$\begin{aligned} \mathbf{D}_i &= \hat{x}_{i-1} + \frac{\beta(h_i)}{h_{i-1}} (\hat{x}_{i-1} - \hat{x}_{i-2}) \\ &= \underbrace{\left(1 + \frac{r_i}{2}\right) \hat{x}_{i-1} - \frac{r_i}{2} \hat{x}_{i-2}}_{\mathbf{D}_i^{\text{sim}}} + \underbrace{\left(\frac{\beta(h_i)}{h_{i-1}} - \frac{r_i}{2}\right) (\hat{x}_{i-1} - \hat{x}_{i-2})}_{\mathcal{O}(h_i^2)}, \end{aligned}$$

since $\beta(h) = \frac{h}{2} + \mathcal{O}(h^2)$.

$$\therefore \tilde{x}_{t_i} = \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}} \tilde{x}_{t_{i-1}} + \alpha_{t_i} (1 - e^{-h_i}) \mathbf{D}_i^{\text{sim}}.$$

Algorithm 7 DPM-Solver++(2M).

Input: initial value $\mathbf{x}_T$, time steps $\{t_i\}_{i=0}^M$, model $\hat{\mathbf{x}}_{\phi^\times}$

- 1: $\tilde{\mathbf{x}}_{t_0} \leftarrow \mathbf{x}_T$; $\lambda_{t_i} \leftarrow \log(\alpha_{t_i}/\sigma_{t_i})$; $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$
2: $\hat{\mathbf{x}}_0 \leftarrow \hat{\mathbf{x}}_{\phi^\times}(\tilde{\mathbf{x}}_{t_0}, t_0)$ ▷ cache at start

Case I. Warm start ($i = 1$) with one anchor (DDIM in x-pred.)

- 3: $\tilde{\mathbf{x}}_{t_1} \leftarrow \frac{\sigma_{t_1}}{\sigma_{t_0}} \tilde{\mathbf{x}}_{t_0} - \alpha_{t_1}(e^{-h_1} - 1) \hat{\mathbf{x}}_0$
4: $\hat{\mathbf{x}}_1 \leftarrow \hat{\mathbf{x}}_{\phi^\times}(\tilde{\mathbf{x}}_{t_1}, t_1)$ ▷ One model call & cache

Case II. Using two history cached anchors (multistep)

- 5: **for** $i \leftarrow 2$ **to** M **do**
6: $r_i \leftarrow h_i/h_{i-1}$ ▷ step ratio
7: $\mathbf{D}_i^{\text{sim}} \leftarrow \left(1 + \frac{1}{2}r_i\right)\hat{\mathbf{x}}_{i-1} - \frac{1}{2}r_i\hat{\mathbf{x}}_{i-2}$
8: $\tilde{\mathbf{x}}_{t_i} \leftarrow \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} + \alpha_{t_i}(1 - e^{-h_i}) \mathbf{D}_i^{\text{sim}}$
9: $\hat{\mathbf{x}}_i \leftarrow \hat{\mathbf{x}}_{\phi^\times}(\tilde{\mathbf{x}}_{t_i}, t_i)$ ▷ One model call & cache
10: **end for**
11: **return** $\tilde{\mathbf{x}}_{t_M}$
-

- Warmup step
- one-call-per-step update

- 1 Prologue: Advanced Solvers for Diffusion Models
- 2 DDIM : Denoising Diffusion Implicit Models
- 3 DEIS : Diffusion Exponential Integrator Sampler
- 4 DPM-Solver : Diffusion Probabilistic Model Solver
- 5 DPM-Solver++
- 6 ParaDiGMS: Parallel Diffusion Generation with Multiple Steps**

Consider the PF-ODE in the v -prediction form:

$$\frac{dx(t)}{dt} = v_{\phi \times} (x(t), t), \quad t \in [0, T], \quad x(T) \sim \mathcal{N}(0, I_d).$$

Given any candidate trajectory $\{y(\tau)\}_{\tau \in [0, T]}$, we define the operator $\mathcal{L}$ by

$$(\mathcal{L}[y(\cdot)])(t) = y(T) + \int_T^t v_{\phi \times} (y(\tau), \tau) d\tau.$$

A true solution $x^*(\cdot)$ of the PF-ODE is the only one that remains unchanged under this mapping:

$$x^*(t) = \mathcal{L}[x^*(\cdot)](t).$$

If we want to find the fixed point of the mapping, we use the Picard iteration:

$$\begin{aligned}x^{(k+1)}(t) &= \mathcal{L}[x^{(k)}(\cdot)](t) \\&= x^{(k)}(T) + \int_T^t v_{\phi \times}(x^{(k)}(\tau), \tau)\end{aligned}$$

By discretization, for a uniform grid $t_j = T - j\Delta t$, $j = 0, 1, \dots, M$, we obtain

$$x_j^{(k+1)} = x_0^{(k)} - \Delta t \sum_{i=0}^{j-1} v_{\phi \times}(x_i^{(k)}, t_i), \quad j = 1, \dots, M.$$

Here, the evaluations of $v_{\phi \times}(x_i^{(k)}, t_i)$ can be performed in parallel, not sequentially.

Fix a window length p and a left index ℓ . During iteration k :

1 Parallel Drift Evaluation on the Window

Compute

$$v_{\phi \times} \left(x_{\ell+i}^{(k)}, t_{\ell+i} \right), \quad i = 0, 1, \dots, p-1.$$

2 Left-Anchored Cumulative Updates

Update

$$x_{\ell+j+1}^{(k+1)} = x_{\ell}^{(k)} - \Delta t \sum_{i=0}^j v_{\phi \times} \left(x_{\ell+i}^{(k)}, t_{\ell+i} \right), \quad j = 0, 1, \dots, p-1.$$

3 Progress Control and Window Advance

For a fixed tolerance $\text{tol}_{\ell+j}$,

$$\text{error}_j := \left\| x_{\ell+j}^{(k+1)} - x_{\ell+j}^{(k)} \right\|^2, \quad j = 1, \dots, p-1$$

$$\text{stride} := \min \left(\{j \geq 1 : \text{error}_j > \text{tol}_{\ell+j}\} \cup \{p\} \right)$$

$$\ell \leftarrow \ell + \text{stride}.$$

(In the next step, initialize the new node using the boundary value.)

Algorithm 8 ParaDiGMS with Sliding Windows

Input: Drift $\mathbf{v}_{\phi \times}(\mathbf{x}, t)$; $\{t_j\}_{j=0}^M$; window length p ; $\{\text{tol}_j\}_{j=1}^M$

Output: Approximate trajectory $\{\mathbf{x}_j^{(k)}\}_{j=0}^M$ with $\mathbf{x}_M^{(k)}$ at $t = 0$

```
1:  $k \leftarrow 0, \ell \leftarrow 0$ 
2: Sample  $\mathbf{x}_0^{(0)} \sim p_{\text{prior}}$ ; set  $\mathbf{x}_j^{(0)} \leftarrow \mathbf{x}_0^{(0)}$  for  $j = 1, \dots, \min(p, M)$ 
    $\triangleright$  constant extrapolation
3: while  $\ell < M$  do
4:    $J \leftarrow \min(p, M - \ell)$ 
    $\triangleright$  current window length
5:   Step 1: Parallel
6:   For  $i = 0, \dots, J - 1$ :  $g_i \leftarrow \mathbf{v}_{\phi \times}(\mathbf{x}_{\ell+i}^{(k)}, t_{\ell+i})$ 
    $\triangleright$  drifts from previous iterate (Picard freezing)
7:   Compute prefix sums  $S_j \leftarrow \sum_{i=0}^j g_i$  for  $j = 0, \dots, J - 1$ 
    $\triangleright$  scan over windowed drifts
8:   Step 2: Cumulative Updates
9:   For  $j = 0, \dots, J - 1$ :  $\mathbf{x}_{\ell+j+1}^{(k+1)} \leftarrow \mathbf{x}_{\ell}^{(k)} - \Delta t S_j$ 
    $\triangleright$  left-anchored update; cf. Equation (9.7.4)
10:  Step 3: Progress Control and Window Advance
11:  For  $j = 1, \dots, J - 1$ :  $\text{error}_j \leftarrow \|\mathbf{x}_{\ell+j}^{(k+1)} - \mathbf{x}_{\ell+j}^{(k)}\|^2$ 
    $\triangleright$  pointwise Picard change
12:   $\text{stride} \leftarrow \min(\{j \in \{1, \dots, J - 1\} : \text{error}_j > \text{tol}_{\ell+j}\} \cup \{J\})$ 
13:  Initialize New Nodes
14:  For  $r = 1, \dots, \text{stride}$ :  $\mathbf{x}_{\ell+J+r}^{(k+1)} \leftarrow \mathbf{x}_{\ell+J}^{(k+1)}$ 
    $\triangleright$  constant extrapolation into newly exposed indices
15:   $\ell \leftarrow \ell + \text{stride}; k \leftarrow k + 1$ 
16: end while
17: return  $\{\mathbf{x}_j^{(k)}\}_{j=0}^M$ 
```

- DDIM scheme ($p = 1$)
- Compatible with Higher-Order Solvers

✂ It allows for fast sampling through parallel evaluation, but doesn't aim to improve the accuracy of the solution.

- [1] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine.
Elucidating the design space of diffusion-based generative models.
[Advances in neural information processing systems](#), 35:26565–26577, 2022.
- [2] Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon.
The principles of diffusion models.
[arXiv preprint arXiv:2510.21890](#), 2025.
- [3] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis.
Align your steps: Optimizing sampling schedules in diffusion models.
[arXiv preprint arXiv:2404.14507](#), 2024.
- [4] Jiaming Song, Chenlin Meng, and Stefano Ermon.
Denoising diffusion implicit models.
[arXiv preprint arXiv:2010.02502](#), 2020.