ENHANCED DIFFUSION SAMPLING VIA EXTRAPOLATION WITH MULTIPLE ODE SOLUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion probabilistic models (DPMs), while effective in generating high-quality samples, often suffer from high computational costs due to the iterative sampling process. To address this, we propose an enhanced ODE-based sampling method for DPMs inspired by Richardson extrapolation, which has been shown to reduce numerical error and improve convergence rates. Our method, termed RX-DPM, utilizes numerical solutions obtained over multiple denoising steps, leveraging the multiple ODE solutions to extrapolate the denoised prediction in DPMs. This significantly enhances the accuracy of estimations for the final sample while preserving the number of function evaluations (NFEs). Unlike standard Richardson extrapolation, which assumes uniform discretization of the time grid, we have developed a more general formulation tailored to arbitrary time step scheduling, guided by the local truncation error derived from a baseline sampling method. The simplicity of our approach facilitates accurate estimation of numerical solutions without additional computational overhead, and allows for seamless and convenient integration into various DPMs and solvers. Additionally, RX-DPM provides explicit error estimates, effectively illustrating the faster convergence achieved as the order of the leading error term increases. Through a series of experiments, we demonstrate that the proposed method effectively enhances the quality of generated samples without requiring additional sampling iterations.

1 INTRODUCTION

Diffusion probabilistic models (DPMs) have emerged as a powerful framework for generating high-quality samples in a wide range of applications and domains for images (Ho et al., 2020; Song et al., 2021b; Dhariwal & Nichol, 2021; Rombach et al., 2022), videos (Ho et al., 2022; Singer et al., 2022; Zhou et al., 2022; Wang et al., 2023), 3D shapes (Zeng et al., 2022), *etc*. While DPMs demonstrate impressive performance in data fidelity and diversity, they also have limitations, particularly their computational inefficiency due to the sequential nature of sampling. Addressing this issue is crucial for enhancing the practicality of DPMs in real-world scenarios, where computational resources are often limited.

The generation process of DPMs can be formulated as a problem of finding solutions to SDEs or ODEs (Song et al., 2021b), where the truncation errors of the numerical solutions are highly correlated to the quality of the generated samples. To enhance the quality of these samples, it is essential to reduce truncation errors, which can be achieved by adopting advanced solvers or numerical techniques that improve the order of accuracy of numerical estimations. In this context, we aim to lower truncation errors by applying numerical extrapolation to existing sampling methods for DPMs, utilizing estimations over different numbers of steps. The key ingredient of the proposed method is Richardson extrapolation, which has been proven to be reliable and is widely used, particularly in the mathematical modeling of physical problems, *e.g.*, fluid dynamics and heat transfer, which demand high computational resources. As a well-established method, numerous variants have be proposed, and its efficacy and application strategies for different problems have been investigated (Richards, 1997; Botchev & Verwer, 2009; Zlatev et al., 2010), but for DPMs. The method uses a simple linear combination of multiple numerical estimates in progressively finer resolutions of a grid to approximate the ideal solution, which is expected to be reached by the limit of the series of estimates.

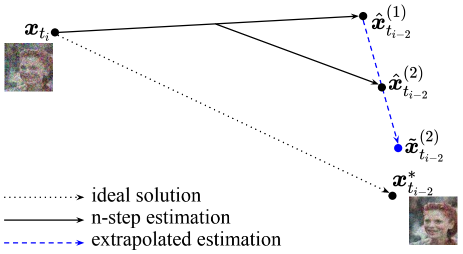

$\boldsymbol{x}_{t_i}$
$\hat{\boldsymbol{x}}^{(1)}_{t_{i-2}}$
$\hat{\boldsymbol{x}}^{(2)}_{t_{i-2}}$
$\tilde{\boldsymbol{x}}^{(2)}_{t_{i-2}}$
$\boldsymbol{x}^*_{t_{i-2}}$
ideal solution
n-step estimation
extrapolated estimation

Figure 1: Application of the proposed extrapolation on two denoising steps ($k = 2$) with time steps of $[t_i, t_{i-1}, t_{i-2}]$. $\hat{\boldsymbol{x}}^{(n)}_{t_{i-2}}$ denotes that n steps are used by the baseline sampler within the same interval. $\tilde{\boldsymbol{x}}^{(2)}_{t_{i-2}}$ represents the extrapolated estimation using two ODE solutions at t_{i-2}, $\hat{\boldsymbol{x}}^{(1)}_{t_{i-2}}$ and $\hat{\boldsymbol{x}}^{(2)}_{t_{i-2}}$.

We propose the extrapolation algorithm that is applied repeatedly every k denoising steps of an ODE-based sampling method to improve the accuracy of intermediate denoising steps. This is achieved by utilizing an additional ODE solution which is computed from a single step estimation over an interval of k time steps. Figure 1 illustrates this concept with $k = 2$ on time step $[t_i, t_{i-1}, t_{i-2}]$, which is an unit block where an extrapolation is performed. Two ODE solutions—single-step and two-step estimations at t_{i-2} from t_i—can be leveraged to achieve approximations closer to the ideal solution $\boldsymbol{x}^*_{t_{i-2}}$, which is unknown. The standard Richardson extrapolation is limited to employing a uniform discretization over a time grid. However, constructing denoising time steps with uniform discretizations might be suboptimal for DPMs; a smaller time interval near the clean sample is often much more beneficial, depending on datasets and DPM backbones (Karras et al., 2022; Song et al., 2021a) despite using the same number of steps. Considering such characteristics of DPMs, in order not to be restricted from existing benefits, we introduce a variant of the Richardson extrapolation algorithm specifically tailored for DPMs, applicable to arbitrary discretizations of time steps. We observe that this non-uniform discretization approach yields better performance than conventional methods.

Although there exist other methods applying extrapolation techniques to diffusion models, their usages of extrapolation are somewhat different from ours. For example, (Zhang et al., 2024; 2023) utilize estimations from earlier steps to improve the estimation of the time step, t_i, whereas our approach adopts two denoised estimations at the same time step, t_i, to enhance their accuracy at t_i. In addition, unlike other extrapolation methods (Zhang et al., 2024; 2023), the main building block of our approach, Richardson extrapolation, is proven to enhance numerical accuracy and provides an explicit estimate of the error, which allows for a clear understanding of the convergence behavior. Furthermore, the implementation of our algorithm is simple and cost-effective because it requires no additional network evaluations and insignificant computational overhead to perform the extrapolation. We refer to the proposed sampling algorithm as RX-DPM.

Our main contributions are summarized below:

- We introduce an improved diffusion sampler, RX-DPM, motivated by Richardson extrapolation, which effectively increases the order accuracy of the existing ODE-based samplers. We develop an algorithm for arbitrary discretization, specifically tailored for DPMs.
- We systematically develop an algorithm on how to leverage Richardson extrapolation to general DPM solvers with arbitrary time step scheduling starting from the derivation of a truncation error formula of the Euler method on a non-uniform grid. We also provide details on how to implement it across various diffusion samplers without incurring additional NFEs.
- Our experiments across various well-known baselines demonstrate that RX-DPM exhibits strong generalization performance and high practicality, regardless of ODE designs, architectures and base samplers.

The rest of the paper is organized as follows. Section 2 discusses closely related papers and Section 3 provides a brief overview of the basics of DPMs and Richardson extrapolation. Following that, Section 4 describes the development process of the proposed extrapolation algorithm with the DPM context. We demonstrate experimental results and their analyses in Section 5, and conclude our paper in Section 6.

2 RELATED WORK

There exists a substantial body of research that seeks to reduce the computational burden of DPMs while maintaining their performance. One approach in this direction involves exploring alternative modeling strategies for the reverse process. For example, the networks in Salimans & Ho (2022);

Song et al. (2023); Kim et al. (2024) learn alternative objectives, the outputs obtained by iterative inferences of the pretrained networks or teacher models, through knowledge distillation. On the other hand, Bao et al. (2022b;a) models more accurate reverse distribution explicitly by incorporating optimal covariance, while Xiao et al. (2022); Kang et al. (2024) implicitly estimate precise reverse distribution by utilizing GAN components.

Another line of research for fast sampling interprets generation process of diffusion models as solving an ODE or SDE (Song et al., 2021b; Karras et al., 2022). For instance, DDIM (Song et al., 2021a) proposes to skip intermediate time steps, which is equivalent to solving an ODE using the Euler method with a large step size. To further improve sampling quality, a large volume of research (Karras et al., 2022; Dockhorn et al., 2022; Liu et al., 2022; Zhang & Chen, 2023; Lu et al., 2022; 2023) applies classical higher-order solvers or tailors them for diffusion models. Specifically, Karras et al. (2022) adopts the second-order Heun's method and Dockhorn et al. (2022) applies the second-order Taylor expansion. In addtion, Liu et al. (2022) proposes a pseudo-numerical solver, which approximates classical higher-order numerical methods such as Runge-Kutta (Süli & Mayers, 2003), and Zhang & Chen (2023) refines the coefficients of the high-order polynomials.

On top of the methods that apply ODE solvers, Zhang et al. (2023; 2024) introduce extrapolation to further improve the sample quality. To be specific, Zhang et al. (2023) linearly extrapolates the previous and current predictions of the solution at $t = 0$—the solution on the image manifold—while Zhang et al. (2024) uses a linear combination of recent $r + 1$ gradients to compute integration. The clear distinction of our approach from these methods is that, while they adopt score (noise) predictions over multiple time steps for extrapolation, we utilize multiple denoised outputs obtained at the same time step.

3 PRELIMINARIES

3.1 DIFFUSION PROBABILISTIC MODELS AS SOLVING ODE

For $\boldsymbol{p}_0 = \boldsymbol{p}_{\text{data}}$ and $\boldsymbol{x} \in \mathbb{R}^d$, Karras et al. (2022) defines a marginal distribution at t as

$$\boldsymbol{p}_t(\boldsymbol{x}) = s(t)^{-d}\boldsymbol{p}(\boldsymbol{x}/s(t); \sigma(t)), \tag{1}$$

where $\boldsymbol{p}(\boldsymbol{x}; \sigma) = \boldsymbol{p}_{\text{data}} * \mathcal{N}(\boldsymbol{0}, \sigma(t)^2\boldsymbol{I})$, and $s(t)$ and $\sigma(t)$ are non-negative functions satisfying $s(0) = 1$, $\sigma(0) = 0$, and $\lim_{t\to\infty} \frac{\sigma(t)}{s(t)} = \infty$. The probability flow ODE,

$$d\boldsymbol{x} = [\dot{s}(t)/s(t) - s(t)^2\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}/s(t); \sigma(t))]dt, \quad \boldsymbol{x}(T) \sim \boldsymbol{p}_T(\boldsymbol{x}), \tag{2}$$

matches the marginal distribution. In Karras et al. (2022), the specific choices $s(t) = 1$ and $\sigma(t) = t$ are adopted and Equation (2) is reduced to the following equation:

$$d\boldsymbol{x} = -t\nabla_{\boldsymbol{x}} \log \boldsymbol{p}(\boldsymbol{x}; t)dt, \quad \boldsymbol{x}(T) \sim \boldsymbol{p}_T(\boldsymbol{x}). \tag{3}$$

Diffusion models now learn the score function $\nabla_{\boldsymbol{x}} \log \boldsymbol{p}(\boldsymbol{x}; t)$, which is the only unknown component in the equation. For sufficiently large T, the marginal distribution $\boldsymbol{p}_T(\boldsymbol{x})$ can be approximated by $\mathcal{N}(\boldsymbol{x}; \boldsymbol{0}, T^2\boldsymbol{I})$ and the generation process is equivalent to solving for $\boldsymbol{x}(0)$ using Equation (3) with the boundary condition, $\boldsymbol{x}(T) \sim \mathcal{N}(\boldsymbol{0}, T^2\boldsymbol{I})$. Since the analytic solution of Equation (3) cannot be expressed in a closed form, numerical methods are used to solve the ODE. Given the time step scheduling, $0 = t_0 < t_1 < \ldots < t_N = T$, the solution is given by

$$\boldsymbol{x}(0) = \boldsymbol{x}(T) + \int_T^0 -t\nabla_{\boldsymbol{x}} \log \boldsymbol{p}(\boldsymbol{x}(t); t)dt \tag{4}$$

$$= \boldsymbol{x}(T) + \sum_{i=N}^{1} \int_{t_i}^{t_{i-1}} -t\nabla_{\boldsymbol{x}} \log \boldsymbol{p}(\boldsymbol{x}(t); t)dt, \tag{5}$$

where each integration from t_i to t_{i-1} can be approximated by ODE solvers such as the Euler or Heun's method.

3.2 RICHARDSON EXTRAPOLATION

Given a boundary condition $x(T) = x_T$ in the 1-dimensional case of Equation (3), let the exact solution at $t = 0$ and the numerical solution at $t = 0$ obtained by a step size h $(0 < h < 1)$ be V^* and $V(h)$, respectively. If $V^* = \lim_{h\to 0} V(h)$ and the order of truncation error is known, the Richardson extrapolation (Richardson, 1911) can be used to identify a faster converging sequence, $\tilde{V}(h)$. For instance, $V(h)$ with a truncation error in the order of $O(h^p)$ is expressed by the following equation:

$$V^* = V(h) + ch^p + O(h^q), \tag{6}$$

for $0 < p < q$ and $\exists c \neq 0$. Then, for a fixed constant $k > 1$,

$$V^* = V(h/k) + \frac{c}{k^p} h^p + O(h^q). \tag{7}$$

From Equations (6) and (7), we obtain

$$(k^p - 1)V^* = k^p V(h/k) - V(h) + O(h^q), \tag{8}$$

and equivalently

$$\tilde{V}(h, k) = \frac{k^p V(h/k) - V(h)}{k^p - 1}. \tag{9}$$

This solution has a truncation error of $O(h^q)$, which is asymptotically smaller than $O(h^p)$.

4 RX-DPM

Before discussing the proposed method, we first outline the algorithmic development process for the most simplified problem and then explore an extension to a general DPM solver.

4.1 TRUNCATION ERROR OF EULER METHOD ON NON-UNIFORM GRID

We now derive the truncation error formula for the Euler method on a non-uniform grid, based on the local truncation error, the error caused by one-step approximation. For intuitive clarity, we consider a one-dimensional ODE of the form

$$dx = f(x, t)dt,$$

where f is a smooth function. Let the numerical solution be obtained using the Euler method with the discretization points $[t_i, t_{i-1}, \dots, t_{i-k}]$ in the time-reversed direction given the boundary condition $x(t_i) = x_{t_i}$. From now on, we denote $\hat{x}^{(n)}_{t_j}$ as the numerical solution at t_j obtained by n iterations and $x^*_{t_j}$ as the exact solution at t_j. For $h = t_i - t_{i-k}$, and $\lambda_j = \frac{1}{h}(t_{i-j+1} - t_{i-j})$, $j = 1, \dots, k$, the local truncation error of the one-step Euler method obtained by the Taylor expansion is given by

$$\hat{x}^{(1)}_{t_{i-1}} = x_{t_i} - \lambda_1 h f(x_{t_i}; t_i) = x^*_{t_{i-1}} - \frac{1}{2} x''_{t_i} \lambda_1^2 h^2 + O(h^3). \tag{10}$$

Then, the truncation error of the two-step numerical solution is derived as

$$\begin{aligned}
\hat{x}^{(2)}_{t_{i-2}} &= \hat{x}^{(1)}_{t_{i-1}} - \lambda_2 h f(\hat{x}^{(1)}_{t_{i-1}}) && (11)\\
&= x^*_{t_{i-1}} - \frac{1}{2} x''_{t_i} \lambda_1^2 h^2 + O(h^3) - \lambda_2 h f(\hat{x}^{(1)}_{t_{i-1}}) && (12)\\
&= x^*_{t_{i-1}} - \lambda_2 h f(x^*_{t_{i-1}}) - \frac{1}{2} x''_{t_i} \lambda_1^2 h^2 + O(h^3) - \lambda_2 h f(\hat{x}^{(1)}_{t_{i-1}}) + \lambda_2 h f(x^*_{t_{i-1}}) && (13)\\
&= x^*_{t_{i-2}} - \frac{1}{2} x^{*\prime\prime}_{t_{i-1}} \lambda_2^2 h^2 - \frac{1}{2} x''_{t_i} \lambda_1^2 h^2 + O(h^3) \quad (\because f \text{ is smooth}) && (14)\\
&= x^*_{t_{i-2}} - \frac{1}{2} x''_{t_i} (\lambda_1^2 + \lambda_2^2) h^2 + O(h^3) \quad (\because f \text{ is smooth}). && (15)
\end{aligned}$$

Inductively, we can obtain the truncation error for the k-step solution as

$$\hat{x}^{(k)}_{t_{i-k}} = x^*_{t_{i-k}} - \frac{1}{2} x''_{t_i} \sum_{j=1}^{k} \lambda_j^2 h^2 + O(h^3), \tag{16}$$

which approximates $x^*_{t_{i-k}}$ with a truncation error of $O(h^2)$.

4.2 RX-EULER

We now describe RX-Euler, performing extrapolation every k steps on the Euler method. Extrapolation is executed as a linear combination of two different numerical solutions $\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}$ and $\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}}$ obtained by the Euler solver over 1 step on the grid $[t_i, t_{i-k}]$ and k steps on the grid $[t_i, t_{i-1}, \dots, t_{i-k}]$. To calculate coefficients for extrapolation, we use the truncation error derived in Section 4.1, which can be also applied to Equation (3) in Section 3.1, as the ideal score function can be considered smooth; its derivative is Lipschitz continuous, referring to the equation in Appendix B.3 of Karras et al. (2022). From Equations (10) and (16), we obtain the following expressions for $\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}$ and $\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}}$, respectively, for a constant c:

$$\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}} = \boldsymbol{x}^*_{t_{i-k}} - ch^2 + O(h^3) \quad \text{and} \tag{17}$$

$$\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}} = \boldsymbol{x}^*_{t_{i-k}} - c\sum_{j=1}^{k} \lambda_j^2 h^2 + O(h^3). \tag{18}$$

Then, by solving the linear system of Equations (17) and (18), we can approximate $\boldsymbol{x}^*_{t_{i-k}}$ through the following extrapolation:

$$\tilde{\boldsymbol{x}}^{(k)}_{t_{i-k}} = \frac{\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}} - \sum_{j=1}^{k} \lambda_j^2 \hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}}{1 - \sum_{j=1}^{k} \lambda_j^2}, \tag{19}$$

which involves a truncation error of $O(h^3)$, smaller than $O(h^2)$.

In the sampling process, we set the initial condition at the next denoising step, t_{i-k}, as $\tilde{\boldsymbol{x}}^{(k)}_{t_{i-1}}$ and repeatedly perform the proposed extrapolation technique every k steps. Because this approach provides provably more accurate solutions at every k steps, we can reduce error propagation and expect better quality of generated examples.

The proposed method is applicable to first-order methods in general, including DDIM (Song et al., 2021a), which is arguably the most widely used DPM sampler. In this context, we interpret DDIM as the Euler method applied to the following ODE:

$$d\boldsymbol{y} = \epsilon_\theta(\boldsymbol{x}(t), t)d\gamma, \tag{20}$$

where $\boldsymbol{y}(t) = \boldsymbol{x}(t)\sqrt{1+\gamma(t)^2}$ and $\gamma(t) = \sqrt{\frac{1-\alpha_t^2}{\alpha_t^2}}$ in the variance-preserving diffusion process (Song et al., 2021b), *i.e.*, $\boldsymbol{p}_t(\boldsymbol{x}|\boldsymbol{x}_0) = \mathcal{N}(\alpha_t \boldsymbol{x}_0, (1-\alpha_t^2)\boldsymbol{I})$. Thus, instead of using a time grid, we compute the $\lambda_{j(t)}$ values from Equation (19) in terms of the corresponding $\gamma(t)$, while the other procedures remain the same.

RX-Euler (RX-DDIM) does not require additional NFEs beyond the number of time steps, as the first prediction of every k-step-interval can be stored during the computation of $\hat{\boldsymbol{x}}^{(k)}$ and reused to obtain $\hat{\boldsymbol{x}}^{(1)}$. The only extra computation involves a linear combination of two estimates, which is negligible compared to the forward evaluations of DPMs.

4.3 RX-DPM WITH HIGHER-ORDER SOLVERS

We now present the algorithm for general ODE samplers of DPMs including high-order solvers. When the extrapolation spans k steps, the error form of the ODE solver satisfies

$$\boldsymbol{x}^*_{t_{i-k}} = \hat{\boldsymbol{x}}^{(1)}_{t_{i-k}} + ch^p + O(h^q) \tag{21}$$

for $0 < p < q$ and $c \neq 0$. Analogous to Equation (18), we suppose the following equation holds for $\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}}$ with the linear error accumulations assumption:

$$\boldsymbol{x}^*_{t_{i-k}} = \hat{\boldsymbol{x}}^{(k)}_{t_{i-k}} + c\sum_{j=1}^{k} \lambda_j^p h^p + O(h^q). \tag{22}$$

By solving linear system of Equations (21) and (22), the extrapolated solution is given by

$$\tilde{\boldsymbol{x}}^{(k)}_{t_{i-k}} = \frac{\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}} - \sum_{j=1}^{k} \lambda_j^p \hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}}{1 - \sum_{j=1}^{k} \lambda_j^p}, \tag{23}$$

which approximates $\boldsymbol{x}^*_{t_{i-k}}$ with a truncation error of $O(h^q)$, asymptotically smaller than $O(h^p)$. Although Equation (22) may not hold in general, this simplification is justified under the standard assumptions of Richardson extrapolation (see Appendix B). Algorithm 1 summarizes the procedure of the proposed method with a generic ODE solver under the assumption that N is a multitude of k for simplicity; it is simple to take care of the last few steps by either adjusting k for the remaining steps or skipping the extrapolation.

However, obtaining two estimations, $\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}$ and $\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}}$, throught naïve application of higher-order solvers requires additional network evaluations compared to the baseline sampling. For the baseline ODE solver that $\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}}$ is computed over k steps, if $\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}$ can be derived directly from $\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}}$ without requiring additional network evaluations, our method can be applied to high-order solvers without increasing NFEs. We will now illustrate how this is achieved using specific examples of higher-order ODE solvers. Common higher-order solvers often rely on interpolation-based techniques such as Runge-Kutta method (Süli & Mayers, 2003) and linear multistep methods (Timothy, 2017). Runge-Kutta family employs the evaluation on multiple intermediate points, and linear multistep methods leverage evaluations of the previous steps.

RX-Runge-Kutta We consider the second-order Runge-Kutta method with $k = 2$. A sequence of one-step estimates are given by

$$\hat{\boldsymbol{x}}^{(1)}_{t_{i-1}} = \boldsymbol{x}_{t_i} - (t_i - t_{i-1})(a_1 \mathbf{z}_i + a_2 \mathbf{z}_{i-\delta}) \quad \text{and} \tag{24}$$

$$\hat{\boldsymbol{x}}^{(2)}_{t_{i-2}} = \hat{\boldsymbol{x}}^{(1)}_{t_{i-1}} - (t_{i-1} - t_{i-2})(a_1 \mathbf{z}_{i-1} + a_2 \mathbf{z}_{i-1-\delta}). \tag{25}$$

where $\mathbf{z}_j = \epsilon_\theta(\boldsymbol{x}(t_j), t_j)$ for $t_{j-1} < t_{j-\delta} \leq t_j$. Then, we can express the single combined-step estimate at t_{i-2} as

$$\hat{\boldsymbol{x}}^{(1)}_{t_{i-2}} = \boldsymbol{x}_{t_i} - (t_i - t_{i-2})(a_1 \mathbf{z}_i + a_2 \mathbf{z}_{i-\delta'}), \tag{26}$$

where, since $\mathbf{z}_i$ is reused from the calculation of $\hat{\boldsymbol{x}}^{(1)}_{t_{i-1}}$, we only need to compute $\mathbf{z}_{i-\delta'}$, which is approximated as $\mathbf{z}_{i-1}$ or $\mathbf{z}_{i-1-\delta}$, depending on the proximity of its time step. This approach allows us to efficiently extrapolate the solutions without compromising the quality of the generated samples.

RX-Adam-Bashforth Suppose that, by the s-step Adams-Bashforth method, extrapolation is performed on a grid with an interval h every k steps. By the Adams-Bashforth method, we are given

$$\hat{\boldsymbol{x}}^{(k)}_{t_{i-k}} = \hat{\boldsymbol{x}}_{t_{i-k+1}} + h \sum_{j=0}^{s} b_j \epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i-k+j}}, t_{i-k+j}) \tag{27}$$

for predefined b_j's. We compute $\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}}$ for extrapolation as

$$\hat{\boldsymbol{x}}^{(1)}_{t_{i-k}} = \hat{\boldsymbol{x}}_{t_i} + kh \sum_{j=0}^{s} b_j \epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i-k+jk}}, t_{i-k+jk}) \tag{28}$$

which requires no additional NFE by storing the network evaluations.

4.4 ANALYSIS ON GLOBAL TRUNCATION ERRORS

We perform global truncation error analyses on Euler method and RX-Euler under the same NFEs. Assume we are solving ODE satisfying Lipshitz condition from $t = 0$ to $t = 1$ with N NFEs.

Euler Since Euler method requires a single network evaluation for each time step, the number of time steps is N. The local truncation error of Euler method on step size of $h = 1/N$ can be expressed as $ch^2 + O(h^3)$ and therefore the global truncation error is

$$(ch^2 + O(h^3)) \times N = \frac{c}{N} + O(N^{-2}). \tag{29}$$

Therefore, the dominating global truncation error term of Euler method is c/N.

Algorithm 1 Sampling of RX-DPM

```
Require: $\epsilon_\theta(\cdot), N, T = t_N > \ldots > t_0 = 0$
Input: $k$, $\Phi(\cdot)$ (ODE solver), $p$
$\boldsymbol{x}_T \sim \boldsymbol{p}_T(\boldsymbol{x})$
for $i = 1$ to $N$ do
    if $i \mod k == 1$ then
        $h \leftarrow t_{N-i+1} - t_{N-i-k+1}$
        $\hat{\boldsymbol{x}}^{(k)}_{t_{N-i+1}} \leftarrow \boldsymbol{x}_{t_{N-i+1}}$                # Initialization.
    end if
    $\lambda_i \leftarrow (t_{N-i+1} - t_{N-i})/h$
    $\hat{\boldsymbol{x}}^{(k)}_{t_{N-i}} \leftarrow \Phi(\hat{\boldsymbol{x}}^{(k)}_{t_{N-i+1}}, t_{N-i+1}, t_{N-i}; \epsilon_\theta(\cdot))$     # Store $\epsilon_\theta(t)$'s if neccessary.
    if $i \mod k == 0$ then
        $\hat{\boldsymbol{x}}^{(1)}_{t_{N-i}} \leftarrow \Phi(\boldsymbol{x}_{t_{N-i+h}}, t_{N-i} + h, t_{N-i}; \epsilon)$     # No NFE required.
        $\tilde{\boldsymbol{x}}^{(k)}_{t_{N-i}} \leftarrow \frac{\hat{\boldsymbol{x}}^{(k)}_{t_{N-i}} - \sum_{j=i}^{i+k-1} \lambda_j^p \hat{\boldsymbol{x}}^{(1)}_{t_{N-i}}}{1 - \sum_{j=i}^{i+k-1} \lambda_j^p}$     # Extrapolation.
        $\boldsymbol{x}_{t_{N-i}} \leftarrow \tilde{\boldsymbol{x}}^{(k)}_{t_{N-i}}$
    end if
end for
return $\boldsymbol{x}_{t_0}$
```

RX-Euler We consider RX-Euler where extrapolation occurs every k steps. Then for equal NFEs, N, as in Euler case, N/k extrapolations are performed. The local truncation error for RX-Euler on every k steps, which has the interval of $h = k/N$, can be expressed as $c'h^3 + O(h^4)$ and therefore the global truncation error is

$$(c'h^3 + O(h^4)) \times \frac{N}{k} = \frac{k^2 c'}{N^2} + O(N^{-3}). \tag{30}$$

Therefore, the dominating global truncation error term of RX-Euler is k^2c'/N^2. Given that RX-Euler exhibits a better global truncation error convergence rate compared to Euler method under the same NFEs, it allows us to attain a similar global error even with larger step sizes, which implies that RX-Euler requires fewer NFEs to produce comparable results. Similarly, for higher-order solvers, we can demonstrate that RX-DPM has a more rapid convergence of the global truncation errors using the same approach.

5 EXPERIMENT

5.1 IMPLEMENTATION DETAILS

We conduct the experiment using EDM (Karras et al., 2022), Stable Diffusion V2[1] (Rombach et al., 2022), DPM-Solver (Lu et al., 2022), PNDM (Liu et al., 2022), SN-DPM and NPR-DPM (Bao et al., 2022a) using their official implementations and provided pretrained models. Throughout all experiments, we do not modify any default settings including the seed number from the official codes except for additional hyperparameters related to the proposed method. For the experiment with EDM, DPM-Solver, PNDM, SN-DPM, and NPR-DPM backbones, we generate 50K images, and then use included code in their implementations to measure FID (Heusel et al., 2017). For Stable Diffusion, we use PyTorch implementation of FID [2] and CLIP (Radford et al., 2021) model [3] with the patch size of 32×32.

5.2 VALIDITY OF RX-DPM

We first conduct validity test of RX-Euler under EDM backbone with $k \in \{2, 3, 4\}$. Smaller k implies that extrapolation occurs more frequently under the same number of time steps. Figure 2 shows that compared to Euler method without extrapolation, RX-Euler with all k improves the FID

---

[1] https://github.com/Stability-AI/stablediffusion

[2] https://github.com/mseitzer/pytorch-fid

[3] https://huggingface.co/openai/clip-vit-base-patch32

scores with great margin. In particular, $k = 2$ which performs extrapolation every two steps achieves the best in general throughout a wide range of NFEs. This indicates that frequent extrapolation helps obtain more accurate prediction of intermediate output thus mitigate negative effect on final sample from error accumulations. In addition, from the results for $k = 4$, which significantly outperform the baseline with few times of extrapolation, we confirm that the reduced truncation error derived in Equation (19) is empirically well-validated for general k. For the rest of our results, we set $k = 2$.

We also compare the proposed method with conventional Richardson extrapolation using Equation (9) with $k = 2$ which is labled as Naïve in Figure 2. We observe that naïve implementation of Richardson extrapolation does not work well for limited NFEs, and even at larger NFEs, it only results in marginal improvements. This is primarily because it fails to benefit from the NFE reduction effect of repeated extrapolation and the mitigation of error propagation. Furthermore, our proposed method reflects the characteristics of DPMs, where the importance of precision varies along the time (Karras et al., 2022), leading to better results.

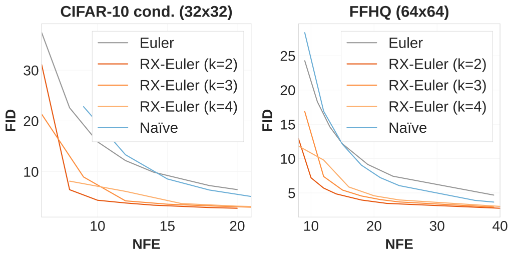


Figure 2: Effect of extrapolation on the Euler method with different k.

5.3 QUANTITATIVE COMPARISONS ON EDM BACKBONE

Figure 3 compares RX-Euler with other methods on four different datasets—CIFAR-10 (Krizhevsky & Hinton, 2009), FFHQ (Karras et al., 2019), AFHQv2 (Choi et al., 2020) and ImageNet (Deng et al., 2009)—using EDM backbone. First, we compare the result of Heun's method, labeled EDM, which is employed by its original paper. Both Heun's method and RX-Euler have a commonality in that they achieve a higher order of accuary compared to Euler method as second-order solvers. We also compares the results with other recent extrapolation-based approaches—LA-DPM (Zhang et al., 2023) and IIA (Zhang et al., 2024). The results for LA-DPM are reproduced by ours with the extrapolation hyperparameter presented in the paper fixed at $\lambda = 0.3$. Since we find that LA-DPM yields better results with Euler method than with Heun's method, we use the values obtained from Euler for LA-DPM results. For the results of IIA (Zhang et al., 2024), the values are brought from the tables of the paper, and between the two suggested methods, IIA and BIIA, the better results are used.

We observe that RX-Euler surpasses the other methods with large margin on a wide range of NFEs especially for small NFEs, *i.e.*, $N \leq 10$ for CIFAR-10, $N \leq 18$ for FFHQ and $N \leq 20$ for AFHQv2 and ImageNet. In the comparison between RX-Euler and Heun, Heun generally performs better at larger NFEs and vice versa, suggesting that each method may be advantageous in different ranges. This implies that we could achieve better results by selecting more favorable solver for each interval. Selecting solver can be viewed as a question of whether interpolation or extrapolation is more advantageous. Even if the solvers have the same order of accuracy, the constant of the leading error term is unknown, making it necessary to conduct experiments to determine which method is superior. However, one can predict that in the earlier steps, close to the noise and thus prone to have a lower accuracy of prediction, interpolation is likely to be more stable than extrapolation. Based on this reasoning, we experiment with a hybrid approach, employing RX-Euler for a portion of the steps and Heun for the remainder. Specifically, we apply RX-Euler to the middle half of the steps for CIFAR-10 and the last half of the steps for the other datasets. We label such case as RX+EDM in Figure 3 and we discover that a proper combination of the two methods yields better results than either method alone, achieving the best performance over a wider range of intervals, especially for FFHQ, AFHQv2, and ImageNet datasets. Although this approach is heuristic, it suggests that there is still room for improvement in our algorithm and provides another direction for future work.

5.4 STABLE DIFFUSION

We also apply RX-DDIM on Stable Diffusion V2 which provides various conditional generations. We run text-to-image generation using 10K different texts from COCO2014 (Lin et al., 2014) validation set to generate 10K images sized 512×512 and measure FID and CLIP scores of DDIM and RX-DDIM. Table 1 shows that our method also works for the large models from improved FID scores. However, we observe RX-DDIM has lower CLIP scores for 15 NFEs. On this issue, we

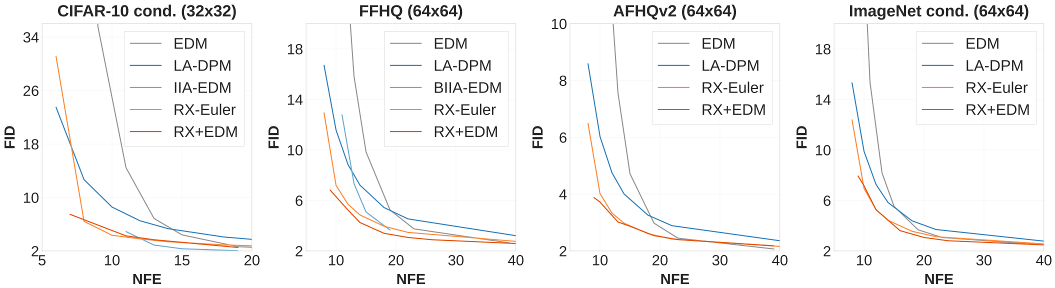


Figure 3: FIDs of RX-Euler, Heun's method (EDM), LA-DPM and IIA (/BIIA) versus NFEs on CIFAR-10 (cond.), FFHQ, AFHQv2, and ImageNet (cond.) datasets using EDM backbone.

Table 1: FID and CLIP scores of DDIM and RX-DDIM using Stable Diffusion V2.

NFE	15		20		30		50	
Metric	FID (↓)	CLIP (↑)	FID (↓)	CLIP (↑)	FID (↓)	CLIP (↑)	FID (↓)	CLIP (↑)
DDIM	19.15	**31.727**	18.43	31.716	19.00	31.750	18.65	31.711
RX-DDIM	**17.24**	31.629	**17.12**	**31.721**	**17.62**	**31.781**	**17.83**	**31.727**

believe it may be related to the classifier-guidance scales. According to Rombach et al. (2022), optimal classifier-free guidance scales vary for different models, and the default setting is optimized for DDIM. As we does not perform a search on such hyperparameters, we expect there is room for improvement in our method. Moreover, Figures 5 and 6 in Appendix F compare the qualitative results, where we confirm that the results of RX-DDIM show better image quality and align more closely with the text conditions.

5.5 HIGHER-ORDER SOLVERS

RX-DPM is also applicable to more advanced ODE-based samplers with higher-order accuracy as depicted in Section 4.3. In Table 2, we show the effectiveness of RX-DPM upon DPM-Solvers (Lu et al., 2022) on CIFAR-10 and LSUN Bedroom (Yu et al., 2015) datasets. Among the variations of DPM solvers, the single-step version of DPM-Solver-2 and DPM-Solver-3 are used. Note that, since a single-step DPM-solver-n can be considered as an n^{th}-order Runge-Kutta-like solver, we apply RX-DPM with $p = n + 1$ in Equation (23) for the DPM-solver-n. Additionally, we compare the results with another accelerated diffusion sampler, DEIS (Zhang & Chen, 2023) on class-conditioned ImageNet (64×64) in Table 5 of Appendix D and observe that RX-DPM demonstrates the best performance across all NFEs.

As another type of advanced sampler, we choose PNDM (Liu et al., 2022); S-PNDM and F-PNDM utilize the linear multistep methods, *i.e.*, 2-step and 4-step Adam-Bashforth methods, respectively, except for the first few time steps. Therefore, we apply RX-DPM with $p = 3$ and $p = 5$ in Equation (23) for S-PNDM and F-PNDM, respectively. The results on CIFAR-10, CelebA (Liu et al., 2015) and LSUN Church (Yu et al., 2015) datasets are shown in Table 3. While RX-DPM shows improved performances in most cases, there is a notable exception with F-PNDM solver on LSUN Church dataset, where RX-DPM does not work well. Analyzing this, we find that the baseline result of F-PNDM is best when the number of time steps is 10, and it gets progressively worse as the number of time steps increases (even up to 250, the result with 10 steps is still the best). RX-DPM leverages the momentum from the improvement of the baseline solver's accuracy over finer time steps. However, in this case, finer intervals result in worse performance, indicating that the extrapolation does not enhance performance as expected. Regarding this, IIA (Zhang et al., 2024) also reported similar phenomena with F-PNDM on LSUN datasets.

5.6 DPMS WITH OPTIMAL COVARIANCES

Previously, we conducted experiments from the perspective of ODE solvers. To verify the effectiveness of our method on models optimized with SDE, we also conduct experiment with the SN-DPM and NPR-DPM (Bao et al., 2022a) on CIFAR-10 (Krizhevsky & Hinton, 2009) and CelebA datasets (Liu et al., 2015). SN-DPM and NPR-DPM are two different models that correct the imperfect mean

Table 2: FID scores of DPM-Solvers (Lu et al., 2022) and RX-DPMs applied to DPM-Solvers on CIFAR-10 and LSUN Bedroom datasets. All baseline results are reproduced under the same setting as RX-DPMs.

	CIFAR-10 (32×32)				LSUN Bedroom (256×256)			
Method \ NFEs	9	10	12	15	9	10	12	15
DPM-Solver-2	–	15.06	11.33	7.36	–	14.67	11.38	6.44
RX-DPM-Solver-2	–	**12.94**	**9.80**	**6.53**	–	**12.66**	**10.13**	**5.72**
DPM-Solver-3	12.39	–	6.76	5.00	8.79	–	5.37	**4.04**
RX-DPM-Solver-3	**11.50**	–	**6.62**	**4.85**	**8.12**	–	**5.18**	**4.04**

Table 3: FID scores of two PNDM solvers (Liu et al., 2022) and RX-DPMs applied to each PNDM on CIFAR-10, CelebA and LSUN Church datasets. Note that S-PNDM and F-PNDM require 1 and 9 additional NFEs to the number of time steps, respectively. The baseline results are copied from PNDM (Liu et al., 2022).

	CIFAR-10 (32×32)			CelebA (64×64)			LSUN Church (256×256)		
Method \ # of steps	5	10	20	5	10	20	5	10	20
S-PNDM	**18.3**	8.64	5.77	15.2	12.2	9.45	**20.5**	11.8	9.20
RX-S-PNDM	19.69	**7.64**	**4.72**	**11.56**	**9.22**	**6.89**	21.15	**10.96**	**8.96**
F-PNDM	18.2	7.05	4.61	11.3	7.71	5.51	14.8	**8.69**	**9.13**
RX-F-PNDM	–	**6.60**	**3.99**	–	**7.10**	**4.99**	–	8.85	9.41

Table 4: FID scores for CIFAR-10 and CelebA on DDIM, NPR-DDIM and SN-DDIM models. The values for each baseline and LA-DDIM results are copied from Zhang et al. (2023).

Dataset	CIFAR-10 (32×32)					CelebA (64×64)				
NFE	10	25	50	100	200	10	25	50	100	200
DDIM	21.31	10.70	7.74	6.08	5.07	20.54	13.45	9.33	6.60	4.96
RX-DDIM	**14.78**	**8.42**	**6.30**	**4.96**	**4.31**	**18.31**	**10.54**	**6.88**	**4.47**	**3.56**
NPR-DDIM	13.40	5.43	3.99	3.52	3.40	14.94	9.18	6.17	4.40	3.67
LA-NPR-DDIM	10.74	4.71	3.64	3.33	3.29	14.25	8.83	5.67	3.76	2.95
RX-NPR-DDIM	**6.35**	**3.92**	**3.34**	**3.13**	**3.18**	**11.58**	**6.61**	**3.98**	**2.75**	**2.51**
SN-DDIM	12.19	4.28	3.39	3.23	3.22	10.17	5.62	3.90	3.21	2.94
LA-SN-DDIM	8.48	**3.15**	**2.93**	**2.92**	**3.08**	8.05	4.56	2.93	**2.39**	**2.19**
RX-SN-DDIM	**7.50**	5.12	4.40	4.18	3.62	**5.20**	**2.72**	**2.25**	2.49	2.44

prediction in the reverse process of existing models through optimal covariance learning. Therefore, when performing DDIM using these models, stochasticity arises due to optimal covariances, resulting in a non-ODE solution. To apply our method in this case, we calculate $\hat{\boldsymbol{x}}^{(1)}_{t_{i-2}}$ and $\hat{\boldsymbol{x}}^{(2)}_{t_{i-2}}$ from t_i using an ODE solver, *i.e.*, DDIM, without adding covariances. Then after the extrapolation, we apply the optimal covariance to $\tilde{\boldsymbol{x}}^{(2)}_{t_{i-2}}$ correct the values. In this way, our method applies the covariance usage steps in a limited manner (half times) compared to the baseline sampling with the same NFEs.

In Table 4, we show the results on NPR-DDIM and SN-DDIM along with the vanilla DDIM and compare with LA-DPM (Zhang et al., 2023) as well. We observe that our method outperforms in most cases, although a performance degradation was noted with SN-RX-DDIM on the CIFAR-10 dataset. Upon analysis, we find that this model has a relatively large covariances compared to other cases. Consequently, our approach of solving the ODE less benefits from the model's optimization. Despite this, we observe significant performance improvements at the most extreme NFEs $= 10$. Moreover, our method significantly exceeds the baseline, even surpassing LA-DPM with a large margin for limited NFEs with models with less stochasticity.

6 CONCLUSION

We presented an advanced sampling method, RX-DPM, which performs extrapolation by employs two ODE solutions with different discretizations. In consequence, we could reduce the local truncation error effectively, thereby achieving better sample qualities. We showed the effectiveness of RX-DPM by conducting experiments on well-known baseline models, datasets, and comparing RX-DPMs with other sampling methods.

ETHICS STATEMENT

This paper leverages pretrained diffusion probabilistic models to generate high-quality images. The proposed method is focused on efficiency and thus our application of diffusion models does not directly introduce hazardous elements. Nonetheless, we recognize that it can potentially be used to synthesize data with unexpectedly inappropriate or sensitive content.

REPRODUCIBILITY STATEMENT

We provide detailed instructions for implementation and reproducibility of our results in Section 5.1 and pseudo code of the main part in Algorithm 1. We will release the code.

REFERENCES

Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. In *ICML*, 2022a.

Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-DPM: An analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *ICLR*, 2022b.

Mike A Botchev and Jan G Verwer. Numerical integration of damped maxwell equations. *SIAM Journal on Scientific Computing*, 31(2):1322–1346, 2009.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-order denoising diffusion solvers. In *NeurIPS*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *NeurIPS*, 2022.

Junoh Kang, Jinyoung Choi, Sungik Choi, and Bohyung Han. Observation-guided diffusion probabilistic models. In *CVPR*, 2024.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion, 2024.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL `http://arxiv.org/abs/1405.0312`.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *ICLR*, 2022.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Shane A Richards. Completed richardson extrapolation in space and time. *Communications in numerical methods in engineering*, 13(7):573–582, 1997.

Lewis Fry Richardson. Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London*, 210(459-470):357–357, 1911. doi: https://doi.org/10.1098/rsta.1911.0009.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-video generation without text-video data. In *ICLR*, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023.

Endre Süli and David Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 1 edition, 2003. ISBN 0-521-00794-1.

Sauer Timothy. *Numerical Analysis*. Pearson, 3 edition, 2017. ISBN 2017028491, 9780134696454, 013469645X.

Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. ModelScope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *ICLR*, 2022.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent point diffusion models for 3D shape generation. In *NeurIPS*, 2022.

Guoqiang Zhang, Niwa Kenta, and W. Bastiaan Kleijn. Lookahead diffusion probabilistic models for refining mean estimation, 2023.

Guoqiang Zhang, Niwa Kenta, and W. Bastiaan Kleijn. On accelerating diffusion-based sampling process via improved integration approximation, 2024.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator, 2023.

Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. MagicVideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022.

Zahari Zlatev, István Faragó, and Ágnes Havasi. Stability of the richardson extrapolation applied together with the θ-method. *Journal of Computational and Applied Mathematics*, 235(2):507–517, 2010.

APPENDIX

A LIMITATIONS

As our method is primarily designed for an ODE solver, to integrate it with an SDE solver (or a stochastic sampling method), we partially apply the stochasticity component of the SDE solver as demonstrated in Section 5.6. Consequently, in some cases, the effectiveness is offset because the full effects of stochasticity are not captured. However, in scenarios where NFE is very limited, which are of greater interest to us, the combined effect of RX-DPM and stochastic sampling has been empirically shown to be highly beneficial. We leave the development of methods that can perform better in more general cases for future work. Additionally, in the extension of the RX-Euler algorithm to a higher-order solver in Section 4.3, there remains room for improvement since we impose assumptions about linear error propagation. We believe that relaxing these assumptions or deriving more accurate equations could further enhance the performance.

B JUSTIFICATION ON EQUATION (22)

Assuming a uniform grid as in the context of conventional Richardson extrapolation in Section 3.2, and the ODE solver with $O(h^{p+1})$ of local truncation error formula, we have

$$V^* = V(h) + ch^p + O(h^{p+1}) \quad \text{and} \tag{31}$$

$$V^* = V(\frac{h}{k}) + c(\frac{h}{k})^p + O(h^{p+1}), \tag{32}$$

since we expect the $O(h^p)$ of the global truncation error. Then, we have the following extrapolated solution with a truncation error of $O(h^{p+1})$ by Richardson extrapolation (Equation (9)):

$$\tilde{V}(h,k) = \frac{k^p V(h/k) - V(h)}{k^p - 1}. \tag{33}$$

Now, considering the case of Equations (21) and (22) with uniform discretization, we have

$$V^* = V(h) + c'h^{p+1} + O(h^{p+2}) \quad \text{and} \tag{34}$$

$$V^* = V(\frac{h}{k}) + kc'(\frac{h}{k})^{p+1} + O(h^{p+2}), \tag{35}$$

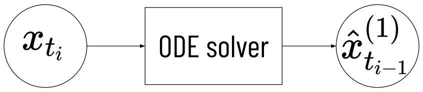
each correspondingly. Then, the extrapolated solution $\tilde{V}_{\text{ours}}$ obtained from solving linear system of Equations (34) and (35) becomes

$$\tilde{V}_{\text{ours}} = \frac{k^p V(h/k) - V(h)}{k^p - 1}, \tag{36}$$

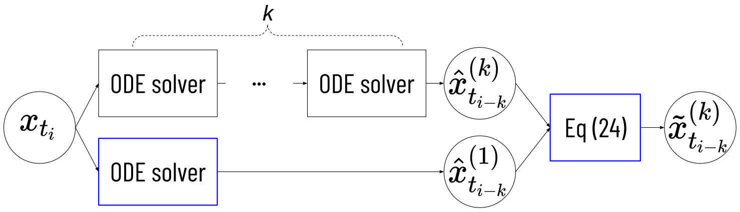
which turns out to be exactly the same as Equation (33). Thus, we believe our approach can be considered to employ assumptions shared by those used in common practices of Richardson extrapolation and also can reduce global errors which is backed by experimental results as well.

C DIAGRAMS

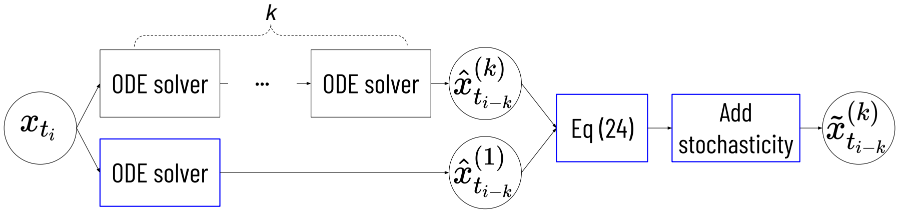
Figure 4 compares the diagrams of an ODE solver, the proposed method with an ODE solver, and the proposed method with an SDE solver. To integrate our method with the SDE solver, we interpret the SDE solver as a combination of a deterministic sampling component and a stochasticity component. We then utilize the deterministic sampling part as an ODE solver, as illustrated in Figure 4 (c).



(a) One step of an ODE solver


k
ODE solver
···
ODE solver
x_{t_i}
$\hat{x}^{(k)}_{t_{i-k}}$
Eq (24)
$\tilde{x}^{(k)}_{t_{i-k}}$
ODE solver
$\hat{x}^{(1)}_{t_{i-k}}$

(b) k steps of RX-DPM applied to an ODE solver


k
ODE solver
···
ODE solver
x_{t_i}
$\hat{x}^{(k)}_{t_{i-k}}$
Eq (24)
Add stochasticity
$\tilde{x}^{(k)}_{t_{i-k}}$
ODE solver
$\hat{x}^{(1)}_{t_{i-k}}$

(c) k steps of RX-DPM applied to an SDE solver

Figure 4: Digarams of the baseline and the proposed sampling methods. The blue-bordered boxes in (b) and (c) indicate that the corresponding operation does not require network evaluation. The ODE solver in (c) refers to the deterministic sampling component of the SDE solver.

D COMPARISON WITH DEIS

Table 5: Comparisons of DEIS variants (Zhang & Chen, 2023), DPM-Solvers and RX-DPMs applied to DPM-Solvers on class-conditioned ImageNet (64×64). All results of DEIS and DPM-Solvers are copied from DEIS (Zhang & Chen, 2023) except for the result of DPM-Solver-3 with NFEs $= 9$.

	NFEs				
Method	9	10	12	18	30
tAB-DEIS	–	6.65	3.99	3.21	2.81
ρAB-DEIS	–	9.28	6.46	3.74	2.87
DPM-Solver-2	–	7.93	5.36	3.63	3.00
ρMid-DEIS	–	9.12	6.78	4.00	2.99
RX-DPM-Solver-2	–	**6.11**	5.61	3.64	2.93
DPM-Solver-3	7.45	–	5.02	3.18	2.84
ρKutta-DEIS	–	–	13.12	3.63	2.82
RX-DPM-Solver-3	**7.08**		**3.90**	**2.36**	**2.18**

E COMPUTATIONAL COMPLEXITY

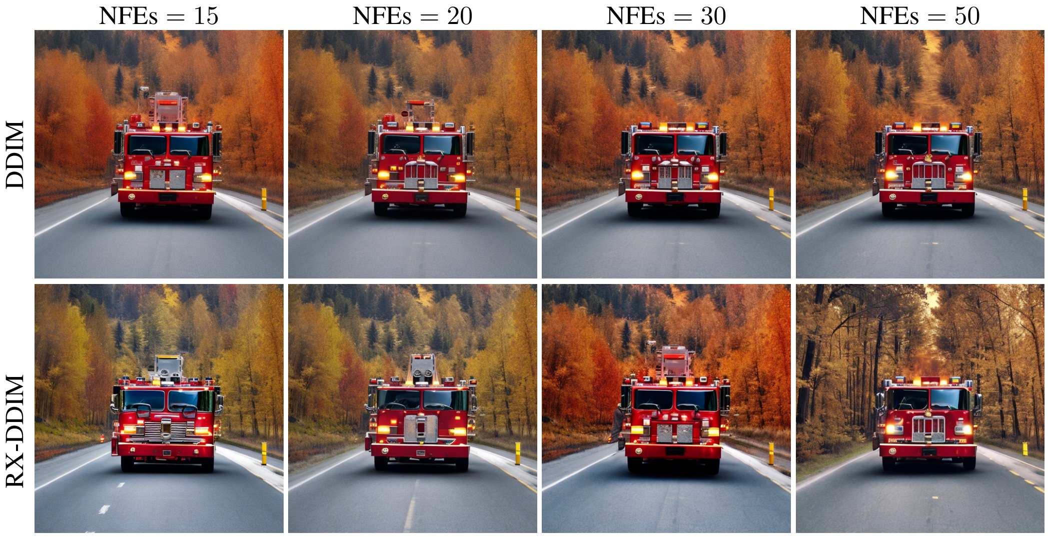
We compare the computational costs of the Euler method and RX-Euler using the EDM backbone in Table 6. The average runtime per batch is measured for 10-step sampling with a batch size of 128. The additional operations introduced by our method, which consist of linear combinations of precomputed values, result in negligible computational overhead compared to the time required for the network forward pass. Furthermore, as the model size increases, the relative overhead diminishes (*e.g.*, only a 0.11% increase for ImageNet class-conditional sampling).

Table 6: Comparison of per-batch computation times between the Euler method and RX-Euler with the EDM backbone. The reported values represent the average runtime across 100 measurements (in seconds).

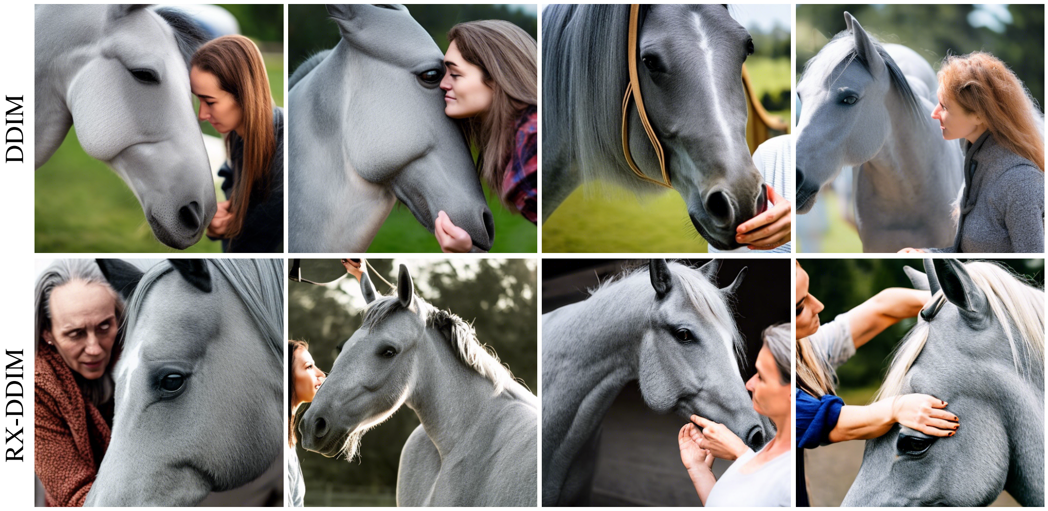
	CIFAR-10 cond. (32x32)	FFHQ (64x64)	ImageNet cond. (64x64)
Euler	1.737 ± 0.028	3.895 ± 0.023	6.436 ± 0.033
RX-Euler	1.743 ± 0.031	3.903 ± 0.025	6.443 ± 0.039

F QUALITATIVE RESULTS

We provide qualitative results on Stable diffusion V2 in Figures 5 and 6 and EDM backbone in Figures 7 to 10.

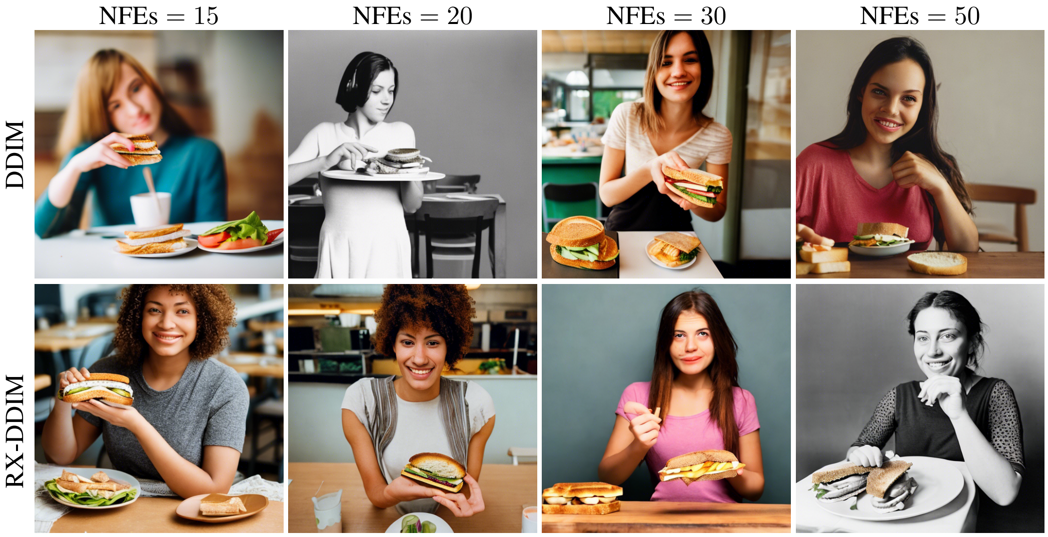


“A red fire truck driving down a road.”

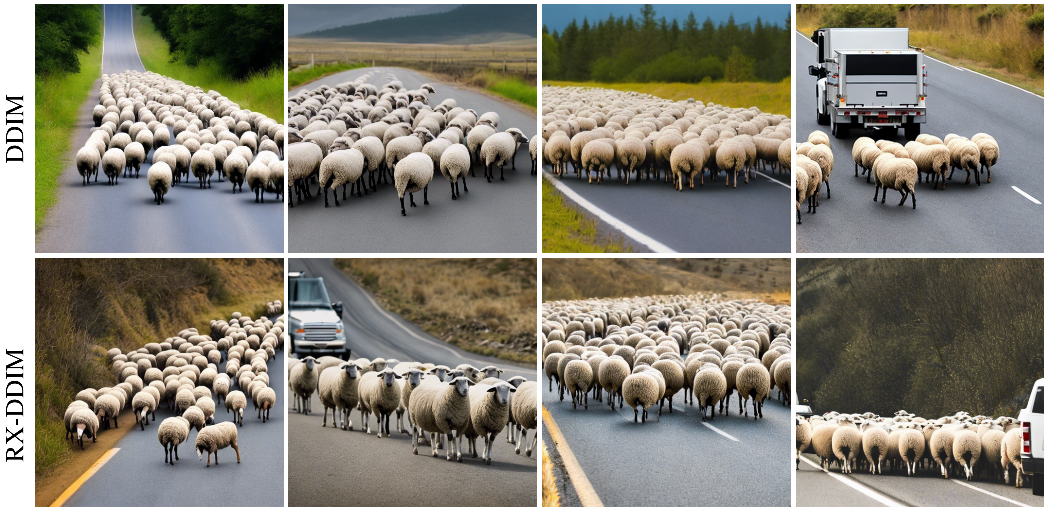


“A woman is looking at a grey horse in the eye.”

Figure 5: Qualitative results on Stable Diffusion V2 of DDIM and RX-DDIM.

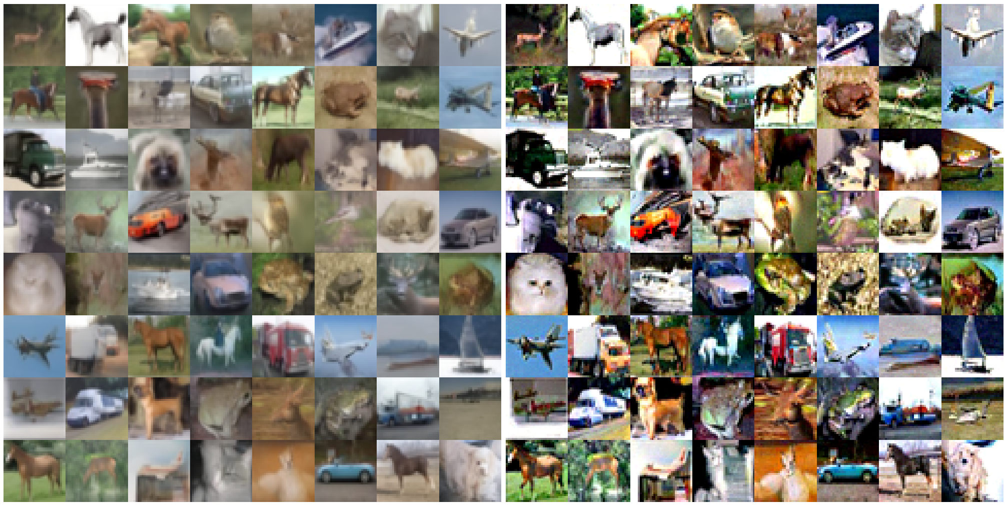


“A young woman holding a sandwich at a table.”

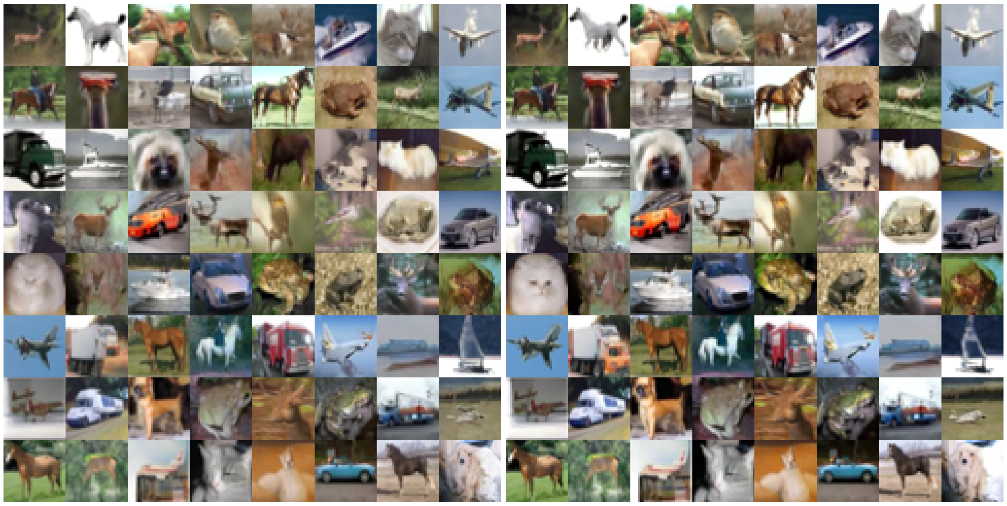


“A herd of sheep walking down a road in front of a silver truck.”

Figure 6: Qualitative results on Stable Diffusion V2 of DDIM and RX-DDIM.

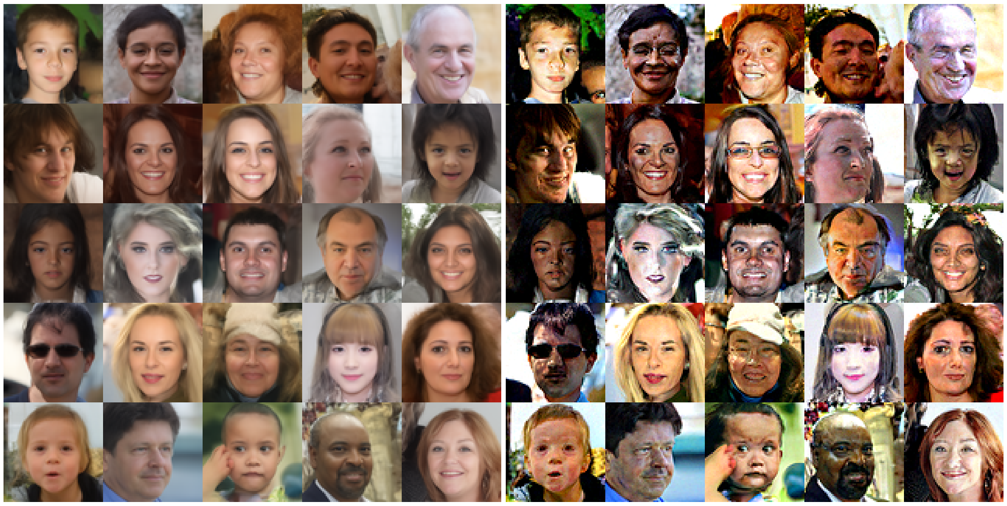


Euler (NFEs $=10$, FID: 15.88) EDM (NFEs $=11$, FID: 14.46)

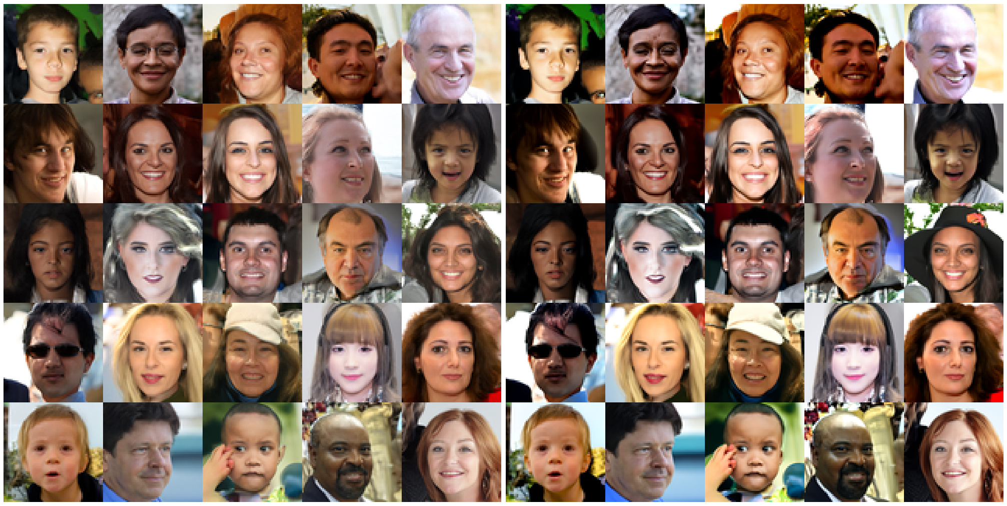


RX-Euler (NFEs $=10$, FID 4.35) RX+EDM (NFEs $=10$, FID 4.26)

Figure 7: Qualitative results of CIFAR-10 of different sampling methods with EDM backbone.

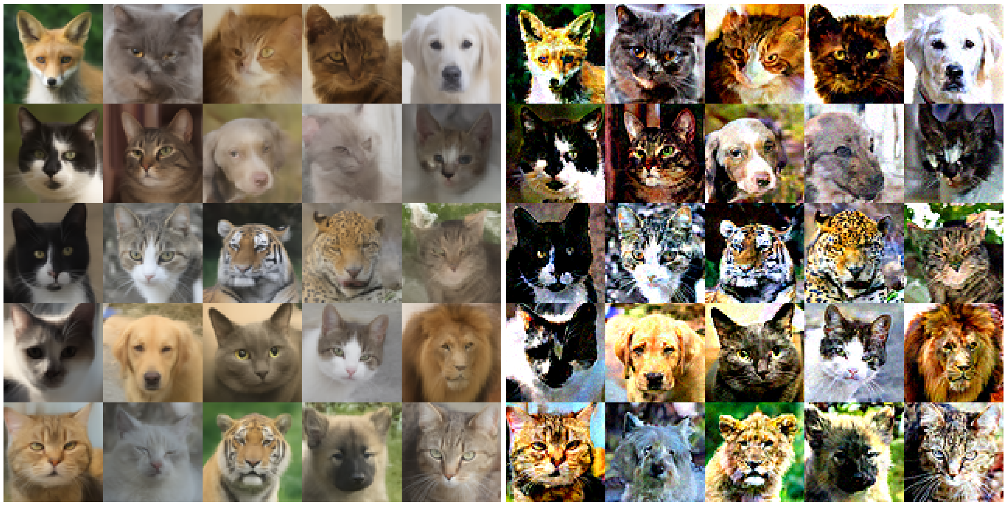


Euler (NFEs $= 11$, FID: 18.30) EDM (NFEs $= 11$, FID: 29.34)

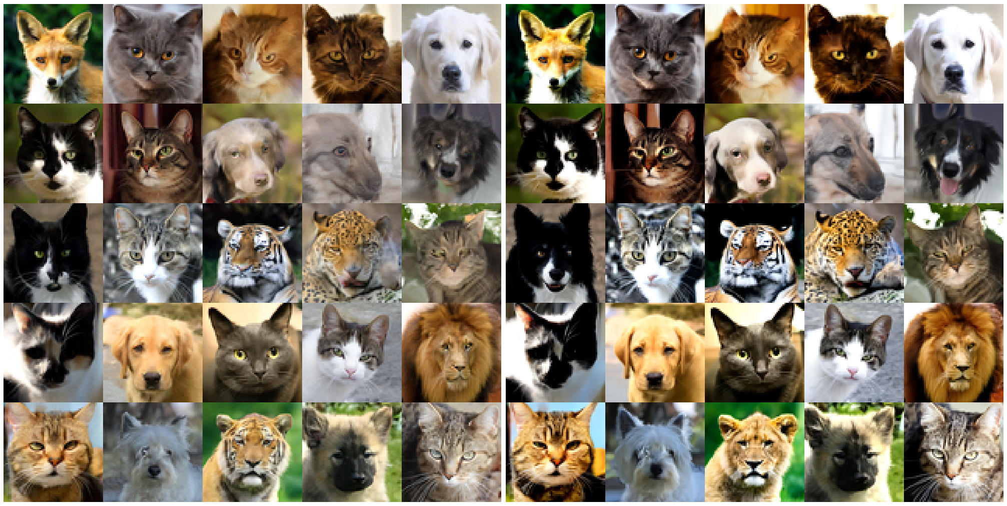


RX-Euler (NFEs $= 10$, FID: 7.20) RX+EDM (NFEs $= 11$, FID: 5.50)

Figure 8: Qualitative results of FFHQ of different sampling methods with EDM backbone.

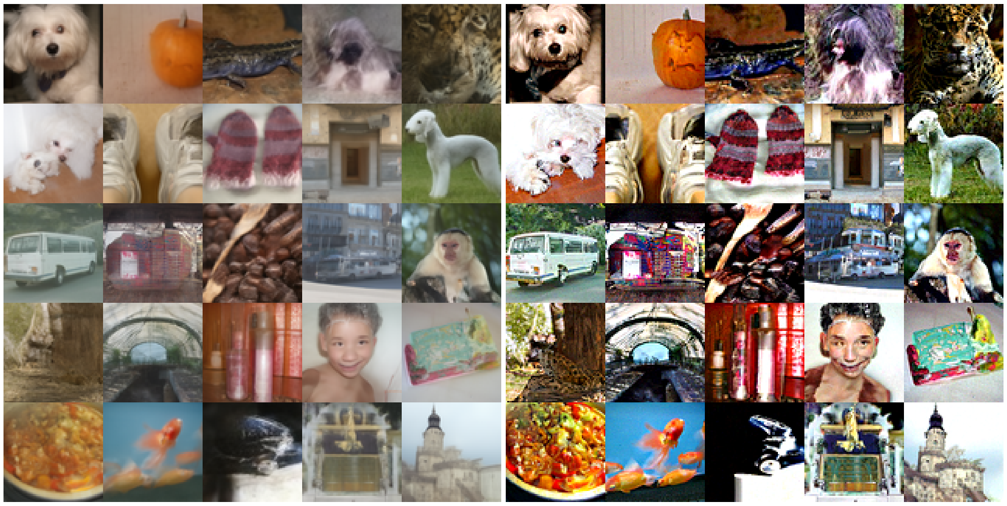


Euler (NFEs $= 9$, FID: 11.81) EDM (NFEs $= 9$, FID: 27.92)

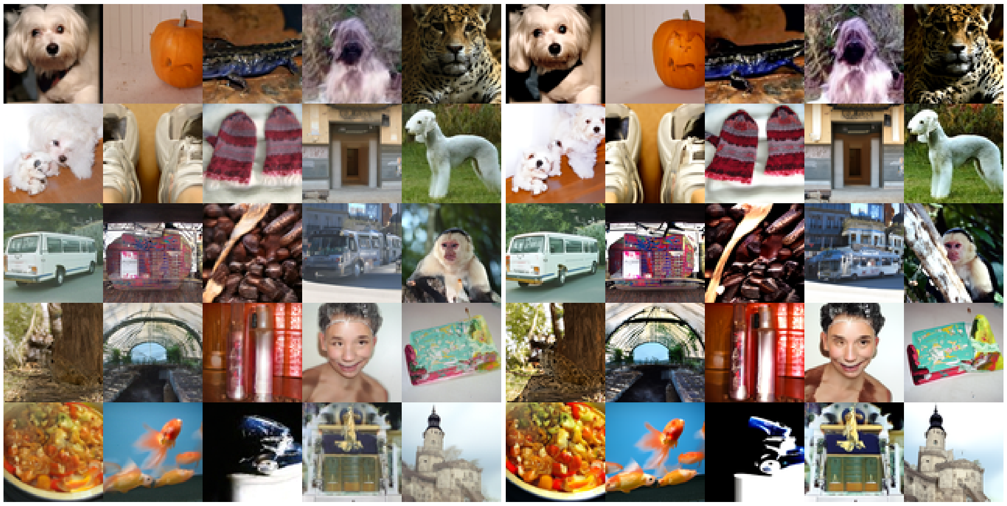


RX-Euler (NFEs $= 8$, FID: 6.49) RX+EDM (NFEs $= 9$, FID: 3.88)

Figure 9: Qualitative results of AFHQv2 of different sampling methods with EDM backbone.



Euler (NFEs = 11, FID: 14.84) EDM (NFEs = 11, FID: 15.35)



RX-Euler (NFEs = 10, FID: 6.95) RX+EDM (NFEs = 11, FID: 5.83)

Figure 10: Qualitative results of ImageNet of different sampling methods with EDM backbone.