Accelerating Parallel Sampling of Diffusion Models

Zhiwei Tang [1 *] **Jiasheng Tang** [2 3] **Hao Luo** [2 3] **Fan Wang** [2] **Tsung-Hui Chang** [1 4]

Abstract

Diffusion models have emerged as state-of-the-art generative models for image generation. However, sampling from diffusion models is usually time-consuming due to the inherent autoregressive nature of their sampling process. In this work, we propose a novel approach that accelerates the sampling of diffusion models by parallelizing the autoregressive process. Specifically, we reformulate the sampling process as solving a system of *triangular nonlinear equations* through fixed-point iteration. With this innovative formulation, we explore several systematic techniques to further reduce the iteration steps required by the solving process. Applying these techniques, we introduce **ParaTAA**, a universal and **training-free** parallel sampling algorithm that can leverage extra computational and memory resources to increase the sampling speed. Our experiments demonstrate that ParaTAA can decrease the inference steps required by common sequential sampling algorithms such as DDIM and DDPM by a factor of **4∼14 times**. Notably, when applying ParaTAA with 100 steps DDIM for Stable Diffusion, a widely-used text-to-image diffusion model, it can produce the same images as the sequential sampling in only **7 inference steps**. The code is available at `https://github.com/TZW1998/ParaTAA-Diffusion`.

1. Introduction

In recent years, diffusion models have been recognized as state-of-the-art for generating high-quality images, demonstrating exceptional resolution, fidelity, and diversity (Ho et al., 2020; Dhariwal & Nichol, 2021; Song et al., 2020b). These models are also notably easy to train and can be effectively extended to conditional generation (Ho & Salimans, 2022). Broadly speaking, diffusion models work by learning to reverse the diffusion of data into noise, a process that can be described by a stochastic differential equation (SDE) (Song et al., 2020b; Karras et al., 2022):

$$dx_t = f(t)x_t dt + g(t)dw_t, \tag{1}$$

where dw_t is the standard Wiener process, and $f(t)$ and $g(t)$ are the drift and diffusion coefficients, respectively. The reverse process relies on the score function $\epsilon(x_t, t) \overset{\text{def.}}{=} \nabla_x \log p(x_t)$, and its closed form can be expressed either as an ordinary differential equation (ODE) (Song et al., 2020b):

$$dx_t = \left(f(t)x_t - \frac{1}{2}g^2(t)\epsilon(x_t, t) \right) dt, \tag{2}$$

or as an SDE:

$$dx_t = \big(f(t)x_t - g^2(t)\epsilon(x_t, t) \big) dt + g(t)dw_t. \tag{3}$$

With the ability to evaluate $\epsilon(x_t, t)$, it becomes possible to generate samples from noise by numerically solving the ODE (2) or the SDE (3). The training process, therefore, involves learning a parameterized surrogate $\epsilon_\theta(x_t, t)$ for $\epsilon(x_t, t)$ following a denoising score matching framework described in (Song et al., 2020b; Karras et al., 2022).

Accelerating Diffusion Sampling. As previously mentioned, the sampling process in diffusion generative models involves solving the ODE (2) or SDE (3). This process requires querying the learned neural network ϵ_θ in an autoregressive way, which can limit sampling speed particularly when ϵ_θ represents a large model such as Stable Diffusion (SD) (Rombach et al., 2022). To accelerate the sampling process, existing works explore several avenues, which we summarize briefly here.

One avenue is to distill the ODE trajectory of the diffusion sampling process into another neural network that enables fewer-step sampling, with representative works including (Song et al., 2023b; Liu et al., 2023; Sauer et al., 2023; Salimans & Ho, 2022; Meng et al., 2023; Geng et al., 2023). However, this class of methods often leads to degradation in image quality and diversity.

[*] This work was done when Zhiwei Tang was intern at DAMO Academy. [1] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China [2] DAMO Academy, Alibaba Group [3] Hupan Lab, Zhejiang Province [4] Shenzhen Research Institute of Big Data, Shenzhen, China. Correspondence to: Zhiwei Tang <zhiweitang1@link.cuhk.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Another direction involves developing faster sequential ODE/SDE solvers for (2)/(3) based on mathematical principles, with contributions from (Lu et al., 2022; Song et al., 2020a; Karras et al., 2022; Zhao et al., 2023). However, the improvements from these approaches tend to be incremental, given the years of progress in the field.

A recent and promising direction, pioneered by (Shih et al., 2023), aims to parallelize the autoregressive sampling process of diffusion models by employing Picard-Lindelöf (PL) iterations for solving the corresponding ODE/SDE. This approach has three main advantages over other existing methods: 1. It does not require extra training; 2. It can lead to (almost) the same images as sequential sampling; 3. It can significantly reduce the inference steps by leveraging extra computing resources. Similar concepts of parallelizing autoregressive inference have also been investigated in the acceleration of Large Language Models (LLMs), such as speculative sampling (Leviathan et al., 2023; Sun et al., 2023), and in common autoregressive procedures (Song et al., 2021; Lim et al., 2023). We focus on this direction in this work, proposing a novel and more efficient algorithm for parallelizing the sampling process of diffusion models.

1.1. Prior Work

To the best of our knowledge, the recent work by (Shih et al., 2023) stands as the only study focusing on the parallel sampling of diffusion models. For a general ODE expressed as $x_t = \int_0^t S(x_u, u)du$, the PL iteration adopted in (Shih et al., 2023) refines an initial discretized trajectory $x_0^{\text{old}}, ..., x_T^{\text{old}}$ through the following fixed-point iteration:

$$x_i^{\text{new}} = \frac{1}{T} \sum_{u=0}^{i-1} S\left(x_u^{\text{old}}, \frac{u}{T}\right), \text{ for } i = 0, ..., T. \qquad (4)$$

This approach allows the computationally intensive task, evaluating $S\left(x_u^{\text{old}}, \frac{u}{T}\right) : u = 0, ..., T$, to be executed in parallel. In practice, (Shih et al., 2023) observed that the PL iteration (4) requires significantly fewer than T steps to converge, thus expediting the autoregressive sampling process.

1.2. Our Contributions

In this paper, we introduce a novel and principled formulation for the parallel sampling of diffusion models, which includes the method proposed by (Shih et al., 2023) as a special case. The primary advantage of this new formulation is that it enables us to rigorously investigate its convergence properties, thus new techniques to improve sampling efficiency are made possible. Besides, differing from (Shih et al., 2023), our study is exclusively concentrated on image generation. Specifically, our contributions are:

(1) We formulate the parallel sampling of diffusion models as solving a system of *triangular nonlinear equations* using fixed-point iteration (FP), which can be seamlessly integrated with any existing sequential sampling algorithms by adjusting the coefficients in the equations.

(2) Inspired by classical optimization theory on nonlinear equations, we develop several techniques to enhance the efficiency of FP. Firstly, we reveal that the convergence behavior of FP is largely attributed to the iteration function, and propose a systematic way to construct an improved iteration function via equivalent transformation on the nonlinear equations. Secondly, to efficiently bootstrap the information from previous iterations, we propose a new variant of the Anderson Acceleration technique (Walker & Ni, 2011) tailored for the triangular nonlinear equations. Lastly, we identify two practical tricks through experiments: early stopping—terminating the iteration once a perceptual criterion is met in the generated image; and a useful initialization strategy—initializing the process with the solution from a similar, previously solved equation.

(3) As a byproduct, particularly for text-to-image generation with Stable Diffusion, we observe that when initializing with the sampling trajectory of a similar prompt, one can obtain a smooth interpolation between the source image and the target image in very few steps. This can have implications for tasks such as image variation, editing (Meng et al., 2022), and prompt optimization (Hao et al., 2022).

Paper Outline. We begin by formulating the diffusion sampling problem as solving triangular nonlinear systems in Section 2, and then discuss how to obtain a better iteration function for FP. In Section 3, we introduce how data from previous iterations should be used to speed up the iteration process. Subsequently, we discuss the two useful tricks to further enhance sampling efficiency in Section 4. Lastly, Section 5 presents experimental results on cutting-edge image diffusion models, demonstrating the effectiveness of our proposed methods.

2. Formulating Diffusion Sampling as Solving Triangular Nonlinear Equations

We observe that every existing sampling algorithm for diffusion models, such as DDIM (Song et al., 2020a), DPM-Solver (Lu et al., 2022), and Heun (Karras et al., 2022), follows the autoregressive procedure in (5). Let T denote the discretization steps for the ODE/SDE, and $\xi_0, .., \xi_T$ be noise vectors drawn from standard Gaussian distribution. Starting with $x_T = \xi_T$, one computes $x_{T-1}, ..., x_0$ sequentially via the following equation from $t = T$ to $t = 1$:

$$x_{t-1} = \sum_{i=t}^{T} a_{t,i} x_i + \sum_{i=t}^{T} b_{t,i} \epsilon_\theta(x_i, i) + c_{t-1}\xi_{t-1}, \qquad (5)$$

where $a_{t,i}, b_{t,i}, c_t$ are coefficients determined by the specific sampling algorithm. Notably, for ODE solvers like DDIM

(Dhariwal & Nichol, 2021), it holds that $c_0 = ... = c_{T-1} = 0$, whereas for SDE solvers like DDPM (Ho et al., 2020), $c_0, ..., c_{T-1}$ are all non-zero.

For simplicity and due to a limit time, this work focuses on commonly used first-order solvers such as DDIM and DDPM, while leaving extensions to higher-order solvers like DPM-Solver and Heun as future works. For first-order solvers, (5) can be simplified to:

$$x_{t-1} = a_t x_t + b_t \epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1}, \; t = 1, ..., T. \quad (6)$$

Following the insights from (Song et al., 2021), we found that this autoregressive procedure (6) can be viewed as triangular nonlinear equations with $x_0, ..., x_{T-1}$ as the unknown variables. Besides, by further examination on (6), we reveal that these equations can be expressed in various equivalent forms. For instance, by incorporating the $(t+1)$-th equation into the first term of the t-th equation in (6), we derive an alternative t-th equation:

$$\begin{aligned} x_{t-1} =& a_t \Big(\underbrace{a_{t+1}x_{t+1} + b_{t+1}\epsilon_\theta(x_{t+1}, t+1) + c_t\xi_t}_{=x_t} \Big) \\ &+ b_t\epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1}. \end{aligned} \quad (7)$$

This leads us to define a series of equivalent nonlinear systems for the autoregressive procedure (6).

Definition 2.1 (k-th order nonlinear equations)**.** For any $1 \le k \le T$ with $x_T = \xi_T$, we define

$$x_{t-1} = F_{t-1}^{(k)}(x_t, x_{t+1}, ..., x_{t_k}) \,, t = 1, ..., T \quad (8)$$

as *the k-th order nonlinear equations* for the autoregressive sampling procedure (6), where $F_{t-1}^{(k)}$ is defined as

$$\begin{aligned} &F_{t-1}^{(k)}(x_t, x_{t+1}, ..., x_{t_k}) \overset{\text{def.}}{=} \bar{a}_{t,t_k} x_{t_k} \\ &+ \sum_{j=t}^{t_k} \bar{a}_{t,j-1} b_j \epsilon_\theta(x_j, j) + \sum_{j=t}^{t_k} \bar{a}_{t,j-1} c_{j-1}\xi_{j-1}, \end{aligned} \quad (9)$$

and $t_k \overset{\text{def.}}{=} \min\{t+k-1, T\}$, $\bar{a}_{i,s} = \prod_{j=i}^{s} a_j$. We denote $\bar{a}_{i,s} = 1$ for $s < i$.

From this definition, it is evident that the equations (8) with $k = 1$ correspond exactly to the autoregressive sampling procedure (6). Regarding this family of nonlinear equations, we assert the following:

Theorem 2.2. *The nonlinear equations (8) with different orders k are all equivalent and possess a unique solution.*

Fixed-point iteration is a classical method for solving nonlinear equations like (8). Given the set of variables $x_0^i, ..., x_{T-1}^i$ at the i-th iteration, the fixed-point iteration calculates the $(i+1)$-th iteration as follows:

$$x_{t-1}^{i+1} = F_t^{(k)}(x_t^i, x_{t+1}^i, ..., x_{t_k}^i), \quad t = 1, ..., T. \quad (10)$$

As can be seen, performing one iteration in (10) involves evaluating $\epsilon_\theta(x_1^i, 1), ..., \epsilon_\theta(x_T^i, T)$, which equates to inferring the neural network ϵ_θ T times. Fortunately, with sufficient computational resources like GPUs, these evaluations can be processed all in parallel, making the time cost comparable to a single query of ϵ_θ. Crucially, as demonstrated in Section 5 and also (Shih et al., 2023), fixed-point iteration (10) typically requires significantly less than T steps to generate a sample matching the one obtained via autoregressive procedure (6), thus accelerating the sampling process.

Notably, the selection of order k for the nonlinear equations influences the computational graph in the fixed-point iteration (10)—determining the number of variables from later timesteps that are employed to update the variables from earlier timesteps. We will explore the effect of order k on the convergence of the fixed-point iteration in Section 2.3 with greater details.

2.1. Stopping Criterion

To examine the convergence of the fixed-point iteration (10), we can employ the residuals of the nonlinear equations (8) for a stopping criterion. Furthermore, given the equivalence of nonlinear equations (8) across different orders k, a universal stopping criterion is applicable for all. In this study, we choose to use the residuals of the first-order equations for the stopping criterion. Specifically, the residual for the t-th equation in (8) is defined as:

$$r_{t-1} \overset{\text{def.}}{=} \|x_{t-1} - a_t x_t - b_t\epsilon_\theta(x_t, t) - c_{t-1}\xi_{t-1}\|_2^2 \quad (11)$$

Owing to the triangular structure of (8), for any $0 < t \le T$, we can conclude the convergence of the variables $x_{t-1}, ..., x_{T-1}$ if the conditions $r_{t-1} \le \varepsilon_{t-1}, ..., r_{T-1} \le \varepsilon_{T-1}$ are met, where $\varepsilon_0, ..., \varepsilon_{T-1}$ represent predetermined time-dependent thresholds. Following previous research (Shih et al., 2023), we set ε_t to $\tau^2 g^2(t) d$, with τ as the tolerance hyperparameter, d as the data dimension, and $g(t)$ as the diffusion coefficient from (1). Once the variables $x_{t-1}, ..., x_{T-1}$ have converged, further updates are unnecessary, and they can remain fixed.

2.2. Saving Computation By Solving Subequations

When T is large, computing $\epsilon_\theta(x_1^i, 1), ..., \epsilon_\theta(x_T^i, T)$ simultaneously may demand substantial memory. To address this, prior work (Shih et al., 2023) introduced the concept of a sliding window—solving only a lower triangular subequations in (8) at a time. For instance, with a window size w, one could initially iterate over the variables $x_{T-w}, ..., x_{T-1}$ by resolving the corresponding subequations. Once the variables $x_{t-1}, ..., x_{T-1}$ converge, as determined by the stopping criterion detailed in Section 2.1, the iteration window can be shifted to update $x_{t-w}, ..., x_{t-1}$ through their respective subequations.

2.3. Effect of the Order of Nonlinear Equations

We have found that despite the equivalence of the nonlinear system (8) across different orders k, the order k influences the optimization landscape of the nonlinear system (8), and consequently, the convergence speed of the fixed-point iteration. It is known that the speed of convergence is associated with the Lipschitz constant of the function $F_{t-1}^{(k)}$ (Argyros & Hilout, 2013). If k is excessively large, the Lipschitz constant of $F_{t-1}^{(k)}$ could be potentially large, since it incorporates more variables, leading to instability and slower convergence. Conversely, the fixed-point iteration (10) generally requires at least $\lceil \frac{T-1}{k} \rceil$ steps to converge due to the structure of the computational graph. This is because x_{t-1} is updated using information from $x_t, ..., x_{t_k}$, meaning the initial condition $x_T = \xi_T$ can only influence x_0 after $\lceil \frac{T-1}{k} \rceil$ iterations.

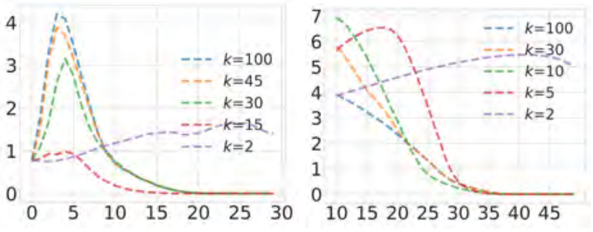
Hence, an appropriate value of k is crucial for expediting the fixed-point iteration. We examined this by running fixed-point iteration (10) under various k for the DDIM (Song et al., 2020a) and DDPM (Ho et al., 2020) sampling algorithms with 100 steps, using the DiT model (Peebles & Xie, 2023). The window size w is set to 100. Figure 1 illustrates the impact of k on the convergence of residuals $\sum_{t=1}^{T} r_{t-1}$. As observed, small values of k lead to slow convergence of residuals, whereas large k values result in instability, particularly at the beginning for DDIM with $T = 100$.

Remark 2.3. While we provide insight into how the order affects fixed-point iteration convergence, predicting the optimal k from a theoretical standpoint is generally not feasible, since the neural network ϵ_θ is a black-box. Thus, we recommend treating k as a hyperparameter and selecting the optimal one based on empirical performance. Appendix C contains grid search results on the effect of k on the convergence speed for different sampling algorithms.

Remark 2.4. It is noteworthy that the PL iteration employed by prior work (Shih et al., 2023) is equivalent to applying a fixed-point iteration to solve the nonlinear equations (8) with order k equal to the chosen window size w, and thus it corresponds to the $k = 100$ in Figure 1.

3. Anderson Acceleration for Triangular Nonlinear Equations

Anderson Acceleration (AA) (Anderson, 1965) is a classical method for expediting fixed-point iterations, which is extensively utilized across various engineering disciplines (Walker & Ni, 2011). The central idea of AA is to leverage information from previous iterations to approximate the inverse Jacobian of the nonlinear system and to implement a Newton-like update using this approximation. In this section, we explore the use of AA within the context of parallel sampling of diffusion models and the resolution of triangular nonlinear systems. First of all, let us describe a straightforward implementation of standard AA for the fixed-point iteration (10) with the use of up to m $(m \geq 1)$ previous iterations. With the initialization $x_0^0, ..., x_{T-1}^0$, the process begins with a standard fixed-point iteration as indicated by (10). For the i-th iteration with $i \geq 1$, we introduce the following notations.



(a) DDIM 100 steps

(b) DDPM 100 steps

Figure 1. Convergence of residuals under different orders. x-axis is the iteration steps while y-axis is the value of $\sum_{t=1}^{T} r_{t-1}$.

Notations. Let $m_i = \min\{m, i\}$, $\Delta x_t^i = x_t^{i+1} - x_t^i$, $\mathcal{X}_t^i = \left[\Delta x_t^{i-m_i}, ..., \Delta x_t^{i-1}\right]$, $R_t^i = F_t^{(k)}\left(x_{t+1}^i, ..., x_{(t+1)_k}^i\right) - x_t^i$, $\Delta R_t^i = R_t^{i+1} - R_t^i$, $\mathcal{F}_t^i = \left[\Delta R_t^{i-m_i}, ...\Delta R_t^{i-1}\right]$. For any $0 \leq t_1 \leq t_2 < T$ and any vectors/matrixes $v_1, ..., v_{T-1}$, we denote $v_{t_1:t_2} = \left[v_{t_1}^\top, ..., v_{t_2}^\top\right]^\top$. For any matrix V, we denote $V[i:j, t:s]$ as the submatrix of V with rows $i, ..., j$ and columns $t, ..., s$. If j and s are not specified, denote $j = T-1$ and $s = T-1$.

Assuming that the subequations in (8) for $t = t_1, ..., t_2$ are being solved and that $\mathcal{F}_{t_1:t_2}^{i\top} \mathcal{F}_{t_1:t_2}^i$ has full rank, the update rule for (AA) is provided by the following equation:

$$x_{t_1:t_2}^{i+1} = x_{t_1:t_2}^i - G^i R_{t_1:t_2}^i, \tag{12}$$

where G^i is considered an approximate inverse Jacobian of $R_{t_1:t_2}^i$, and is computed as follows:

$$G^i = -I + (\mathcal{X}_{t_1:t_2}^i + \mathcal{F}_{t_1:t_2}^i)(\mathcal{F}_{t_1:t_2}^{i\top} \mathcal{F}_{t_1:t_2}^i)^{-1} \mathcal{F}_{t_1:t_2}^{i\top} \tag{13}$$

The justification for (13) is that G^i satisfies the **Inverse Multisecant Condition** (Fang & Saad, 2009):

$$G^i \mathcal{F}_{t_1:t_2}^i = \mathcal{X}_{t_1:t_2}^i, \tag{14}$$

and the Frobenius norm $\left\|G^i + I\right\|_F$ is the smallest possible for all matrices meeting this condition (14) (Walker & Ni, 2011). It is evident from (12) that when G^i is set to $-I$, the AA update simplifies to the standard fixed-point iteration.

3.1. Triangular Anderson Acceleration

We identified a critical issue in the update rule of the AA as given in (12): For some timesteps $j < t$, the update of x_t^{i+1} could be influenced by the value of x_j^i due to the matrix G^i potentially being dense. This has occasionally led to numerical instability in our practices[1]. To understand this instability, we find that x_t always converges before x_j[2], which suggests that using the state of x_j to update x_t can be counterproductive, particularly when x_t is near convergence but x_j is not.

Armed with this key observation, we propose an adapted version of AA that is well-suited for triangular nonlinear equations like (8). The principal idea is to constrain the matrix G^i in (12) to be *block upper triangular*—A formal definition is given as follows:

Definition 3.1 (Block Upper Triangular Matrix)**.** Consider a matrix $G \in \mathbb{R}^{(t_2-t_1)d\times(t_2-t_1)d}$. We define G as block upper triangular if, for any $t_1 \le t \le t_2$, $j \le (t-t_1)d$, and $1 \le s \le d$, it holds that $G[(t-t_1)d+s, j] = 0$.

By doing so, the updated value x_t^{i+1} in (12) receives information exclusively from those x_j^i with $j \ge t$. In the subsequent theorem, we present a closed-form solution that fulfills both the inverse multisecant condition (14) and the block upper triangular stipulation as defined in Definition 3.1, while also being optimally close to $-I$ with respect to the Frobenius norm.

Theorem 3.2. *Assume $m < d$ and that $\mathcal{F}_{t_2}^{i\top}\mathcal{F}_{t_2}^i$ has full rank. Let $Q^i \in \mathbb{R}^{(t_2-t_1)d\times(t_2-t_1)d}$ be a block upper triangular matrix, and for any $t_1 \le t \le t_2$:*

$$Q^i\left[t' : t'', t' :\right] = (\mathcal{X}_t^i + \mathcal{F}_t^i)(\mathcal{F}_{t:t_2}^{i\top}\mathcal{F}_{t:t_2}^i)^{-1}\mathcal{F}_{t:t_2}^{i\top}, \quad (15)$$

where $t' \overset{\text{def.}}{=} (t-t_1)d+1$ and $t'' \overset{\text{def.}}{=} (t-t_1)d+d$. Then the matrix $T^i = -I + Q^i$ meets both the inverse multisecant condition (14) and the block upper triangular requirement from Definition 3.1, and $\left\|T^i + I\right\|_F$ is minimal among all matrices that comply with these conditions.

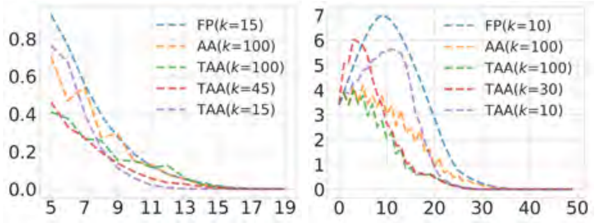
Employing the T^i derived from Theorem 3.2, we introduce a tailored update rule for AA in the context of triangular nonlinear equations: $x_{t_1:t_2}^{i+1} = x_{t_1:t_2}^i - T^i R_{t_1:t_2}^i$, and we refer this method as **Triangular Anderson Acceleration** (**TAA**). In this study, we do not undertake a detailed theoretical analysis on TAA. This omission is because even the theoretical aspects of standard AA are still actively being researched in the field of optimization (Evans et al., 2020; Rebholz & Xiao, 2023). Instead, we concentrate on assessing the empirical performance of this new type of Anderson Acceleration approach.

Figure 2 shows the results of comparing fixed-point iteration, AA, and TAA in the same scenario as Figure 1. We observe that both AA and TAA improve upon the optimal fixed-point iteration from Figure 1 by a large margin, regardless of the k used. Moreover, TAA is notably faster than AA, especially for the DDPM with 100 steps, and it remains stable even when using 16-bit precision for calculations. Additionally, similar to the fixed-point iteration, TAA can also benefit from selecting an optimal k.

Remark 3.3. In practice, we utilize $(\mathcal{F}_{t:t_2}^{i\top}\mathcal{F}_{t:t_2}^i + \lambda I)^{-1}$ with $\lambda > 0$ being a small constant, for stabilizing the computation of T^i in (15).

Remark 3.4. Apart from the method for determining T^i as outlined in Theorem 3.2, we also explored a heuristic approach to acquire a block upper triangular matrix by directly extracting the upper triangular portion of G^i from (13). While this method also enhances standard AA, it still faced numerical instability and was less effective compared to the approach using T^i from Theorem 3.2. Further details are available in Appendix B.

Remark 3.5. The computation of the matrix T^i in Theorem 3.2 adds only minimal computational and memory overhead to the standard fixed-point iteration (10). Firstly, the storage for the history matrices $\mathcal{F}_{t_1:t_2}^i$ and $\mathcal{X}_{t_1:t_2}^i$, of dimension $(t_2-t_1)d \times m_i$, is neglectable compared to that of the neural network ϵ_θ. Secondly, the operations in (15) consist of simple matrix multiplication and inversion; the matrix $\mathcal{F}_{t:t_2}^{i\top}\mathcal{F}_{t:t_2}^i \in \mathbb{R}^{m_i\times m_i}$ can be efficiently computed as the value of m is typically chosen to be between 2 and 5.



(a) DDIM 100 steps (b) DDPM 100 steps

Figure 2. Convergence of FP, AA, TAA under different k.

3.2. Safeguarding Triangular Anderson Acceleration

Fixed-point iteration, as described in (10), is known to converge within T steps for triangular systems like (8) (see Proposition 1 in (Song et al., 2021)). Unfortunately, neither the original AA nor the TAA possesses this worst-case convergence guarantee. To address this, we have identified a sufficient condition for the general update rule in the form of (12) to ensure convergence within T steps.

Theorem 3.6. *Consider a general update rule: In the i-th iteration, the update is $x_{0:T-1}^{i+1} = x_{0:T-1}^i - G^i R_{0:T-1}^i$, with G^i being any arbitrary matrix. If for any j where $R_{j+1}^i = \ldots = R_T^i = \mathbf{0}$*[3]*, the matrix G^i satisfies $G^i[jd:, jd:] = -I$,*

[1] Specifically, we have observed instances of numerical overflow when applying AA with 16-bit precision.

[2] Refer to Figure 6a in Appendix B for empirical evidence.

[3] Note that $R_T^i \equiv \mathbf{0}$ since the initial variable x_T is set to ξ_T.

then the update rule will converge within T steps.

Please see Appendix A for the proof and a more detailed explanation of this theorem. In practice, we impose this condition from Theorem 3.6 on the TAA update rule by post-processing. Fortunately, this post-processing step applied to the matrix T^i has virtually no impact on the empirical performance of TAA in our experiments. Additional details can be found in Appendix B.

4. Early Stopping and Initialization

This section introduces two practical techniques that can further accelerate the fixed-point iteration.

4.1. Early Stopping

In our experiments, we observe that high-quality images are often produced much earlier than the residuals of fixed-point iteration meet the stopping criterion. As an example, while using TAA with the fixed-point iteration for DDIM 100 steps, we find that high-quality images, nearly indistinguishable from those generated sequentially, can appear as early as step $7 \sim 11$. In contrast, the stopping criterion is typically met at around step $13 \sim 17$. More demonstrations on this point can be found in Section 5 and Appendix D. Consequently, it is feasible to halt the parallel sampling process whenever a satisfactory image is obtained.

From a practical standpoint, such early termination is easy to implement, particularly for tasks like interactive image generation where users can judge the quality of the current image and decide when to terminate the iteration process.

4.2. Initialize from Existing Sampling Trajectory

To further accelerate parallel sampling, one can initialize the fixed-point iteration using an existing sampling trajectory with a similar input condition. The underlying principle is that if two nonlinear equations are similar, the solution to one can serve as the initial point for solving the other. For instance, in text-to-image generation with two similar prompts P1 and P2, if we have already obtained a sampling trajectory $x_0, ..., x_{T-1}$ for P1, we can use this to initialize parallel sampling for P2. This is a common scenario as users often adjust prompts to achieve the desired image, leading to a wealth of available trajectories for initialization. Additionally, akin to the method in SDEdit (Meng et al., 2022), one can also fix the later steps of the trajectory (e.g., $x_{T_{\text{init}}}, ..., x_{T-1}$) when initializing sampling for P2, and only update the earlier steps (e.g., $x_0, ..., x_{T_{\text{init}}-1}$). This ensures the resulting image to be close to the one from prompt p_1.

As we will show in Section 5.3 and Appendix F, starting from an existing sampling trajectory can greatly reduce the steps needed for parallel sampling to converge. Furthermore, this method often transforms the image to abide the new prompt in a smooth way if T_{init} is properly set.

With all the techniques discussed in Sections 2, 3, and here, the complete version of our proposed algorithm, ParaTAA, is summarized in Algorithm 1.

Algorithm 1 ParaTAA: Parallel Sampling of Diffusion Models with Triangular Anderson Acceleration

```
Require: Diffusion model $\epsilon_\theta$, $k$-th order nonlinear equations $\left[F_0^{(k)}, ..., F_{T-1}^{(k)}\right]$, histroy size $m$, tolerance $\tau$, diffusion coefficients $g(t)$, window size $w$, initialization $x_{0:T-1}^0$ and $x_T$, fixed initaizliation steps $T_{\text{init}}$, maximum iteration steps $s_{\max}$.
$t_1, t_2 \leftarrow \max\{0, T_{\text{init}} - w\}, T_{\text{init}} - 1$
for $s = 1$ to $s = s_{\max}$ do
    Compute $\epsilon_\theta(x_{t+1}^{s-1}, t+1), t = t_1, ..., t_2$ in parallel.
    Compute the residuals $r_{t_1:t_2}$ as (11).
    Update $t_2 \leftarrow \max\{t_1 \le t \le t_2 | r_t > \tau g^2(t) d\}$
    if $t_2$ is Null then
        Break loop
    end if
    Update $t_1 \leftarrow \max\{0, t_2 - w\}$
    Compute and store $R_{t_1:t_2}^{s-1}$, $\mathcal{X}_{t_1:t_2}^{s-1}$, $\mathcal{F}_{t_1:t_2}^{s-1}$ as in Sec. 3.
    Compute $T^{s-1}$ as in Theorem 3.2 and 3.6, do
        $$x_{t_1:t_2}^s = x_{t_1:t_2}^{s-1} - T^{s-1} R_{t_1:t_2}^{s-1}$$
end for
Return $x_{0:T-1}^s$.
```

5. Experiments

5.1. Accelerating Image Diffusion Sampling

In this section, we present the effectiveness of our approach in accelerating the sampling process for two prevalent diffusion models: DiT (Peebles & Xie, 2023), a class-conditioned diffusion model trained on the Imagenet dataset at a resolution of 256x256, and text-conditioned Stable Diffusion v1.5 (SD) (Rombach et al., 2022) with a resolution of 512x512.

Scenarios. We consider accelerating four typical sequential sampling algorithms: Euler-type ODE sampling algorithm DDIM (Song et al., 2020a) with 25, 50, and 100 steps, respectively, and SDE sampling algorithm DDPM (Ho et al., 2020)[4] with 100 steps.

Algorithms. We compare our proposed algorithm ParaTAA with two baselines: (1) The fixed-point (FP) iteration (10) with the order of equations k equal to the window-size w, equivalent to the method proposed in (Shih et al., 2023), and (2) The fixed-point iteration (10) with the optimal order of equations k determined by grid search, referred as FP+. ParaTAA has two hyperparameters: the history size m and

[4] Following (Song et al., 2020a), we treat DDIM with $\eta = 1$ as a DDPM sampler.

the order k, both of which are chosen via grid search. For hyperparameter analysis in all four tested scenarios, please refer to Appendix C. For all algorithms, we use the same stopping threshold $\varepsilon_t = \tau^2 g^2(t) d$ with $\tau = 10^{-3}$, and initialize all variables with standard Gaussian Distribution.

Setting. We run these experiments using 8 A800 GPUs, each with 80GB of memory. We set the window size w to match the number of sampling steps for all scenarios, except in the case of the DDPM 100 steps for SD, where we select $w = 40$ to aim for an acceptable wall-clock time speedup. In all scenarios, we employ classifier-free guidance (Ho & Salimans, 2022) with a guidance scale of 5.

Evaluation. For DiT models, we assess the quality of sampled images using the FID score (Heusel et al., 2017) and the Inception Score (IS) (Salimans et al., 2016) with 5000 generated samples. For SD models, we generate random text prompts combining a color and an animal, such as ”green duck,” and evaluate the quality of the sampled images by computing the CLIP Score (CS) (Radford et al., 2021) with 1000 samples.

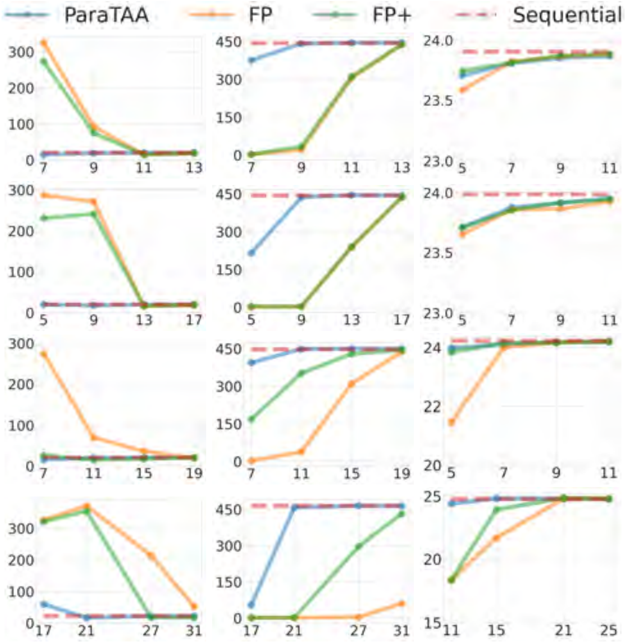
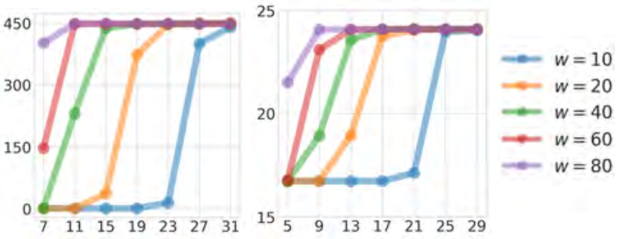


Figure 3. Comparison of parallel sampling methods and sequential sampling across various scenarios. The x-axis for all plots represents the maximum number of steps, $s_{\max}$. The first two columns from the left show the FID and IS scores for the DiT model, respectively, while the third column depicts the CS for the SD model. The rows, from top to bottom, correspond to the scenarios with DDIM 25 steps, DDIM 50 steps, DDIM 100 steps, and DDPM 100 steps, respectively. For visual examples of generated images related to these results, please refer to Appendix D.



(a) DiT - IS

(b) SD - CS

Figure 4. Convergence of ParaTAA under different window sizes. The x-axis and y-axis are the same as Figure 3

Results. Our primary findings are detailed in Figure 3 and Table 1, offering several insightful observations. Firstly, Figure 3 corroborates that early-stopping is a valid approach[5]. Across all algorithms, the quality metrics of the generated images match those of sequentially sampled images in significantly fewer steps. By comparing FP and FP+, we can clearly see the importance of choosing a properly order k for nonlinear equations (8). Furthermore, our proposed ParaTAA outperforms both fixed-point algorithms substantially, underscoring the effectiveness of our Triangular Anderson Acceleration technique. In Table 1, we encapsulate the key outcomes from Figure 3, including data on wall-clock time and inference steps. Notably, ”Steps” in Table 1 refers to the number of parallelizable inference steps for the neural network ϵ_θ. It is evident that all parallel sampling algorithms greatly reduce inference steps, particularly for larger T scenarios, with ParaTAA consistently spending the fewest steps in every case and cutting down the steps required by sequential sampling by **4∼14x**! Additionally, it is apparent that, generally, DDPM needs more steps to converge compared to DDIM. In terms of wall-clock time speedup, ParaTAA can achieve a **1.5∼2.9x** improvement.

Remark 5.1. The wall-clock time reported in Table 1 can be further enhanced with optimized implementation, computing devices, and inter-GPU communication environments. Theoretically, the achievable speedup is determined by the ratio of inference steps required by sequential versus parallel sampling, ranging from 4 to 14 times as discussed earlier.

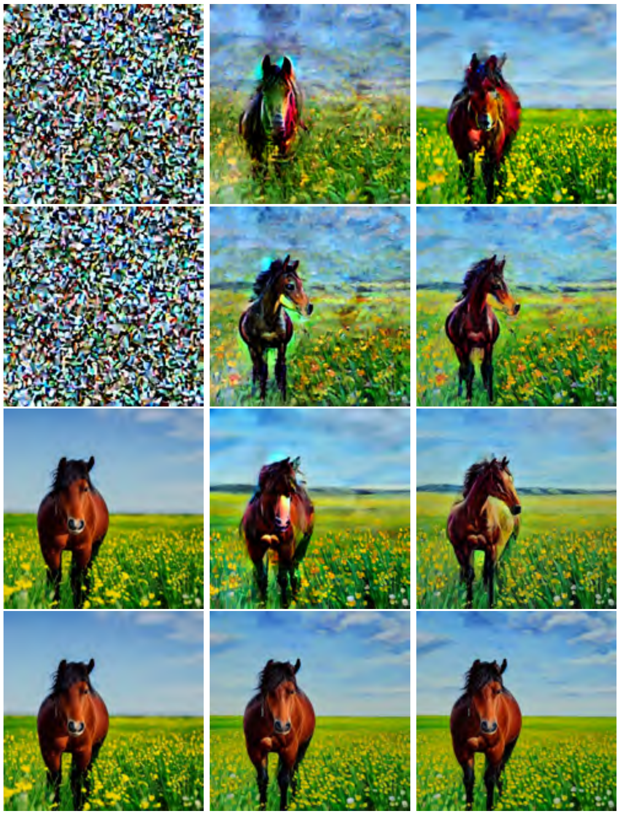
Remark 5.2. Our own implementation of the fixed-point iteration achieves results comparable to those in (Shih et al., 2023). However, we opted not to adjust the stopping criterion, as we observed it impacts the uniqueness of the generated image. In our SD experiments, we used 16-bit precision instead of the 32-bit used in (Shih et al., 2023), which made our measured wall-clock time significantly faster.

Remark 5.3. A key advantage of parallel sampling over other acceleration methods is its ability to produce images that are (almost) identical to those from sequential sampling. Theorem 2.2 provides a guarantee for this assertion. For real examples on this point, please refer to Appendix D.

[5] For the information on the number of inference steps to reach the stopping criterion, we refer readers to Appendix C.

Method	DiT DDIM-25				DiT DDIM-50				SD DDIM-25			SD DDIM-50		
	Steps	Time	FID ↓	IS↑	Steps	Time	FID ↓	IS↑	Steps	Time	CS↑	Steps	Time	CS↑
Sequential	25	0.41s	20.5	442.6	50	0.84s	20.3	443.4	25	0.73s	23.9	50	1.44s	24.0
FP	17.8	0.42s	19.8	441.2	21.6	0.69s	20.2	442.0	14.1	0.98s	23.8	15.7	1.36s	24.0
FP+	13	0.32s	18.7	436.5	17	0.58s	18.7	436.8	**7**	**0.62s**	23.8	**7**	**0.93s**	23.9
ParaTAA	**9**	**0.25s**	18.8	441.0	**9**	**0.34s**	19.1	441.9	**7**	0.63s	23.8	**7**	**0.93s**	23.9
	DiT DDIM-100				DiT DDPM-100				SD DDIM-100			SD DDPM-100		
Sequential	100	1.65s	20.6	446.9	100	1.69s	22.7	464.8	100	2.95s	24.2	100	2.98s	24.8
FP	23.0	0.98s	19.7	444.2	42.3	1.90s	21.4	459.6	15.8	2.16s	24.2	28.9	3.23s	24.8
FP+	19	0.81s	19.8	443.7	31	1.29s	17.0	432.3	**7**	1.56s	24.2	21	2.45s	24.5
ParaTAA	**11**	**0.56s**	20.0	448.3	**21**	**0.95s**	22.1	457.8	**7**	**1.53s**	24.2	**15**	**1.97s**	24.8

Table 1. Performance comparison of various parallel sampling methods across different scenarios. It should be noted that for FP+ and ParaTAA, the early-stopping step is selected based on insights from Figure 3, whereas for FP, early-stopping is not employed, and the step value indicates the average number of inference steps required to satisfy the stopping criterion.



(a) Initialization (b) After 3 steps (c) After 5 steps

Figure 5. Iterations of ParaTAA with different initializations. The rows from top to bottom shows: 1. Sampling with P1 with random initialization; 2. Sampling with P2 with random initialization. 3. Sampling with P2 with trajectory of P1 as initialization and $T_{\text{init}} = 50$. 4. Same as 3 except that $T_{\text{init}} = 35$. For optimal viewing, please zoom in on the figure.

5.2. Effect of Window Size

In this section, we examine how the window size w affects the trade-off between convergence and computation on the DDIM 100 steps scenario for both DiT and SD models, testing ParaTAA with varying window sizes.

Results. As depicted in Figure 4, the relationship between the increase in window size and the reduction in inference steps is not proportional. For instance, with SD, at $w = 10$, ParaTAA needs 25 steps to achieve the desired CS level, which is 4x fewer than that of sequential sampling. However, when we double the computation by setting $w = 20$, the inference steps reduce only marginally to 21. This implies that users should select a window size that balances convergence speed and computational effort to optimize wall-clock time speedup.

5.3. Initialization from Existing Trajectory

This section explores the impact of initializing parallel sampling using a pre-existing trajectory through a case study. We conduct an experiment utilizing the SD model with DDIM 50 steps and two similar prompts, P1: ”A 4k detailed photo of a horse in a field of flowers”, P2: ”An oil painting of a horse in a field of flowers”. Our objective is to investigate the difference in image generation for P2 when starting from a random initialization versus using the trajectory of P1. Additionally, we assess how the initial step count, T_{init}, influences the convergence of sampling for P2 with this initialization.

Results. As shown in Figure 5, using random initialization for prompts P1 and P2, ParaTAA does not yield high-quality images within the first 5 steps. In contrast, initializing the sampling of P2 with the trajectory from P1 results in a considerably better image by the 5th step. By setting T_{init} to 35, ParaTAA manages to produce a good image by the 3rd step, with a smooth transition from the initial image. Hence, we conclude that starting parallel sampling with an existing trajectory can significantly decrease the number of inference steps needed for the sampling process to converge. For an extended and quantitative evaluation of these findings, please refer to Appendix E.

6. Conclusion

In this study, we frame parallel sampling for diffusion models as solving a system of triangular nonlinear equations. We introduce a novel parallel sampling algorithm, ParaTAA, which can substantially decrease the inference steps required by sequential sampling while maintaining image quality. Moreover, the triangular Anderson acceleration technique developed in this work could be a subject of independent interest, and we expect that the optimization research community will be interested in further exploring its theoretical aspects in the near future.

While this work primarily demonstrates the acceleration for image diffusion models, we anticipate that our proposed method could have broader applications on tasks that involve an autoregressive process, and one notable example is autoregressive video generative models in (Ho et al., 2022; Esser et al., 2023; Gupta et al., 2023).

Currently, for large models like SD, ParaTAA requires the use of multiple GPUs to achieve considerable speedup in wall-clock time. Nonetheless, as advancements in GPU technology and parallel computing infrastructures evolve, we anticipate that the cost will be significantly lower and ParaTAA will become increasingly important for accelerating the sampling of large-scale diffusion models.

Acknowledgements

This work was supported by Alibaba Group through Alibaba Research Intern Program. The work was also supported by Shenzhen Science and Technology Program under Grant No. ZDSYS20230626091302006 and RCJC20210609104448114, and by Guangdong Provincial Key Laboratory of Big Data Computing.

Impact Statement

This work focuses on accelerating the sampling process of existing diffusion generative models. As far as we can see, there is no foreseeable negative impact on the society.

References

Anderson, D. G. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

Anonymous. T-stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=rnHqwPH4TZ. under review.

Argyros, I. K. and Hilout, S. *Computational methods in nonlinear analysis: efficient algorithms, fixed point theory and applications*. World Scientific, 2013.

Chen, Y., Liu, C., Huang, W., Cheng, S., Arcucci, R., and Xiong, Z. Generative text-guided 3d vision-language pretraining for unified medical image segmentation. *arXiv preprint arXiv:2306.04811*, 2023.

Chen, Y., Liu, C., Liu, X., Arcucci, R., and Xiong, Z. Bimcv-r: A landmark dataset for 3d ct text-image retrieval. *arXiv preprint arXiv:2403.15992*, 2024.

Dang, B., Zhao, W., Li, Y., Ma, D., Yu, Q., and Zhu, E. Y. Real-time pill identification for the visually impaired using deep learning. *arXiv preprint arXiv:2405.05983*, 2024.

Dennis, Jr, J. E. and Schnabel, R. B. Least change secant updates for quasi-newton methods. *Siam Review*, 21(4): 443–459, 1979.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Ding, Z., Li, P., Yang, Q., Shen, X., Li, S., and Gong, Q. Regional style and color transfer. *arXiv preprint arXiv:2404.13880*, 2024.

Esser, P., Chiu, J., Atighehchian, P., Granskog, J., and Germanidis, A. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7346–7356, 2023.

Evans, C., Pollock, S., Rebholz, L. G., and Xiao, M. A proof that anderson acceleration improves the convergence rate in linearly converging fixed-point methods (but not in those converging quadratically). *SIAM Journal on Numerical Analysis*, 58(1):788–810, 2020.

Fang, H.-r. and Saad, Y. Two classes of multisecant methods for nonlinear acceleration. *Numerical linear algebra with applications*, 16(3):197–221, 2009.

Feng, X., Wang, C., Wu, C., Li, Y., He, Y., Wang, S., and Wang, Y. Fdnet: Feature decoupled segmentation network for tooth cbct image. *arXiv preprint arXiv:2311.06551*, 2023.

Geng, Z., Pokle, A., and Kolter, J. Z. One-step diffusion distillation via deep equilibrium models. *arXiv preprint arXiv:2401.08639*, 2023.

Gupta, A., Yu, L., Sohn, K., Gu, X., Hahn, M., Fei-Fei, L., Essa, I., Jiang, L., and Lezama, J. Photorealistic video generation with diffusion models. *arXiv preprint arXiv:2312.06662*, 2023.

Hao, Y., Chi, Z., Dong, L., and Wei, F. Optimizing prompts for text-to-image generation. *arXiv preprint arXiv:2212.09611*, 2022.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, P., Yang, Q., Geng, X., Zhou, W., Ding, Z., and Nian, Y. Exploring diverse methods in visual question answering. *arXiv preprint arXiv:2404.13565*, 2024a.

Li, Z., Guan, B., Wei, Y., Zhou, Y., Zhang, J., and Xu, J. Mapping new realities: Ground truth image creation with pix2pix image-to-image translation. *arXiv preprint arXiv:2404.19265*, 2024b.

Lim, Y. H., Zhu, Q., Selfridge, J., and Kasim, M. F. Parallelizing non-linear sequential models over the sequence length. *arXiv preprint arXiv:2309.12252*, 2023.

Liu, X., Zhang, X., Ma, J., Peng, J., and Liu, Q. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

Meng, C., Rombach, R., Gao, R., Kingma, D., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Mo, Y., Qin, H., Dong, Y., Zhu, Z., and Li, Z. Large language model (llm) ai text generation detection based on transformer deep learning algorithm. *International Journal of Engineering and Management Research*, 14 (2):154–159, 2024.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rebholz, L. G. and Xiao, M. The effect of anderson acceleration on superlinear and sublinear convergence. *Journal of Scientific Computing*, 96(2):34, 2023.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Sauer, A., Lorenz, D., Blattmann, A., and Rombach, R. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

Shih, A., Belkhale, S., Ermon, S., Sadigh, D., and Anari, N. Parallel sampling of diffusion models. *arXiv preprint arXiv:2305.16317*, 2023.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Song, X., Wu, D., Zhang, B., Peng, Z., Dang, B., Pan, F., and Wu, Z. Zeroprompt: Streaming acoustic encoders are zero-shot masked lms. *arXiv preprint arXiv:2305.10649*, 2023a.

Song, X., Wu, D., Zhang, B., Zhou, D., Peng, Z., Dang, B., Pan, F., and Yang, C. U2++ moe: Scaling 4.7 x parameters with minimal impact on rtf. *arXiv preprint arXiv:2404.16407*, 2024.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Song, Y., Meng, C., Liao, R., and Ermon, S. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, pp. 9791–9800. PMLR, 2021.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023b.

Sun, Z., Suresh, A. T., Ro, J. H., Beirami, A., Jain, H., and Yu, F. Spectr: Fast speculative decoding via optimal transport. *arXiv preprint arXiv:2310.15141*, 2023.

Tang, Z., Rybin, D., and Chang, T.-H. Zeroth-order optimization meets human feedback: Provable learning via ranking oracles. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=TVDUVpgu9s.

Tang, Z., Wang, Y., and Chang, T.-H. z-signfedavg: A unified stochastic sign-based compression for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15301–15309, 2024b.

Walker, H. F. and Ni, P. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

Wang, H., Tang, Z., Zhang, S., Shen, C., and Chang, T.-H. Embracing uncertainty: A diffusion generative model of spectrum efficiency in 5g networks. In *2023 International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 880–885. IEEE, 2023.

Wu, C., Wang, C., Wang, Y., Zhou, H., Zhang, Y., Wang, Q., and Wang, S. Mmfusion: Multi-modality diffusion model for lymph node metastasis diagnosis in esophageal cancer. *arXiv preprint arXiv:2405.09539*, 2024.

Xin, Y., Du, J., Wang, Q., Lin, Z., and Yan, K. Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16085–16093, 2024a.

Xin, Y., Du, J., Wang, Q., Yan, K., and Ding, S. Mmap: Multi-modal alignment prompt for cross-domain multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16076–16084, 2024b.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Zhang, D., Zhou, F., Wei, Y., Yang, X., and Gu, Y. Unleashing the power of self-supervised image denoising: A comprehensive review. *arXiv preprint arXiv:2308.00247*, 2023.

Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023.

A. Proof

Proof of Theorem 2.2. Initially, it is simple to verify that the sequential sampling procedure (6) has a unique solution. Considering the initial conditions $x_T = y_T = \xi_T$, let us assume for the sake of contradiction that there exist two distinct solutions $x_{0:T-1}$ and $y_{0:T-1}$.

$$\begin{aligned} x_{t-1} &= a_t x_t + b_t \epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1},\ t = 1, ..., T, \\ y_{t-1} &= a_t y_t + b_t \epsilon_\theta(y_t, t) + c_{t-1}\xi_{t-1},\ t = 1, ..., T. \end{aligned}$$

Using an induction argument, let us assume that for some $0 < t \leq T$, we have $x_t = y_t$. Under this assumption, we can show that

$$\begin{aligned} x_{t-1} &= a_t x_t + b_t \epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1} \\ &= a_t y_t + b_t \epsilon_\theta(y_t, t) + c_{t-1}\xi_{t-1} \\ &= y_{t-1}. \end{aligned}$$

Hence the two solutions $x_{0:T-1}$ and $y_{0:T-1}$ are the same.

We will now demonstrate that for any $1 \leq k \leq T$, the nonlinear equations given by (8) are equivalent. This implies that all sets of nonlinear equations share the same unique solution, since the case of $k = 1$ corresponds to the sequential procedure outlined in (6).

For the purpose of this proof, we define two sets of nonlinear equations to be equivalent if any solution to one set is also a solution to the other, and vice versa. To simplify the exposition, we will prove the equivalence of the 1st order equations to the 2nd order equations, while noting that the proof that k-th order equations are equivalent to $(k+1)$-th order equations follows a similar procedure. Assume that $x_{0:T-1}$ is a solution to the 1st order equations. It follows directly that $x_{0:T-1}$ satisfies the 2nd order equations as well, which can be seen by (7). Conversely, if we consider $x_{0:T-1}$ as a solution to the 2nd order equations, it can be shown that

$$x_{t-1} = \begin{cases} a_t\Big(a_{t+1}x_{t+1} + b_{t+1}\epsilon_\theta(x_{t+1}, t+1) + c_t\xi_t\Big) + b_t\epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1},\ t < T, \\ a_t x_t + b_t\epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1},\ t = T. \end{cases}.$$

With $x_{T-1} = a_T x_T + b_T \epsilon_\theta(x_T, T) + c_{T-1}\xi_{T-1}$, we can show that

$$\begin{aligned} x_{T-2} = a_{T-1}&\Big(a_T x_T + b_T\epsilon_\theta(x_T, T) + c_{T-1}\xi_{T-1}\Big) + b_{T-1}\epsilon_\theta(x_{T-1}, T-1) + c_{T-2}\xi_{T-2} \\ &= a_{T-1}x_{T-1} + b_{T-1}\epsilon_\theta(x_{T-1}, T-1) + c_{T-2}\xi_{T-2}. \end{aligned}$$

With the same procedure, we can show that $x_{t-1} = a_t x_t + b_t\epsilon_\theta(x_t, t) + c_{t-1}\xi_{t-1}$ for $t = 1, ..., T-2$. Hence, $x_{0:T-1}$ is a solution of 1st order equations.

□

Proof of Theorem 3.2. Since $T^i = -I + Q^i$, the inverse multisecant condition (14) can be written as

$$Q^i \mathcal{F}^i_{t_1:t_2} = \mathcal{X}^i_{t_1:t_2} + \mathcal{F}^i_{t_1:t_2}. \tag{16}$$

Since Q^i is a block upper triangular matrix, the condition (16) can be further simplified as

$$Q^i[t' : t'', t' :]\mathcal{F}^i_{t:t_2} = \mathcal{X}^i_{t:t_2} + \mathcal{F}^i_{t:t_2}, \quad t = t_1, t_1+1, ..., t_2. \tag{17}$$

As one can see, for each t, the linear equations $Q^i[t' : t'', t' :]\mathcal{F}^i_{t:t_2} = \mathcal{X}^i_{t:t_2} + \mathcal{F}^i_{t:t_2}$ is underdertermined, because $Q^i[t' : t'', t' :] \in \mathbb{R}^{d\times(t_2-t)d}$, $\mathcal{F}^i_{t:t_2} \in \mathbb{R}^{(t_2-t)d\times m_i}$, $\mathcal{X}^i_{t:t_2} \in \mathbb{R}^{(t_2-t)d\times m_i}$, rank$(\mathcal{F}^i_{t:t_2}) = m_i$ and $m_i = \min\{m, i\} < d$.

As a classical result in linear regression analysis (Dennis & Schnabel, 1979), the minimum-norm solution for $Q^i[t' : t'', t' :]$ is given by

$$Q^i[t' : t'', t' :] = \underset{Q\mathcal{F}^i_{t:t_2} = \mathcal{X}^i_{t:t_2} + \mathcal{F}^i_{t:t_2}}{\arg\min} \|Q\|_F \tag{18}$$

$$= (\mathcal{X}^i_t + \mathcal{F}^i_t)(\mathcal{F}^{i\top}_{t:t_2}\mathcal{F}^i_{t:t_2})^{-1}\mathcal{F}^{i\top}_{t:t_2}. \tag{19}$$

Therefore, $\left\|T^i + I\right\|_F$ is minimal among all matrices satisfying both the inverse multisecant condition (14) and the block upper triangular condition in Definition 3.1.

□

Proof of Theorem 3.6. In this proof, we aim to establish that following t iterations of the update rule, the variables $x_{T-t}, \ldots, x_{T-1}$ converge. We will demonstrate this result using inductive reasoning. At the initial step of the induction, given that R^0_T is identically zero (denoted by $R^0_T \equiv \mathbf{0}$), it follows that

$$G^0[(T-1)d :, (T-1)d :] = -I.$$

Therefore, the update of x_{T-1} is given by

$$x^1_{T-1} = x^0_{T-1} - (-I)R^0_{T-1} = x^0_{T-1} + R^0_{T-1} = x^0_{T-1} + F^{(k)}_{T-1}(x_T) - x^0_t = F^{(k)}_{T-1}(x_T).$$

Therefore, x^1_{T-1} converges and hence $R^1_{T-1} = F^{(k)}_{T-1}(x_T) - x^1_t = \mathbf{0}$. Now we suppose that after $t < T$ steps, $x_{T-t}, ..., x_{T-1}$ converges, i.e., $R^t_{T-t} = R^t_{T-1} = \mathbf{0}$. Then, we have

$$G^t[(T-t-1)d :, (T-t-1)d :] = -I.$$

Hence similarly, we have

$$\begin{aligned} x^{t+1}_{T-t-1} &= x^t_{T-t-1} - (-I)R^t_{T-t-1} = x^t_{T-t-1} + R^t_{T-t-1} \\ &= x^t_{T-t-1} + F^{(k)}_{T-t-1}(x^t_{T-t}, ..., x^t_{(T-t)_k}) - x^t_{T-t-1} \\ &= F^{(k)}_{T-t-1}(x^t_{T-t}, ..., x^t_{(T-t)_k}), \end{aligned}$$

and hence $R^{t+1}_{T-t-1} = F^{(k)}_{T-t-1}(x^{t+1}_{T-t}, ..., x^{t+1}_{(T-t)_k}) - x^{t+1}_{T-t-1} = F^{(k)}_{T-t-1}(x^t_{T-t}, ..., x^t_{(T-t)_k}) - x^{t+1}_{T-t-1} = \mathbf{0}$. Thus x_{T-t-1} converges after $T+1$ steps. By this induction, we can conclude that all the variables $x_0, ..., x_{T-1}$ converges after T steps.

□

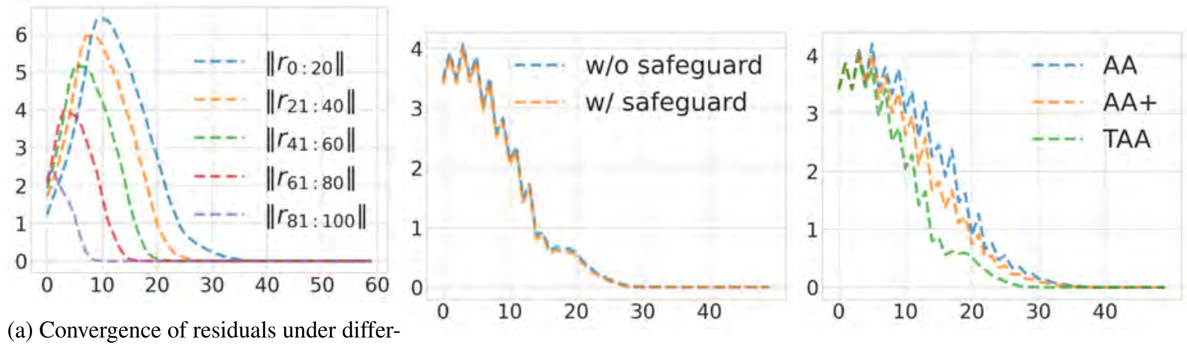
B. Further Exploration

In this section, we delve deeper into the study of the nonlinear equations (8) and the performance of Algorithm 1 in this section. For all the experiments in this section, we adopt the DDPM with 100 steps as the sequential sampling algorithm and employing DiT models. We present our findings in Figure 6.

In Figure 6a, we plot the convergence behavior of the variables $x_0, \ldots, x_{T-1}$ under the fixed-point iteration (10), and notice that their residuals do not converge uniformly. This is largely attributed to the triangular structure present in (8). Specifically, the earlier step variables $x_{81}, \ldots, x_{100}$ reach convergence within fewer than 10 steps, while the later step variables $x_0, \ldots, x_{20}$ take approximately 35 steps to converge. This observation reinforces our motivation for introducing Triangular Anderson Acceleration—to prevent the updating of near-converged variables with information from those that have not yet converged.

In Figure 6b, we examine the impact of the safeguarding technique described in Theorem 3.6. While this technique offers a worst-case guarantee, we find that it does not detract from the empirical effectiveness of Triangular Anderson Acceleration.

The third figure, Figure 6c, demonstrates that simply extracting the upper triangular portion of the original Anderson Acceleration matrix (13) (denoted as AA+), despite of an improvement over the standard Anderson Acceleration, still falls short of our proposed Triangular Anderson Acceleration. More importantly, as shown in (13), utilizing only the upper triangular component of G^i does not ensure that x^{i+1}_t in (12) is exclusively updated using information from previous iterations x^i_j where $j \geq t$. This is because the inputs from x^i_j, with $j < t$, are still incorporated into G^i through the inversion of the matrix $(\mathcal{F}^{i\top}_{t_1:t_2}\mathcal{F}^i_{t_1:t_2})^{-1}$. It is also important to note that for these experiments, we utilize 32-bit precision in our computations, as the AA and AA+ methods do not exhibit stability with 16-bit precision.



(a) Convergence of residuals under different timesteps

(b) TAA with/without Safeguarding

(c) Comparing TAA to AA+

Figure 6. More investigation on TAA.

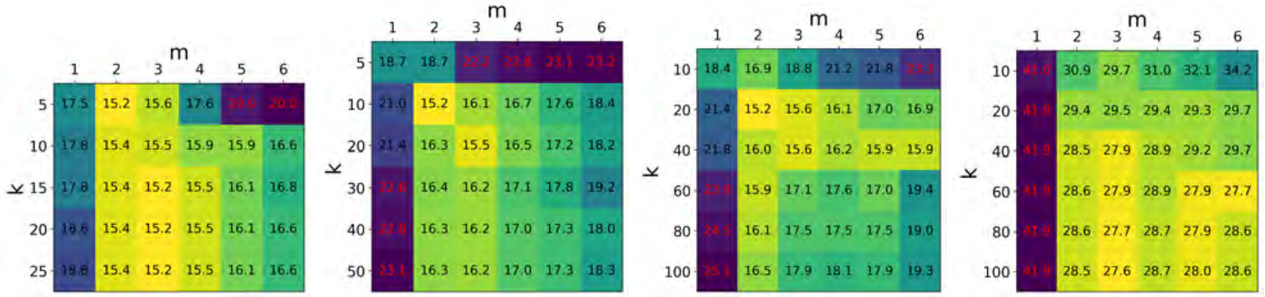
C. Hyperparameter Analysis

In this section, we present grid search results for sampling with DiT models under the four scenarios outlined in Section 5.1: DDIM with 25 steps, DDIM with 50 steps, DDIM with 100 steps, and DDPM with 100 steps. We fixed the window size w to match the total number of sequential sampling steps. The grid search is performed for the order k and history size m in Algorithm 1. We used the average number of steps required to achieve convergence for 100 different seeds as a metric to assess the performance of different hyperparameters. It is important to observe that when $m = 1$, Algorithm 1 reverts to the fixed-point iteration (10) since it does not utilize historical information. The results are summarized in Figure 7.

Based on the grid search results shown in Figure 7, we can draw several conclusions. Firstly, the optimal history size m appears to be between 2 and 4, as utilizing additional historical information may be detrimental to performance. Secondly, for $m \geq 2$, the algorithm becomes quite resilient to changes in the order k, provided that k is sufficiently large. Conversely, with $m = 1$, corresponding to fixed-point iteration, the algorithm performs best with a smaller k.

An interesting observation from Figure 7 is that for all DDIM scenarios (25, 50, and 100 steps), the ParaTAA algorithm tends to converge in roughly the same number of steps. Furthermore, we note that the DDPM typically demands more steps to reach convergence compared to the DDIM. We speculate that the inclusion of a noise term in the nonlinear equations (8) may exacerbate the optimization landscape for the fixed-point iteration.

Since the SD models exhibit a similar pattern of hyperparameters as shown in Figure 7, we choose to use the same optimal hyperparameters for both the DiT and SD experiments in Section 5.1.



k \ m	1	2	3	4	5	6
5	17.5	15.2	15.6	17.6	19.6	20.0
10	17.8	15.4	15.5	15.9	15.9	16.6
15	17.8	15.4	15.2	15.5	16.1	16.8
20	18.6	15.4	15.2	15.5	16.1	16.6
25	18.8	15.4	15.2	15.5	16.1	16.6

(a) DDIM 25 steps

k \ m	1	2	3	4	5	6
5	18.7	18.7	22.2	22.8	23.1	23.2
10	21.0	15.2	16.1	16.7	17.6	18.4
20	21.4	16.3	15.5	16.5	17.2	18.2
30	22.6	16.4	16.2	17.1	17.8	19.2
40	22.8	16.3	16.2	17.0	17.3	18.0
50	23.1	16.3	16.2	17.0	17.3	18.3

(b) DDIM 50 steps

k \ m	1	2	3	4	5	6
10	18.4	16.9	18.8	21.2	21.8	23.3
20	21.4	15.2	15.6	16.1	17.0	16.9
40	21.8	16.0	15.6	16.2	15.9	15.9
60	23.9	15.9	17.1	17.6	17.0	19.4
80	24.5	16.1	17.5	17.5	17.5	19.0
100	25.1	16.5	17.9	18.1	17.9	19.3

(c) DDIM 100 steps

k \ m	1	2	3	4	5	6
10	41.0	30.9	29.7	31.0	32.1	34.2
20	41.9	29.4	29.5	29.4	29.3	29.7
40	41.9	28.5	27.9	28.9	29.2	29.7
60	41.9	28.6	27.9	28.9	27.9	27.7
80	41.9	28.6	27.7	28.7	27.9	28.6
100	41.9	28.5	27.6	28.7	28.0	28.6

(d) DDPM 100 steps

Figure 7. Hyperparameter Analysis for ParaTAA.

D. Qualitative Comparison

This section presents a qualitative visual comparison of the convergence behaviors of ParaTAA, FP, and FP+ as illustrated in Figure 3. We feature examples from the following scenarios: DiT with DDIM at 100 steps, DiT with DDPM at 100 steps, SD with DDIM at 100 steps, and SD with DDPM at 100 steps. The images displayed showcase the convergence process at different iteration stages for each algorithm. Sequentially generated images are provided in Figure 8 for comparison, while the images generated through parallel sampling are depicted in Figures 9, 10, 11, and 12, corresponding to the four aforementioned scenarios.

As is evident from the visualizations, it is clear that our proposed ParaTAA algorithm significantly outperforms the naive fixed-point iteration (FP) and its variant with optimal order (FP+). Moreover, for both DiT and SD models with DDIM at 100 steps, ParaTAA successfully produces images of similar quality to those obtained via sequential sampling within a mere 7 iterations. In the case of DDPM at 100 steps, ParaTAA achieves comparable results to sequential sampling within only 21 iterations.

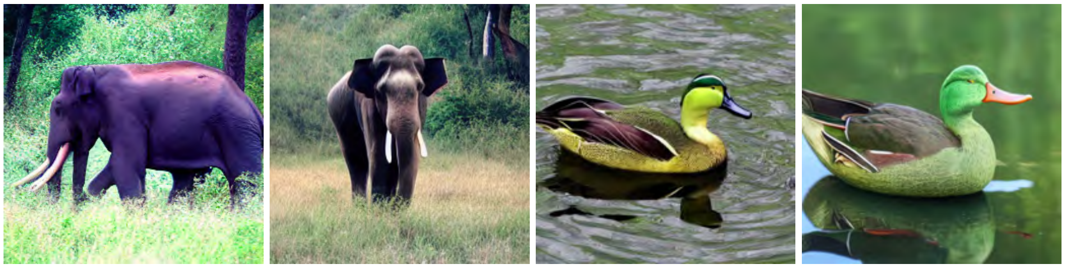

(a) DiT DDIM-100 (b) DiT DDPM 100 (c) SD DDIM 100 (d) SD DDPM 100

Figure 8. Generated Images from Sequential Sampling. For DiT model, we use the class for ”elephant” as the input condition. For SD model, we use the ”green duck” as the text prompt.

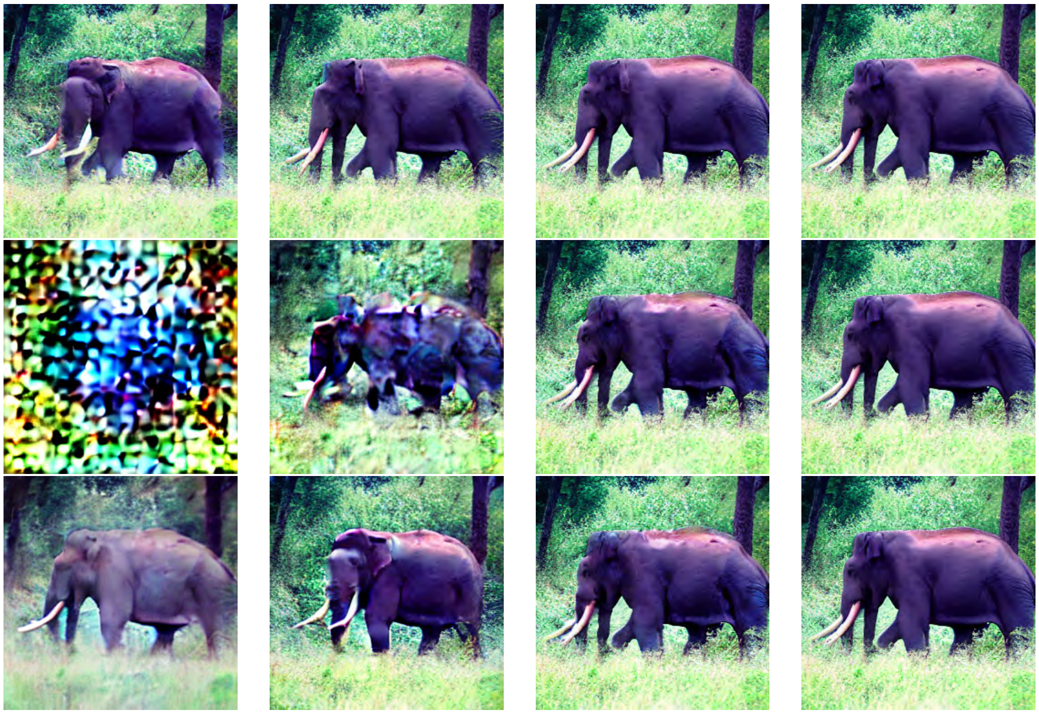
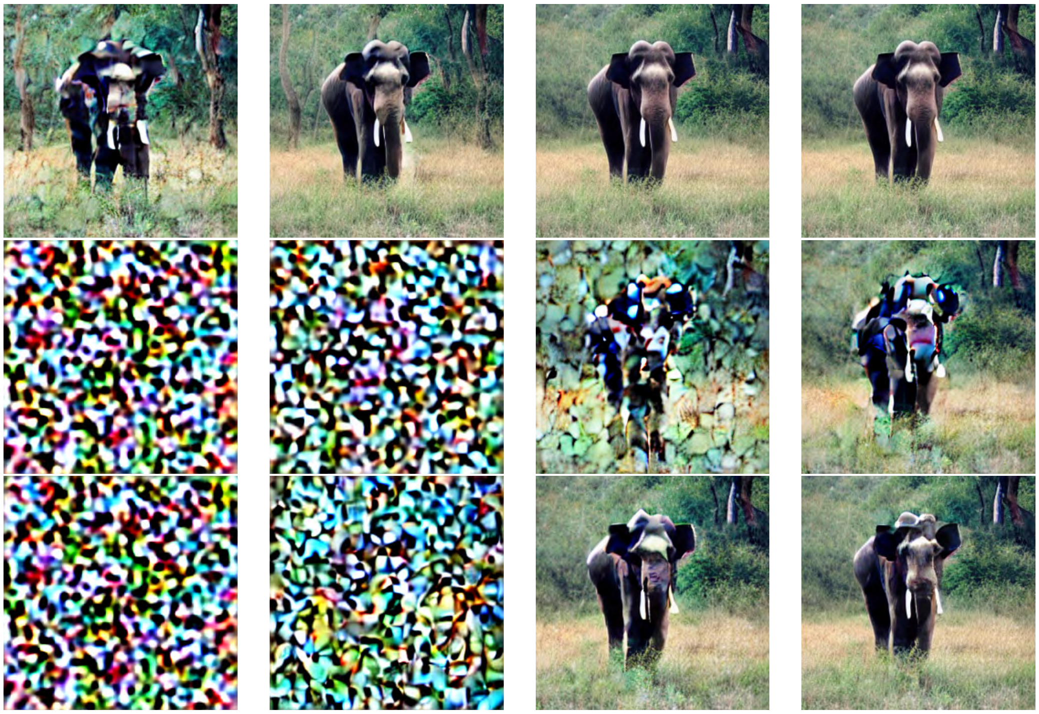
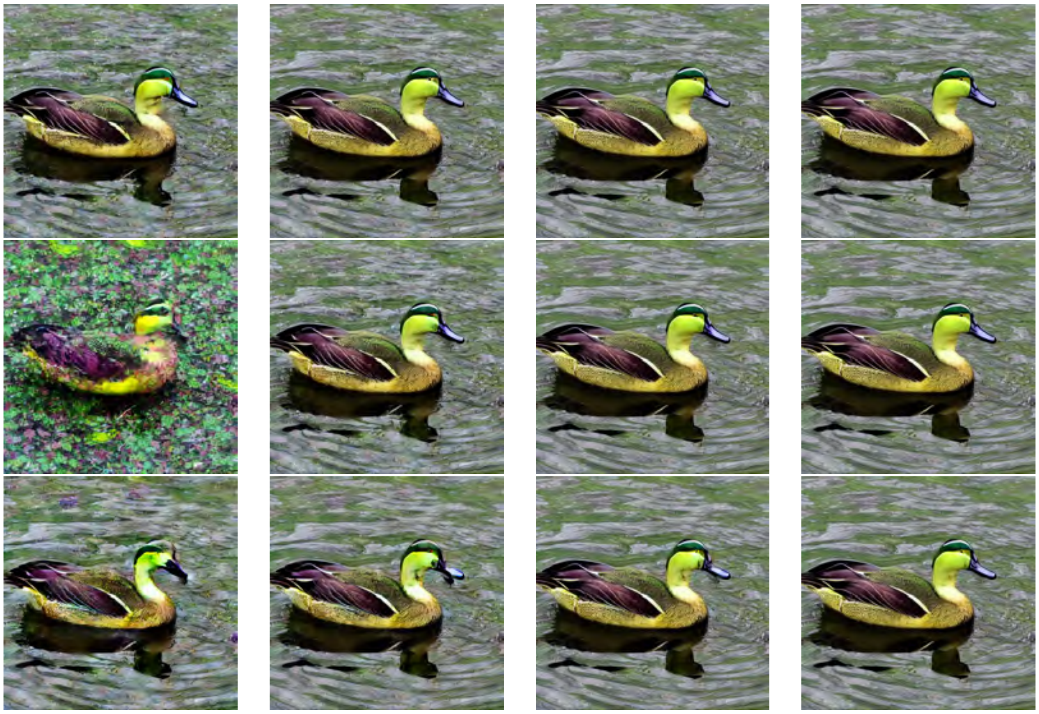

(a) After 7 steps (b) After 11 steps (c) After 15 steps (d) After 19 steps

Figure 9. Iterations of parallel sampling for DDIM 100 steps with DiT model. From top to bottom, the images are generated by ParaTAA, FP and FP+ respectively.



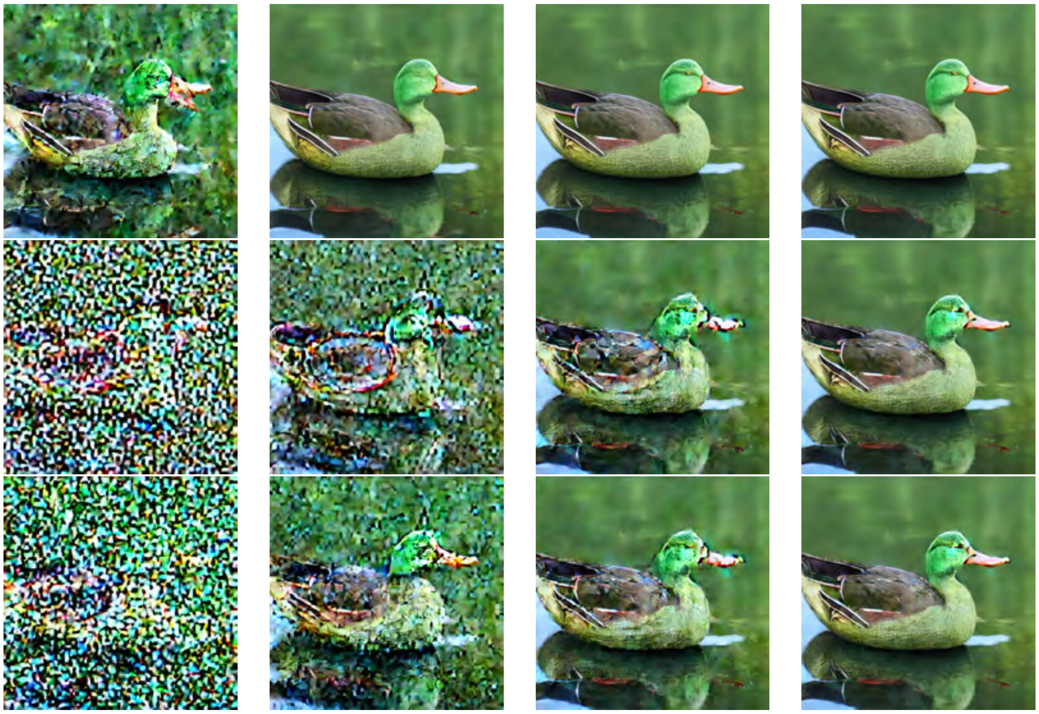
(a) After 17 steps (b) After 21 steps (c) After 27 steps (d) After 31 steps

Figure 10. Iterations of parallel sampling for DDPM 100 steps with DiT model. From top to bottom, the images are generated by ParaTAA, FP and FP+ respectively.



(a) After 5 steps (b) After 7 steps (c) After 9 steps (d) After 11 steps

Figure 11. Iterations of parallel sampling for DDIM 100 steps with SD model. From top to bottom, the images are generated by ParaTAA, FP and FP+ respectively.



(a) After 11 steps (b) After 15 steps (c) After 21 steps (d) After 25 steps

Figure 12. Iterations of parallel sampling for DDPM 100 steps with SD model. From top to bottom, the images are generated by ParaTAA, FP and FP+ respectively.

E. Quantitative Evaluation of Initialization from Existing Trajectory

In this section, we expand the results discussed in Section 5.3. We present a more detailed set of convergence images in Figure 13 as a fine-grained complement to Figure 5. This allows for a clearer observation on convergence when initialized with different methods. Additionally, we conduct a quantitative evaluation of the results shown in Figure 13, which is depicted in Figure 14. Specifically, Figure 14 illustrates the progression of CLIP scores in relation to the second prompt P2. It is evident that initializing with the trajectory leads to significantly faster convergence in terms of the CLIP scores compared to initializing from noise.

F. Additional Examples of Smooth Image Variation

In Figure 15, additional examples are provided to demonstrate the capability of ParaTAA in facilitating smooth image transitions. Specifically, for DDIM with 50 steps, we utilize ParaTAA between two similar prompts, P1 and P2. Initially, we generate a trajectory from P1 using ParaTAA, which is then employed as the starting point for sampling from P2, with the initialization timestep T_{init} set between 35 and 40. The results indicate that ParaTAA can lead transformation from the source to the target image in a seamless manner along the image manifold within very few iteration steps.

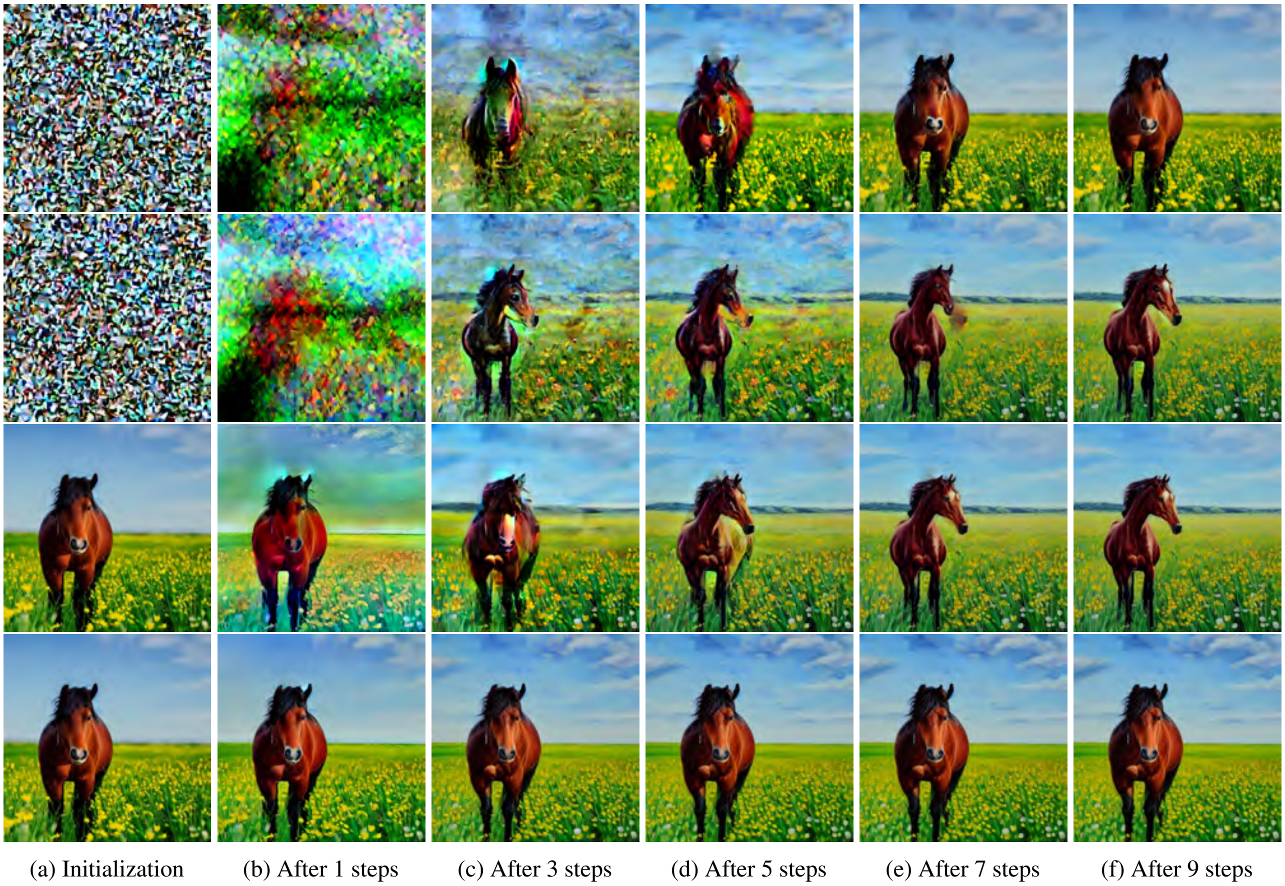


Figure 13. Iterations of ParaTAA with different initaizliations. P1: ”A 4k detailed photo of a horse in a field of flowers”. P2: ”An oil painting of a horse in a field of flowers”. From top to bottom, the rows represents: 1. Sampling with P1 with random initialization; 2. Sampling with P2 with random initialization. 3. Sampling with P2 with trajectory of P1 as initialization and $T_{\text{init}} = 50$. 4. Sampling with P2 with trajectory of P1 as initialization and $T_{\text{init}} = 35$.

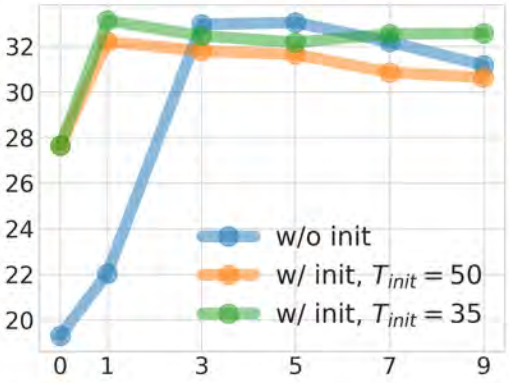
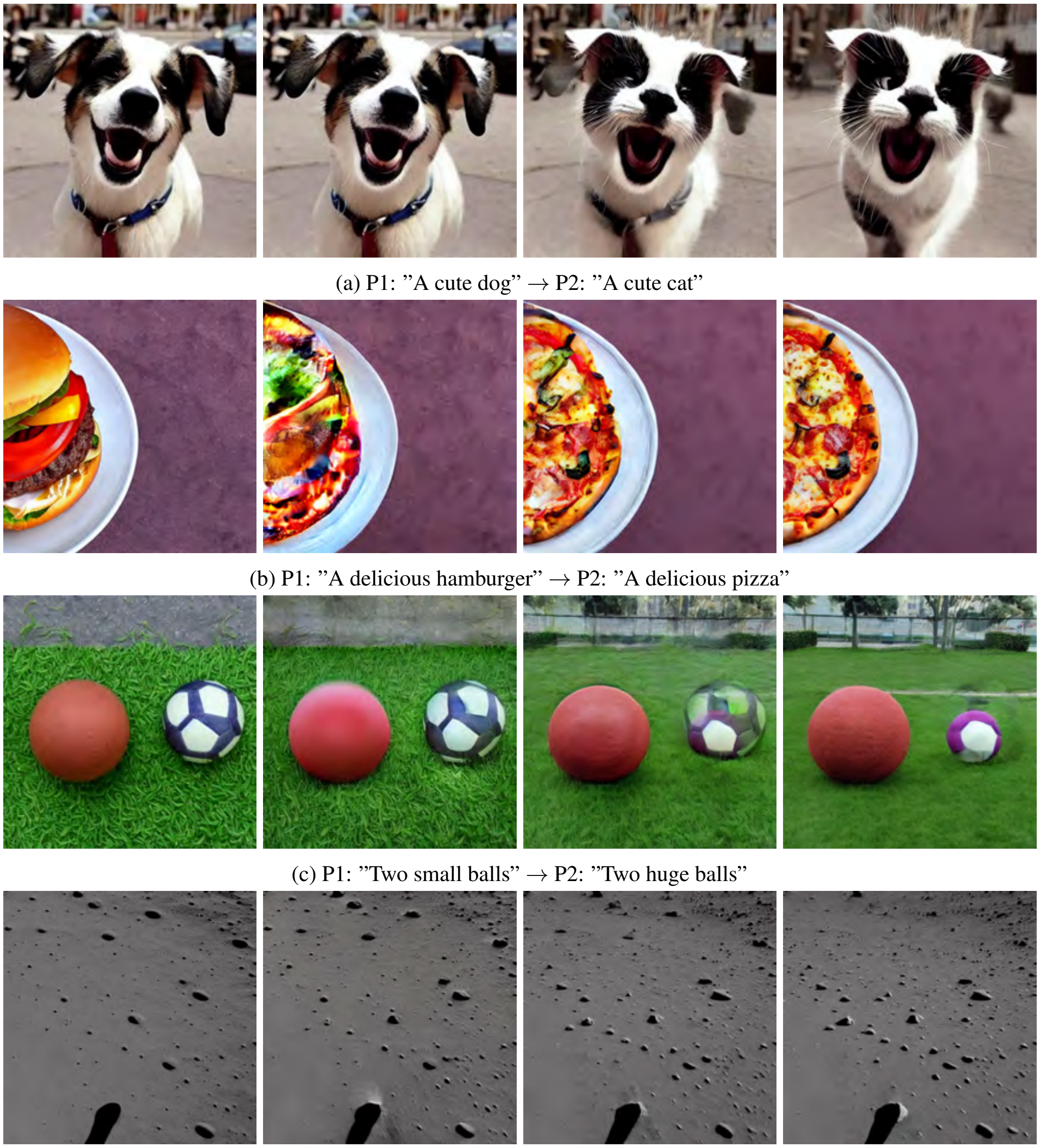


Figure 14. Quantative evaluation for the three settings: 1. Sampling with P2 with random initialization. 2. Sampling with P2 with trajectory of P1 as initialization and $T_{\text{init}} = 50$. 3. Sampling with P2 with trajectory of P1 as initialization and $T_{\text{init}} = 35$. The y-axis is the CLIP scores w.r.t the prompt ”An oil painting of a horse in a field of flowers”.



(a) P1: ”A cute dog” → P2: ”A cute cat”

(b) P1: ”A delicious hamburger” → P2: ”A delicious pizza”

(c) P1: ”Two small balls” → P2: ”Two huge balls”

(d) P1: ”Walking on Moon” → P2: ”Walking on Mars”

Figure 15. Iterations of ParaTAA using an existing trajectory for initialization. From left to right, the columns represent the initial image, the image after 1 step, the image after 3 steps, and the image after 5 steps.