ParaPC-FM: Accelerating Parallel Sampling via Principled Initialization

Bangwei Li[*]
Zhejiang University
libangwei@zju.edu.cn

Chuan Gou[*]
Zhejiang University
chuangou@zju.edu.cn

Jianrong Lu[*,†]
Zhejiang University
jrong.alvin@gmail.com

Guoyao Yu[*]
Zhejiang University
yuguoyao23@zju.edu.cn

Jianhai Chen[†]
Zhejiang University
chenjh919@zju.edu.cn

Abstract

Flow matching and diffusion models achieve high-quality generation by solving continuous-time generative dynamics, but their sampling remains inherently sequential and costly. Parallel sampling reduces wall-clock latency, yet often requires many function evaluations (NFEs) due to inaccurate future-state initialization and repeated correction. We propose **ParaPC-FM**, a fixed-point predictor–corrector framework that unifies high-order solver correction and future-state initialization, enabling both components to be incorporated into different parallel samplers. In particular, our UniP-based truncated initialization reuses past velocity predictions to estimate future states more accurately, reducing correction iterations and NFEs. Initialization requires no additional model evaluations and incurs negligible arithmetic overhead relative to a model forward pass. On a ParaDiGMS-style framework, ParaPC-FM achieves up to $1.25\times$ and $1.49\times$ speedups on FLUX.1-dev at 50 and 100 steps, and up to $1.22\times$ and $1.16\times$ speedups on Stable Diffusion 3 and Stable Diffusion 3.5 at 50 steps. When integrated into ParaTAA, the proposed initialization further achieves up to $1.17\times$ speedup at 50 steps. Under matched-tolerance comparisons, ParaPC-FM improves latency with comparable reported reference-fidelity and preference metrics. Code is available at `https://github.com/libangwei01/ParaPC-FM`.

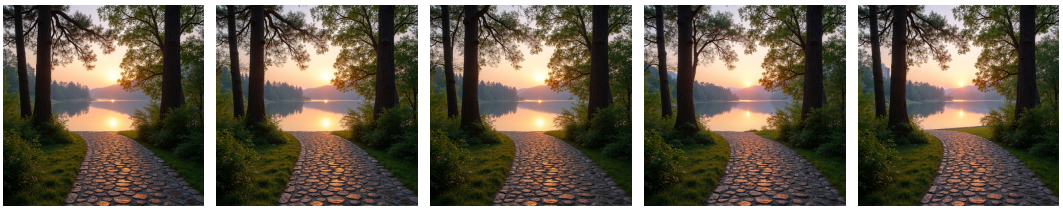
1 Introduction

Flow-based continuous generative models [30, 31, 9, 21, 22, 6, 35, 11] have recently emerged as a competitive alternative to diffusion-based approaches [17] for high-dimensional data generation. Among them, Flow Matching (FM) [21] formulates generation as the solution of a deterministic ordinary differential equation (ODE) [7] by directly learning a velocity field that transports a simple prior distribution to the target data distribution. By avoiding explicit likelihood computation and stochastic sampling [31, 18, 1], flow matching offers improved training stability and conceptual simplicity, and has demonstrated strong empirical performance across a variety of generative tasks.

Despite these advantages, inference in flow matching remains inherently sequential in time, since solving the learned ODE requires each state to depend on previously computed states. This sequential dependency leads to high wall-clock latency for large-scale and high-resolution generation [10, 19, 3, 34, 36]. A natural remedy is parallel-in-time sampling, which predicts multiple future states

[*]Equal contribution.
[†]Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Sequential (50 steps) (b) $p = 0,\ q = 0$ (13 steps) (c) $p = 0,\ q = 1$ (9 steps) (d) $p = 1,\ q = 1$ (10 steps) (e) $p = 2,\ q = 2$ (10 steps)

Figure 1: Comparison of sequential and parallel sampling results on FLUX.1-dev. All images are 512×512 samples generated using the prompt *"A winding stone path leading to a serene lake at sunrise, surrounded by tall trees and lush greenery."* The first image is generated by sequential sampling with 50 steps, while the remaining images are produced by parallel sampling. Here, (p, q) denote the update order and initialization order, respectively. Specifically, $p = 0$ corresponds to the Euler method [13], $p = 1$ to Heun's method [13], and $p = 2$ is equivalent to UniC-2 [41]. The case $q = 0$ indicates that no special initialization is applied.

simultaneously and refines them through Picard-style iterations. However, existing parallel methods often suffer from high NFEs because inaccurate guesses of future states require multiple correction rounds before convergence. Motivated by this limitation, we observe that past velocity predictions along a flow matching trajectory already encode useful temporal information for estimating future states. Therefore, if these previously computed velocities can be reused to produce more accurate initial guesses, parallel correction can converge in fewer iterations, reducing NFEs and improving practical acceleration.

In this paper, we propose **ParaPC-FM**, a parallel predictor–corrector framework for flow matching models. Our main contribution is a UniP-style truncated initialization strategy inspired by UniPC [41], which reuses previously computed velocity predictions to construct higher-order estimates for future states. This provides more accurate initial guesses for Picard-style parallel correction without additional training, prompt-level trajectory reuse, or external reference samples. Since the initialization only depends on available past velocities, it requires no additional model evaluations and incurs negligible arithmetic overhead relative to a model forward pass. This plug-and-play initialization is orthogonal to existing parallel sampling methods and can be incorporated into different parallel-in-time frameworks to reduce correction iterations and lower the total NFEs.

ParaPC-FM also contributes a parallel-computable predictor–corrector solver that fully utilizes the parallel structure of inference. By simultaneously evaluating velocities at multiple states and incorporating a UniC-style correction mechanism, the solver updates parallel states consistently within the same iteration and leverages cross-time velocity information for efficient trajectory refinement.

Experiments on FLUX.1-dev, Stable Diffusion 3, and Stable Diffusion 3.5 show that, under matched-tolerance comparisons, ParaPC-FM reduces wall-clock time with comparable reported reference-fidelity and preference metrics. Compared with no special initialization, ParaPC-FM reaches $1.06\times \sim 1.25\times$ speedup at 50 steps and $1.30\times \sim 1.49\times$ speedup at 100 steps on FLUX.1-dev. Under the 50-step setting, it further reaches $1.07\times \sim 1.22\times$ and $1.03\times \sim 1.16\times$ speedup on Stable Diffusion 3 and Stable Diffusion 3.5, respectively. Moreover, when incorporated into the DDIM sampler of ParaTAA, the proposed initialization achieves $1.14\times \sim 1.17\times$ speedup, further demonstrating its transferability to other parallel samplers.

2 Related Work

2.1 Parallel Sampling for Diffusion and Flow Models

Parallel-in-time sampling aims to reduce the wall-clock latency of generative ODE or SDE sampling by advancing multiple states simultaneously. A representative method is ParaDiGMS [29], which formulates diffusion sampling as a parallelizable fixed-point iteration over time. We consider a

general ODE formulation for diffusion-based generative sampling,

$$x(t) = x(0) + \int_0^t S(x(u), u)\, du, \tag{1}$$

where $S(\cdot, \cdot)$ denotes the drift function. Let $\{x_i\}_{i=0}^T$ denote the discretized trajectory, where x_i corresponds to time t_i. Given an initial trajectory $\{x_i^{\text{old}}\}_{i=0}^T$, the parallel fixed-point iteration updates all states as

$$x_i^{\text{new}} = x_0 + \frac{1}{T} \sum_{u=0}^{i-1} S\big(x_u^{\text{old}}, t_u\big), \quad i = 1, \ldots, T. \tag{2}$$

Eq. (2) allows the evaluations of $\{S(x_u^{\text{old}}, t_u)\}_{u=0}^{T-1}$ to be performed fully in parallel, thereby decoupling the strict temporal dependency of sequential sampling.

Recent parallel sampling methods for diffusion and flow-based generative models can be grouped into fixed-point or Parareal-style trajectory refinement methods [33, 4, 28], parallelizable midpoint or discretization schemes [12, 43], and multi-core hierarchical solver frameworks [25, 14]. These methods reduce sequential dependency by parallelizing trajectory updates or solver communication, but their practical speedup is often limited by the correction rounds and NFEs needed for accurate refinement. In contrast, our method improves future-state initialization by reusing past velocity predictions from the flow matching ODE, providing a plug-and-play way to reduce correction iterations.

2.2 Sequential Sampling Acceleration and UniPC

Existing acceleration methods for diffusion and flow-based generative models fall into two main categories. Training-based methods, such as distillation and consistency-based models [27, 32, 26, 44, 5], reduce sampling steps by learning a faster generator but require additional training. Training-free methods accelerate pretrained models through high-order solvers [23, 42, 24], adaptive step-size control [40, 38, 39], or cross-step reuse [45]. In particular, UniPC [41] builds a unified predictor–corrector framework from previously computed model outputs, inspiring our UniP-style initialization and UniC-style correction in parallel flow matching inference.

Let $\lambda(t)$ denote the half log-SNR parameterization and $h_i = \lambda(t_i) - \lambda(t_{i-1})$ be the step size in the λ domain. Let $\{r_m\}_{m=1}^p$ be a strictly increasing sequence with $r_p = 1$, which determines the auxiliary time points

$$s_m = \lambda^{-1}(r_m h_i + \lambda(t_{i-1})), \quad m = 1, \ldots, p. \tag{3}$$

Denote by $\hat{x}_{t_i}$ the k-th order UniPC update given by

$$\begin{aligned} \hat{x}_{t_i} =& \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \hat{x}_{t_{i-1}} - \sigma_{t_i} \left(e^{h_i} - 1\right) \epsilon_\theta(\hat{x}_{t_{i-1}}, t_{i-1}) \\ & - \sigma_{t_i} B(h_i) \sum_{m=1}^{k} \frac{a_m}{r_m} \left[\epsilon_\theta(\hat{x}_{s_m}, s_m) - \epsilon_\theta(\hat{x}_{t_{i-1}}, t_{i-1})\right], \end{aligned} \tag{4}$$

where $B(h) = \mathcal{O}(h)$ is a non-zero function of h.

Under the temporal smoothness assumption of the model output, the coefficients $\{a_m\}_{m=1}^p$ are chosen to cancel low-order error terms between the numerical approximation and the exact solution expansion. As shown in Zhao et al. [41], the $(p-1)$-th order update ($k = p-1$) corresponds to the UniP-p predictor, achieving p-th order accuracy, while the p-th order update ($k = p$) corresponds to the UniC-p corrector, achieving $(p+1)$-th order accuracy. Our method is inspired by this predictor–corrector perspective but targets a different bottleneck: instead of only improving sequential solver accuracy, we use a UniP-style rule to initialize future states in parallel sampling and incorporate a UniC-style correction into a parallel-computable solver for flow matching models.

3 Parallel Predictor–Corrector Framework For Flow Matching Models

In this section, we introduce ParaPC-FM, a framework that provides both an iterative solver update scheme and an initialization strategy for parallel sampling in flow matching models.

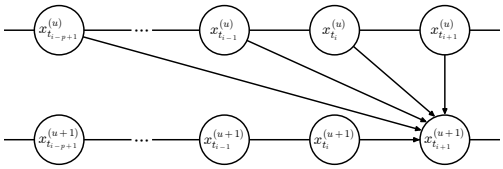


Figure 2: Schematic illustration of the ParaPC-FM update at time step t_{i+1}. The top row shows the states $\{x^{(u)}_{t_{i-p+1}}, \ldots, x^{(u)}_{t_{i+1}}\}$ from the u-th iteration, while the bottom row shows the corresponding states $\{x^{(u+1)}_{t_{i-p+1}}, \ldots, x^{(u+1)}_{t_{i+1}}\}$ after the update. All states in the u-th iteration contribute, via velocity differences, together with the updated state $x^{(u+1)}_{t_i}$, to the computation of $x^{(u+1)}_{t_{i+1}}$.

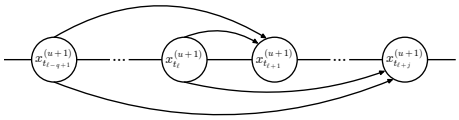


Figure 3: Schematic illustration of truncated ParaPC-FM initialization. The available states $\{x^{(u+1)}_{t_{\ell-q+1}}, \ldots, x^{(u+1)}_{t_\ell}\}$ jointly provide truncated velocity corrections for initializing the future states $\{x^{(u+1)}_{t_{\ell+1}}, \ldots, x^{(u+1)}_{t_{\ell+j}}\}$. Each initialized state aggregates information only from past states within the truncated window.

3.1 Iterative Update Rule

We consider generative sampling formulated as solving an ordinary differential equation (ODE)

$$\frac{dx(t)}{dt} = v_\theta(x(t), t), \tag{5}$$

defined on the time interval $t \in [0, 1]$ with boundary conditions $t_0 = 0$ and $t_N = 1$, and discretized on a time grid $\{t_i\}_{i=0}^N$ with corresponding states $\{x_{t_i}\}_{i=0}^N$.

We define the iterative ParaPC-FM update at time step t_i as follows. Let $x^{(u)}_{t_i}$ denote the estimate of the state at time t_i after the u-th parallel correction iteration. Then the $(u+1)$-th iterate at t_{i+1} is given by

$$x^{(u+1)}_{t_{i+1}} = x^{(u+1)}_{t_i} + \Delta t_i \, v_\theta\Big(x^{(u)}_{t_i}, t_i\Big) + \sum_{m=1}^{p} a_m \, D^{(u)}_m, \tag{6}$$

where $\Delta t_i = t_{i+1} - t_i$, and the velocity difference term is defined as

$$D^{(u)}_m \;\coloneqq\; \begin{cases} v_\theta\Big(x^{(u)}_{t_{i-m}}, t_{i-m}\Big) - v_\theta\Big(x^{(u)}_{t_i}, t_i\Big), & m < p, \\ v_\theta\Big(x^{(u)}_{t_{i+1}}, t_{i+1}\Big) - v_\theta\Big(x^{(u)}_{t_i}, t_i\Big), & m = p. \end{cases} \tag{7}$$

This formulation incorporates cross-time velocity differences to refine the local Euler update, enabling higher-order correction while preserving a simple additive structure. The parallel dependency pattern of this iterative update, in which multiple states from the u-th iteration jointly contribute to the computation of $x^{(u+1)}_{t_{i+1}}$, is illustrated schematically in Figures 2.

The coefficients $\{a_m\}_{m=1}^p$ are constructed to achieve higher-order accuracy by matching local integral moments of the velocity field. We define the normalized backward time offsets as

$$r_m \coloneqq \frac{t_{i-m} - t_i}{\Delta t_i}, \qquad m = 1, \ldots, p-1, \tag{8}$$

and additionally set $r_p = 1$ for convenience.

We then form a Vandermonde-like matrix $R \in \mathbb{R}^{p \times p}$ and define the target moment vector $b \in \mathbb{R}^p$ as

$$R_{k,m} = r_m^k, \qquad b_k = \frac{\Delta t_i}{k+1}, \qquad k = 1, \ldots, p. \tag{9}$$

The coefficient vector $a = (a_1, \ldots, a_p)^\top$ is obtained by solving the linear system

$$Ra = b, \tag{10}$$

which yields the correction weights used in Eq. (6). For completeness, the full computational procedure of the ParaPC-FM iterative update, including the construction of velocity differences and the predictor–corrector update, is summarized in Algorithm 2 in Appendix A. Under mild conditions, Eq. (6) can achieve $(p+1)$-th order accuracy during parallel iterations; see Proposition B.12 and Section C.4 for the full statement and proof.

3.2 Initialization with Truncated Correction

The ParaPC-FM update in Eq. (6) requires multiple past velocity predictions. During parallel inference, however, velocity predictions are only available up to a certain time index. We denote by t_ℓ the last time step for which a velocity prediction $v_\theta(x_{t_\ell}^{(u)}, t_\ell)$ is available. Our goal is to initialize the subsequent states $\{x_{t_{\ell+1}}^{(u+1)}, \dots, x_{t_{\ell+j}}^{(u+1)}\}$ before full parallel correction becomes feasible.

We introduce an *initialization order* q. A q-th order initialization employs $(q-1)$ past velocity differences relative to t_ℓ. In particular, when $q=0$, $\{x_{t_{\ell+1}}^{(u+1)}, \dots, x_{t_{\ell+j}}^{(u+1)}\}$ are initialized to the value of $x_{t_\ell}^{(u+1)}$. When $q=1$, the initialization reduces to a pure first-order Euler step. Higher values of q incorporate additional truncated correction terms based on available past velocity predictions, providing a more accurate initial guess and improving convergence speed in subsequent parallel correction.

Specifically, for each initialization step from t_ℓ to $t_{\ell+k}$, where $k=1,\dots,j$, we apply a truncated ParaPC-FM update of order q,

$$x_{t_{\ell+k}}^{(u+1)} = x_{t_\ell}^{(u+1)} + \Delta t_{\ell,k}\, v_\theta(x_{t_\ell}^{(u)}, t_\ell) + \sum_{m=1}^{q-1} a_m\, D_m^{(u)}, \tag{11}$$

where $\Delta t_{\ell,k} = t_{\ell+k} - t_\ell$ and

$$D_m^{(u)} \coloneqq v_\theta(x_{t_{\ell-m}}^{(u)}, t_{\ell-m}) - v_\theta(x_{t_\ell}^{(u)}, t_\ell). \tag{12}$$

Compared to the full ParaPC-FM update in Eq. (6), the truncated initialization omits the highest-order correction term and relies only on the available past velocity predictions $\{v_{t_{\ell-(q-1)}}^{(u)}, \dots, v_{t_{\ell-1}}^{(u)}\}$. The coefficients $\{a_m\}_{m=1}^{q-1}$ are computed using the same moment-matching principle as in Section 3.1, restricted to the truncated basis. The resulting dependency structure of the initialization step is illustrated in Figures 3, where each newly initialized state aggregates information from all available past states within the truncated window.

For clarity and reproducibility, the complete computational procedure of the truncated ParaPC-FM initialization is summarized in Algorithm 3 in Appendix A. As additional velocity predictions become available, the method naturally transitions from truncated initialization of order q to the complete ParaPC-FM update of order p, ensuring a smooth and stable inference process. When $p \geq 0$ and $q \geq 1$, the initialization in Eq. (11) guarantees faster convergence than the case without special initialization. A complete discussion is provided in Remark B.17.

3.3 Sliding-Window Parallel Inference

We apply ParaPC-FM to parallel flow matching inference using an adaptive sliding-window scheme. At each stage, we consider a window of length n covering time steps $\{t_i, t_{i+1}, \dots, t_{i+n-1}\}$, where t_i denotes the most recent time step whose state is regarded as converged.

Parallel correction within a window. Given the current reference state $x_{t_i}^{(u)}$, all states inside the window are available, either from the initial setup or from previous initialization. Velocity predictions $\{v_\theta(x_{t_{i+k}}^{(u)}, t_{i+k})\}_{k=0}^{n-1}$ are evaluated in parallel. Using these predictions, we apply the iterative ParaPC-FM update from Section 3.1 sequentially along the window to obtain corrected states $\{x_{t_i}^{(u+1)}, \dots, x_{t_{i+n-1}}^{(u+1)}\}$.

Convergence detection. To determine how far the window can advance, we measure the discrepancy between the corrected and previous states. For $k=1,\dots,n-1$, we define the squared

Algorithm 1 Parallel inference for ParaPC-FM

Require: Initial state x_{t_0}; time grid $\{t_i\}_{i=0}^{N}$; window length n; update order p; initialization order q; tolerance tol

Initialize reference index $i \leftarrow 0$
Set $x_{t_{i+k}} \leftarrow x_{t_i}$ for $k = 1, \ldots, n-1$
while $i < N$ **do**
  Evaluate $\{v_{t_{i+k}}\}_{k=0}^{n-1}$ in parallel
  Compute $\hat{x}_{t_{i+k}}$ by (6) for $k = 0, \ldots, n-1$
  $s \leftarrow n$
  for $k = 1$ **to** $n-1$ **do**
    if $\left\|\hat{x}_{t_{i+k}} - x_{t_{i+k}}\right\|_2^2 > \mathrm{tol}^2 \Delta t_{i+k-1}^2 d$ **then**
      $s \leftarrow k$
      break
    end if
  end for
  Set $x_{t_{i+k}} \leftarrow \hat{x}_{t_{i+k}}$ for $k = 1, \ldots, n-1$
  Reinitialize $x_{t_{i+k}}$ by (11) with $q = p$ for $k = s, \ldots, n-1$
  Initialize $\{x_{t_{i+n}}, \ldots, x_{t_{i+n+s-1}}\}$ by (11)
  $i \leftarrow i + s$
end while
return $\{x_{t_i}\}_{i=0}^{N}$

correction error

$$e_{i+k} \coloneqq \left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{(u)} \right\|_2^2. \tag{13}$$

We identify the smallest index s such that

$$e_{i+s} > \mathrm{tol}^2 \cdot \Delta t_{i+s-1}^2 \cdot d, \tag{14}$$

where $\Delta t_{i+s-1} = t_{i+s} - t_{i+s-1}$, d is the dimensionality of the latent state, and tol is a user-specified tolerance parameter that controls the admissible correction magnitude in the parallel iteration.

All states up to t_{i+s-1} are then regarded as converged, and the window is advanced by s steps. For the remaining unconverged states within the current window, i.e., $\{t_{i+s}, \ldots, t_{i+n-1}\}$, we reuse the corrected values $\{x_{t_{i+s}}^{(u+1)}, \ldots, x_{t_{i+n-1}}^{(u+1)}\}$ as their initial states in the next parallel iteration. Then the state $x_{t_{i+s}}^{(u+1)}$ serves as the starting point of the next sliding window. Its velocity is re-evaluated in the subsequent parallel iteration to ensure consistency with the updated window context.

Initialization beyond the window. After shifting the window by s steps, the new sliding window becomes $\{t_{i+s}, t_{i+s+1}, \ldots, t_{i+s+n-1}\}$. For the states that remain within the previously predicted range, we first reinitialize $\{x_{t_{i+s}}^{(u+1)}, \ldots, x_{t_{i+n-1}}^{(u+1)}\}$ using the velocity predictions available up to the immediately preceding time step, where the initialization order is fixed as $q = p$. Among the states in the new window, $x_{t_{i+n-1}}^{(u+1)}$ is the last state for which a velocity prediction is available from the previous iteration. Consequently, we treat t_{i+n-1} as the last available prediction index and initialize the subsequent states $\{x_{t_{i+n}}^{(u+1)}, \ldots, x_{t_{i+n+s-1}}^{(u+1)}\}$ using the truncated ParaPC-FM initialization in Section 3.2. Together, these reinitialized and newly initialized states complete the new sliding window and provide stable starting points for the next round of parallel correction.

By decoupling correction and initialization in this manner, ParaPC-FM ensures that all parallel updates are performed within a window of validated states, while initialization is only used to extend the horizon once convergence has been detected.

The overall sliding-window parallel inference procedure that integrates the ParaPC-FM update, truncated initialization, and adaptive window shifting is summarized in Algorithm 1.

Finally, we note that the proposed sliding-window parallel inference scheme does not increase the number of iterations compared to sequential sampling. A formal proof is provided in Appendix C (Proof of Proposition B.18). We further provide an end-to-end error analysis in Appendix D, where

Table 1: Performance comparison on FLUX.1-dev under different update orders p and initialization orders q. Here, $p = 0$ corresponds to the Euler update, $p = 1$ corresponds to Heun's method, and $p = 2$ corresponds to UniPC-2. The setting $(p, q) = (0, 0)$ is the original ParaDiGMS baseline. $S = T_{p,0}/T_{p,q}$ denotes initialization speedup at fixed p, sampling steps, and tolerance; time is in seconds. RMSE is computed against the corresponding 100-step sequential reference of the same update order, i.e., Euler, Heun, or UniPC-2, after normalizing images to $[0, 1]$. The detailed tolerance settings are provided in Table 5.

Update order	Init. order	FLUX.1-dev (25 steps)					FLUX.1-dev (50 steps)					FLUX.1-dev (100 steps)				
		Steps↓	Time↓	$S\uparrow$	CLIP↑	RMSE↓	Steps↓	Time↓	$S\uparrow$	CLIP↑	RMSE↓	Steps↓	Time↓	$S\uparrow$	CLIP↑	RMSE↓
	Seq.	25	11.739	–	26.1	0.119	50	22.788	–	26.0	0.073	100	45.821	–	26.0	–
$p = 0$ Euler	$q = 0$	7.926	3.271	1.00	26.3	0.142	15.243	6.385	1.00	26.1	0.097	33.295	14.409	1.00	26.0	0.034
	$q = 1$	**7.430**	**3.088**	**1.06**	26.3	0.142	**13.282**	**5.581**	**1.14**	26.1	0.098	**27.566**	**11.948**	**1.21**	26.0	0.035
	$q = 2$	8.170	3.373	0.97	26.3	0.142	15.051	6.296	1.01	26.1	0.098	29.778	12.889	1.12	26.0	0.035
$p = 1$ Heun	$q = 0$	8.387	3.393	1.00	26.2	0.154	16.268	6.687	1.00	26.0	0.110	34.307	14.780	1.00	26.0	0.072
	$q = 1$	**8.057**	**3.266**	**1.04**	26.2	0.154	**14.532**	**5.984**	**1.12**	26.0	0.111	**29.317**	**12.641**	**1.17**	26.0	0.072
	$q = 2$	8.839	3.523	0.96	26.2	0.154	16.003	6.594	1.01	26.0	0.111	30.844	13.264	1.11	26.0	0.072
$p = 2$ UniPC-2	$q = 0$	8.697	3.531	1.00	26.2	0.146	16.625	6.878	1.00	26.0	0.087	30.730	13.284	1.00	26.0	0.059
	$q = 1$	**8.308**	**3.393**	**1.04**	26.2	0.146	**14.876**	**6.167**	**1.12**	25.9	0.088	**27.224**	**11.791**	**1.13**	26.0	0.059
	$q = 2$	8.930	3.612	0.98	26.3	0.147	16.164	6.730	1.02	26.0	0.088	28.302	12.226	1.09	25.9	0.059

the total error is decomposed into the sequential discretization error and a tolerance-controlled parallel early-stopping error.

4 Experiments

4.1 Experimental Setup

ParaDiGMS-based evaluation. Our main experiments are conducted on top of a ParaDiGMS-style [29] parallel sampling framework. We evaluate ParaPC-FM on FLUX.1-dev [19], Stable Diffusion 3, and Stable Diffusion 3.5 [10] at 512×512 resolution. For FLUX.1-dev, we compare different sampling steps, update orders p, initialization orders q, and tolerance values tol. For Stable Diffusion 3 and Stable Diffusion 3.5, we focus on the 50-step setting and compare different (p, q) combinations. We report effective correction iterations, wall-clock time, normalized RMSE, ImageReward [37], and CLIP score [2, 8, 15]. For the reported sequential baselines, $\mathrm{NFE} = \mathrm{Steps}$, while for eight-device parallel sampling with $n = 8$ we estimate $\widehat{\mathrm{NFE}} = 8 \times \mathrm{Steps}$. The actual NFE is lower because the final correction rounds may use fewer than all eight devices. RMSE is computed after normalizing images to $[0, 1]$ and comparing them with sequential references generated by the corresponding update-order solver. These experiments are evaluated on $1{,}000$ prompts, with additional implementation details provided in Appendix E.1. Comparisons with simple extrapolation baselines and a window-size study are provided in Appendix G.

Transfer to ParaTAA. To examine whether the proposed initialization is specific to ParaDiGMS, we further integrate ParaPC-FM initialization into ParaTAA-style [33] parallel sampling. We use Stable Diffusion 1.5 and compare $q \in \{0, 1, 2\}$ while keeping the ParaTAA sampling procedure unchanged. This experiment is evaluated on $10{,}000$ prompts using average inference time, average correction steps, ImageReward, and CLIP score. Detailed settings are provided in Appendix E.2.

4.2 Definition and Experimental Results

Definition 4.1 (Initialization speedup)**.** For a fixed update order p, we take the setting without special initialization ($q = 0$) as the baseline. Let T_0 denote the convergence time of this baseline, and let T_q denote the convergence time when using initialization order q. The initialization speedup for order q is defined as

$$\mathrm{Speedup}(q) := \frac{T_0}{T_q}, \tag{15}$$

where $\mathrm{Speedup}(0) = 1$ by definition.

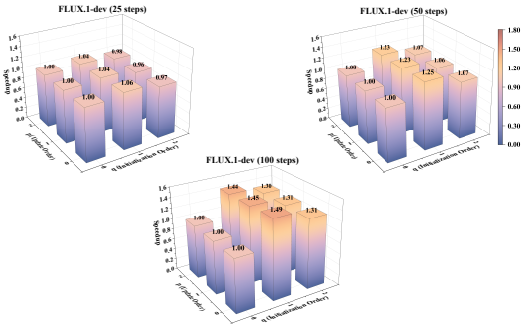


Figure 4: Initialization speedup on FLUX.1-dev under different sampling steps. Each panel reports the speedup of different initialization orders q relative to the corresponding $q = 0$ baseline under the same update order p, with $\mathrm{tol} = 1.0$.

Table 2: Effect of the convergence tolerance tol on FLUX.1-dev with 50 sampling steps. All ParaPC-FM experiments use $p = 0$ and $q = 1$. RMSE is computed against the 50-step sequential Euler reference (Seq.) after normalizing images to $[0, 1]$. S is the speedup relative to the corresponding $q = 0$ setting.

Tol.	Steps↓	Time↓	S ↑	ImRe↑	RMSE↓
Seq.	50	22.788	–	0.842	–
0.5	13.282	5.581	1.144	0.845	**0.084**
0.6	12.528	5.256	1.157	0.849	0.091
0.7	11.940	5.028	1.171	0.854	0.097
0.8	11.449	4.803	1.199	0.854	0.101
0.9	10.977	4.624	1.223	0.855	0.106
1.0	**10.562**	**4.444**	**1.255**	**0.855**	0.109

Experimental results. Table 1 reports the RMSE-controlled FLUX.1-dev results under different update orders p, initialization orders q, and sampling steps, where the tolerance is selected to keep the RMSE variation across different q values within 0.001 whenever possible. Table 3 reports the corresponding 50-step results on Stable Diffusion 3 and Stable Diffusion 3.5. Table 5 summarizes the detailed tolerance settings used in both Table 1 and Table 3. To isolate the acceleration effect of initialization, Figures 4 shows the initialization speedup on FLUX.1-dev under $\mathrm{tol} = 1.0$ for 25, 50, and 100 steps, while Figures 5 reports the corresponding 50-step results on Stable Diffusion 3 and Stable Diffusion 3.5. Finally, Table 4 shows that the proposed initialization also accelerates ParaTAA DDIM, supporting its transferability as a plug-and-play module for different parallel samplers. The computational resource usage and estimated reproduction time are summarized in Appendix E.3. More qualitative comparisons are provided in Appendix H.

4.3 Analysis

Effect of update order. Figures 4 and 5 further analyze the effect of the update order p on initialization-induced acceleration. Under the same tolerance setting, changing p does not lead to a clear or consistent change in the speedup obtained from different initialization orders q. This suggests that the acceleration benefit mainly comes from the improved future-state initialization rather than the specific update order. In contrast, p primarily affects the numerical trajectory itself, making the parallel results closer to the corresponding sequential solver of the same order, as reflected by the RMSE against the 100-step sequential references.

Effect of initialization order. Under the fixed-tolerance setting, Figures 4 and 5 show that using a nonzero initialization order generally improves the speedup over the baseline $q = 0$. Among the tested settings, $q = 1$ provides the most consistent and largest acceleration, suggesting that first-order truncated initialization offers a good balance between initialization accuracy and overhead. In the RMSE-controlled results, Tables 1 and 3 show that $q = 1$ and $q = 2$ still reduce the effective correction steps and wall-clock time in many cases when RMSE remains nearly unchanged across different q settings. This indicates that the observed speedup comes from improved future-state initialization rather than degraded image fidelity. By contrast, $q = 2$ is more sensitive to the sampling step size. When the number of steps is small, such as the 25-step setting, the step size is relatively large and the higher-order temporal approximation used by $q = 2$ becomes less reliable, which can lead to negative acceleration. As the number of steps increases and the trajectory is more finely discretized, the higher-order information becomes more accurate and the benefit of $q = 2$ becomes more evident.

Effect of sampling steps. Figures 4 also shows that, for the same tolerance and the same (p, q) combination, the acceleration becomes stronger as the number of sampling steps increases. This trend is consistent with the behavior of truncated initialization: when the step size becomes smaller, the local truncation error decreases, so the initialization based on past velocity predictions is closer

Table 3: Performance comparison on Stable Diffusion 3 and Stable Diffusion 3.5 under different update orders p and initialization orders q using ParaPC-FM with 50 sampling steps. The setting $(p, q) = (0, 0)$ is the original ParaDiGMS baseline. $S = T_{p,0}/T_{p,q}$ is initialization speedup within each model at fixed p and tolerance; time is in seconds. RMSE is computed against the corresponding 50-step sequential reference after normalizing images to $[0, 1]$. The detailed tolerance settings are provided in Table 5.

Update order	Init. order	Stable Diffusion 3 (50 steps)					Stable Diffusion 3.5 (50 steps)				
		Steps↓	Time↓	$S\uparrow$	CLIP↑	RMSE↓	Steps↓	Time↓	$S\uparrow$	CLIP↑	RMSE↓
	Seq.	50	2.232	–	26.2	–	50	21.614	–	26.6	–
$p = 0$	$q = 0$	15.720	0.855	1.00	26.3	0.092	14.287	5.800	1.00	26.6	0.114
Euler	$q = 1$	**13.338**	**0.736**	**1.16**	26.3	0.093	**12.272**	**4.983**	**1.16**	26.6	0.115
	$q = 2$	14.743	0.807	1.06	26.3	0.093	14.016	5.648	1.03	26.6	0.115
$p = 1$	$q = 0$	17.547	0.955	1.00	26.2	0.112	15.596	6.382	1.00	26.6	0.120
Heun	$q = 1$	**15.312**	**0.844**	**1.13**	26.2	0.112	**13.850**	**5.597**	**1.14**	26.6	0.119
	$q = 2$	16.467	0.900	1.06	26.2	0.113	15.303	6.191	1.03	26.6	0.120
$p = 2$	$q = 0$	18.635	1.019	1.00	26.3	0.098	15.989	6.545	1.00	26.5	0.122
UniPC-2	$q = 1$	**16.334**	**0.896**	**1.14**	26.3	0.098	**14.175**	**5.746**	**1.14**	26.5	0.122
	$q = 2$	17.388	0.943	1.08	26.3	0.099	15.678	6.309	1.04	26.6	0.122

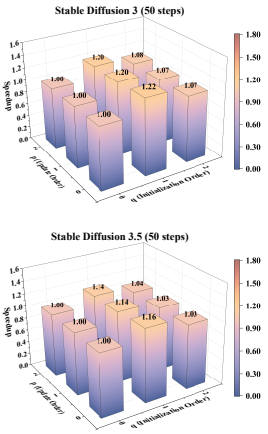


Figure 5: Initialization speedup of ParaPC-FM on Stable Diffusion 3 and Stable Diffusion 3.5 under $\mathrm{tol} = 1.0$ with 50 sampling steps. For each update order p, the baseline without special initialization ($q = 0$) is normalized to a speedup of 1, and the bars report the relative acceleration achieved by higher-order initialization ($q = 1, 2$).

Table 4: Combining ParaPC-FM initialization with ParaTAA DDIM on Stable Diffusion 1.5. The $q = 0$ baseline is the original ParaTAA DDIM implementation; $S = T_{q=0}/T_q$ denotes initialization speedup.

Method	Time (s)↓	Steps↓	$S\uparrow$	ImRe↑	CLIP↑
Original ParaTAA DDIM ($q = 0$)	1.738	20.794	1.000	-0.303	24.2
ParaTAA DDIM + ParaPC-FM ($q = 1$)	**1.481**	**18.144**	**1.174**	-0.304	24.2
ParaTAA DDIM + ParaPC-FM ($q = 2$)	1.525	18.525	1.140	-0.304	24.2

to the converged parallel trajectory than the $q = 0$ initialization. Consequently, the correction process requires fewer iterations to reach the same tolerance, and the relative speedup becomes more pronounced under finer discretizations.

Effect of tolerance. Table 2 shows the trade-off controlled by the convergence tolerance tol under $p = 0$ and $q = 1$. A smaller tolerance enforces stricter correction, so the RMSE decreases toward zero but the speedup also approaches $1\times$. This observation is consistent with the end-to-end error analysis in Appendix D. In contrast, a larger tolerance accepts more states per round, reducing correction steps and improving wall-clock speedup, while increasing RMSE relative to the 50-step sequential Euler reference. Across the tested range, however, ImageReward varies only slightly, providing complementary preference-oriented evidence. Appendix F explicitly visualizes the speedup–reference-fidelity trade-off.

Transferability to other parallel samplers. Table 4 evaluates whether the proposed initialization strategy can be transferred to another parallel sampling method. We integrate ParaPC-FM initialization into ParaTAA DDIM while keeping the original ParaTAA sampling procedure unchanged. Compared with the baseline $q = 0$ setting, ParaPC-FM initialization achieves $1.174\times$ speedup when $q = 1$ and $1.140\times$ speedup when $q = 2$, while maintaining comparable ImageReward and CLIP scores. These results show that the proposed initialization is not tied to a specific ParaDiGMS-style correction scheme, but can also accelerate other parallel samplers as a plug-and-play module.

4.4 Limitations

While ParaPC-FM demonstrates consistent acceleration across different flow-matching models and can be incorporated into different parallel sampling methods, several limitations remain. In low-step regimes with large step sizes, truncation errors can become pronounced, reducing the effectiveness of initialization and weakening the acceleration. This suggests that the initialization strategy can be further improved, especially under coarse discretization. Future work may refine the prediction

process, for example by adaptively weighting prediction terms according to the step size to mitigate the adverse effects of coarse discretization.

5 Conclusion

In this work, we propose **ParaPC-FM**, a time-parallel inference framework for flow matching models based on fixed-point iteration and predictor–corrector schemes. ParaPC-FM unifies high-order solver correction and future-state initialization, while decoupling them through the update order p and initialization order q. This design enables the solver and initialization components to be flexibly incorporated into existing parallel sampling methods without modifying model architectures or training procedures. Our results show that initialization quality is crucial for the convergence speed of parallel sampling. The proposed truncated initialization reuses past velocity predictions to produce more accurate future-state guesses without additional model evaluations and with negligible arithmetic overhead relative to a model forward pass, thereby reducing the required number of function evaluations and improving inference efficiency.

Experiments on FLUX.1-dev, Stable Diffusion 3, and Stable Diffusion 3.5 demonstrate consistent acceleration with comparable reported reference-fidelity and preference metrics under matched tolerances. Moreover, the proposed initialization also accelerates ParaTAA when integrated into its sampling procedure, suggesting that it can serve as a general plug-and-play acceleration module for existing parallel generative sampling methods. We hope that this work offers useful insights into the role of initialization in parallel sampling and inspires future research on efficient parallel inference for generative models.

Acknowledgments and Disclosure of Funding

This work was supported by the Key R&D Program of Zhejiang Province under Grant No. 2025C01084 and the CAAI–MindSpore Open Fund, developed on the OpenI Community, under Contract No. CAAIXSJLJJ 2025 MindSpore 04.

References

[1] Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209): 1–80, 2025. URL `http://jmlr.org/papers/v26/23-1605.html`.

[2] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Abdul Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Shang-Wen Li, Piotr Dollar, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=INqBOmwIpG`.

[3] Siyu Cao, Hangting Chen, Peng Chen, Yiji Cheng, Yutao Cui, Xinchi Deng, Ying Dong, Kipper Gong, Tianpeng Gu, Xiusen Gu, et al. Hunyuanimage 3.0 technical report. *arXiv preprint arXiv:2509.23951*, 2025.

[4] Haoxuan Chen, Yinuo Ren, Lexing Ying, and Grant M Rotskoff. Accelerating diffusion models with parallel sampling: Inference at sub-linear time complexity. *Advances in Neural Information Processing Systems*, 37:133661–133709, 2024.

[5] Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16185–16195, October 2025.

[6] Ricky TQ Chen and Yaron Lipman. Flow matching on general geometries. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[8] Jang Hyun Cho, Andrea Madotto, Effrosyni Mavroudi, Triantafyllos Afouras, Tushar Nagarajan, Muhammad Maaz, Yale Song, Tengyu Ma, Shuming Hu, Suyog Jain, Miguel Martin, Huiyu Wang, Hanoona Abdul Rasheed, Peize Sun, Po-Yao Huang, Daniel Bolya, Nikhila Ravi, Shashank Jain, Tammy Stark, Seungwhan Moon, Babak Damavandi, Vivian Lee, Andrew Westbury, Salman Khan, Philipp Kraehenbuehl, Piotr Dollar, Lorenzo Torresani, Kristen Grauman, and Christoph Feichtenhofer. PerceptionLM: Open-access data and models for detailed visual understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=5NkfjxMpWe.

[9] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=CzceR82CYc.

[10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[11] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uWj4s7rMnR.

[12] Shivam Gupta, Linda Cai, and Sitan Chen. Faster diffusion sampling with randomized midpoints: Sequential and parallel. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MT3aOfXIbY.

[13] Ernst Hairer, Gerhard Wanner, and Syvert P Nørsett. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

[14] Jiaqi Han, Haotian Ye, Puheng Li, Minkai Xu, James Zou, and Stefano Ermon. Chords: Diffusion sampling accelerator with multi-core hierarchical ode solvers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19386–19395, 2025.

[15] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 7514–7528, 2021.

[16] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

[19] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

[20] John Denholm Lambert et al. *Numerical methods for ordinary differential systems*, volume 146. Wiley New York, 1991.

[21] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[22] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[23] 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. *Advances in neural information processing systems*, 35:5775–5787, 2022.

[24] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, pages 1–22, 2025.

[25] Jianrong Lu, Zhiyu Zhu, and Junhui Hou. Parasolver: A hierarchical parallel integral solver for diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[26] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[27] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

[28] Nikil R Selvam, Amil Merchant, and Stefano Ermon. Self-refining diffusion samplers: Enabling parallelization via parareal iterations. *Advances in Neural Information Processing Systems*, 37: 5429–5453, 2024.

[29] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:4263–4276, 2023.

[30] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[31] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[32] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html.

[33] Zhiwei Tang, Jiasheng Tang, Hao Luo, Fan Wang, and Tsung-Hui Chang. Accelerating parallel sampling of diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

[34] Z-Image Team. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

[35] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

[36] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

[37] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15903–15935, 2023.

[38] Shuchen Xue, Zhaoqiang Liu, Fei Chen, Shifeng Zhang, Tianyang Hu, Enze Xie, and Zhenguo Li. Accelerating diffusion sampling with optimized time steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8292–8301, 2024.

[39] Hui Zhang, Zuxuan Wu, Zhen Xing, Jie Shao, and Yu-Gang Jiang. Adadiff: Adaptive step selection for fast diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 9914–9922, 2025.

[40] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023.

[41] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:49842–49869, 2023.

[42] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36: 55502–55542, 2023.

[43] Huanjian Zhou and Masashi Sugiyama. Parallel simulation for log-concave sampling and score-based diffusion models. In *International Conference on Machine Learning*, pages 79192–79225. PMLR, 2025.

[44] Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. *Advances in Neural Information Processing Systems*, 37:40831–40860, 2024.

[45] Shangwen Zhu, Han Zhang, Zhantao Yang, Qianyu Peng, Zhao Pu, Huangji Wang, and Fan Cheng. Accelerating diffusion sampling via exploiting local transition coherence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18284–18293, October 2025.

Algorithm 2 ParaPC-FM iterative update of order p

Require: Current state x_{t_i}; time steps $\{t_{i-p+1}, \ldots, t_{i+1}\}$; velocity predictions $\{v_{t_{i-p+1}}, \ldots, v_{t_{i+1}}\}$; update order p

$p_i \leftarrow \min(p-1,\, i)$
$\Delta t_i \leftarrow t_{i+1} - t_i$
for $m = 1$ **to** p_i **do**
    $r_m \leftarrow (t_{i-m} - t_i)/\Delta t_i$
end for
$r_{p_i+1} \leftarrow 1$
for $k = 1$ **to** $p_i + 1$ **do**
    for $m = 1$ **to** $p_i + 1$ **do**
        $R_{k,m} \leftarrow r_m^k$
    end for
    $b_k \leftarrow \Delta t_i/(k+1)$
end for
Solve the linear system $Ra = b$ to obtain $\{a_m\}_{m=1}^{p_i+1}$
for $m = 1$ **to** p_i **do**
    $D_m \leftarrow v_{t_{i-m}} - v_{t_i}$
end for
$D_{p_i+1} \leftarrow v_{t_{i+1}} - v_{t_i}$
$x_{t_{i+1}} \leftarrow x_{t_i} + \Delta t_i v_{t_i} + \sum_{m=1}^{p_i+1} a_m D_m$
return $x_{t_{i+1}}$

Algorithm 3 Truncated ParaPC-FM initialization of order q

Require: Reference state x_{t_ℓ}; initialization length j; time steps $\{t_{\ell-(q-1)}, \ldots, t_{\ell+j}\}$; velocity predictions $\{v_{t_{\ell-(q-1)}}, \ldots, v_{t_\ell}\}$; initialization order q

$q_\ell \leftarrow \min(q-1,\, \ell)$
for $k = 1$ **to** j **do**
    $\Delta t_{\ell,k} \leftarrow t_{\ell+k} - t_\ell$
    for $m = 1$ **to** q_ℓ **do**
        $r_m \leftarrow (t_{\ell-m} - t_\ell)/\Delta t_{\ell,k}$
    end for
    for $k' = 1$ **to** q_ℓ **do**
        for $m = 1$ **to** q_ℓ **do**
            $R_{k',m} \leftarrow r_m^{k'}$
        end for
        $b_{k'} \leftarrow \Delta t_{\ell,k}/(k'+1)$
    end for
    Solve the linear system $Ra = b$ to obtain $\{a_m\}_{m=1}^{q_\ell}$
    for $m = 1$ **to** q_ℓ **do**
        $D_m \leftarrow v_{t_{\ell-m}} - v_{t_\ell}$
    end for
    $x_{t_{\ell+k}} \leftarrow x_{t_\ell} + \Delta t_{\ell,k} v_{t_\ell} + \sum_{m=1}^{q_\ell} a_m D_m$
end for
return $\{x_{t_{\ell+1}}, \ldots, x_{t_{\ell+j}}\}$

A Algorithmic Details of ParaPC-FM

For completeness, we provide the full algorithmic descriptions of the ParaPC-FM iterative update and the truncated initialization procedures in Algorithms 2 and 3, respectively.

B Acceleration Guarantee

Proposition B.1 (Reduction to Heun's Method)**.** *When $p = 1$, the ParaPC-FM update in Eq.* (6) *reduces to Heun's method applied to the generative ODE.*

Definition B.2 (Exact Flow-Matching Trajectory)**.** Let $v_\theta(x,t)$ denote the learned velocity field of the flow-matching model. We define $x(t)$ as the *exact continuous-time trajectory* of the generative process, given by the solution of the ordinary differential equation

$$\frac{dx(t)}{dt} = v_\theta(x(t),t), \qquad t \in [0,1], \tag{16}$$

with initial condition $x(0) = x_0$. The value $x(t)$ therefore represents the ground-truth state of the flow-matching model at time t.

Definition B.3 (ParaPC-FM Update Operator)**.** Given the state x'_{t_i} and $\{x_{t_n}\}_{n=i-p+1}^{i+1}$, the operator Φ_p is defined as

$$\Phi_p\big(\{x_{t_n}\}_{n=i-p+1}^{i+1},\, x'_{t_i}\big) \;\coloneqq\; x'_{t_i} + \Delta t_i\, v_\theta(x_{t_i},t_i) + \sum_{m=1}^{p} a_m\, D_m, \tag{17}$$

where $\Delta t_i = t_{i+1} - t_i$ and the velocity difference terms are defined as

$$D_m \;\coloneqq\; \begin{cases} v_\theta\big(x_{t_{i-m}}, t_{i-m}\big) - v_\theta(x_{t_i},t_i)\,, & m < p, \\[2ex] v_\theta\big(x_{t_{i+1}}, t_{i+1}\big) - v_\theta(x_{t_i},t_i)\,, & m = p. \end{cases} \tag{18}$$

Definition B.4 (Local Accuracy)**.** We define the local truncation error at step i as

$$\varepsilon_i \;\coloneqq\; x_{t_i} - x(t_i).$$

If $\varepsilon_i = \mathcal{O}\big(\delta^{\,p+1}\big)$, we say that x_{t_i} achieves *p-th order local accuracy*.

Remark B.5. Definition B.4 characterizes the error introduced by a single iteration in numerical analysis [20].

Proposition B.6 (Moment-matching representation)**.** *Given the coefficients $\{a_m\}_{m=1}^p$ specified in Eq.* (6) *and Eq.* (11)*, define*

$$c_m \;\coloneqq\; \frac{r_m}{\Delta t_i}\, a_m, \qquad m = 1,\dots,p.$$

Then the coefficients $\{c_m\}_{m=1}^p$ satisfy

$$\sum_{m=1}^{p} c_m\, r_m^{k-1} = \frac{1}{k+1}, \qquad k = 1,\dots,p. \tag{19}$$

Remark B.7. Proposition B.6 transforms the coefficients $\{a_m\}_{m=1}^p$ into a set of step-size–independent coefficients $\{c_m\}_{m=1}^p$ by factoring out the time step, which facilitates the analysis of the order of accuracy.

Assumption B.8 (Lipschitz Continuity of the Velocity Field)**.** *The learned velocity field $v_\theta(x,t)$ is Lipschitz continuous with respect to the state variable x, uniformly in time. That is, there exists a constant $L > 0$ such that for all $x_1, x_2 \in \mathbb{R}^d$ and all $t \in [0,T]$,*

$$\big\|v_\theta(x_1,t) - v_\theta(x_2,t)\big\| \;\le\; L\,\|x_1 - x_2\|.$$

We refer to L as the Lipschitz constant of the model prediction.

Assumption B.9 (Temporal Regularity)**.** *The time derivatives*

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k} v_\theta(x,t), \quad k = 1,\dots,p,$$

exist and are continuous.

Remark B.10. Assumption B.8 is standard in the analysis of ODEs. In addition, Assumption B.9 ensures the validity of Taylor expansions with respect to the time variable, which is a common requirement in the analysis of high-order numerical methods.

Proposition B.11 (Order of Accuracy)**.** *Under the conditions of Assumption B.9, if the inputs $\{x_{t_n}\}_{n=i-p+1}^{i+1}$ coincide with the exact solution $\{x(t_n)\}_{n=i-p+1}^{i+1}$ of the underlying ODE, with x'_{t_i} satisfying $x'_{t_i} = x(t_i)$, Φ_p achieves $(p+1)$-th order local accuracy.*

Proposition B.12 (Stability of $(p+1)$-th Order Accuracy)**.** *Assume Assumption B.8 and Assumption B.9 hold. Assuming that, at the u-th iteration, the states prior to the sliding window are regarded as converged, and that the states from the sliding window start up to index $i+1$ achieve $(p+1)$-th order accuracy, then at the $(u+1)$-th iteration, the states updated by the ParaPC-FM update in Eq.* (6) *also achieve $(p+1)$-th order accuracy.*

Remark B.13. Proposition B.12 guarantees that the error upper bound does not increase across iterations, which makes the assumption in Proposition B.16—that the unconverged states achieve $(p+1)$-th order accuracy—reasonable.

Definition B.14 (ParaPC-FM Initialization Operator)**.** Let x_{t_ℓ} denote the last time point at which a velocity prediction $v_\theta(x_{t_\ell}, t_\ell)$ is available. Given the past states $\{x_{t_{\ell-m}}\}_{m=0}^{q-1}$, we define the q-th order initialization operator Ψ_q as

$$\Psi_q\Big(\{x_{t_{\ell-m}}\}_{m=0}^{q-1},\, j\Big) \;\coloneqq\; x_{t_\ell} + \Delta t_{\ell,j}\, v_\theta(x_{t_\ell}, t_\ell) + \sum_{m=1}^{q-1} a_m\, D_m, \tag{20}$$

where $\Delta t_{\ell,j} = t_{\ell+j} - t_\ell$ and the velocity difference terms are defined as

$$D_m \;\coloneqq\; v_\theta\big(x_{t_{\ell-m}}, t_{\ell-m}\big) - v_\theta(x_{t_\ell}, t_\ell)\,. \tag{21}$$

Proposition B.15 (Accuracy of Truncated Initialization)**.** *Under the conditions of Assumption B.9, if the inputs $\{x_{t_{\ell-m}}\}_{m=0}^{q-1}$ coincide with the exact solution $\{x(t_{\ell-m})\}_{m=0}^{q-1}$ of the underlying ODE, then Ψ_q achieves q-th order local accuracy.*

Proposition B.16 (Sufficient Conditions for Faster Convergence)**.** *Assume Assumption B.8 and Assumption B.9 hold. Suppose that after the $(u+1)$-th iteration, the unconverged states within the sliding window achieve $(p+1)$-th order accuracy with respect to the exact ODE solution. Then the ParaPC-FM initialization* (11) *achieves $min(p+1, q)$-th order accuracy.*

Remark B.17. From Proposition B.16, ParaPC-FM with a p-th order update and a q-th order initialization achieves $\min(p+1, q)$-th order accuracy. Without special initialization, i.e., $x_{t_{\ell+k}}^{(u+1)} = x_{t_\ell}^{(u+1)}$, the state $x_{t_{\ell+k}}^{(u+1)}$ only attains zeroth-order accuracy, namely

$$x_{t_{\ell+k}}^{(u+1)} - x(t_{\ell+k}) = \mathcal{O}(\delta).$$

When the step size is sufficiently small, i.e., δ is small, and $\min(p+1, q) > 0$—equivalently, $p \geq 0$ and $q \geq 1$—the ParaPC-FM initialization enables faster convergence. In contrast, overly large step sizes can lead to excessive errors, which is consistent with our empirical observations.

Proposition B.18 (Upper Bound on Parallel Iterations)**.** *Consider a generative process discretized on T time steps $\{t_0, t_1, \ldots, t_T\}$. When applying the ParaPC-FM sliding-window parallel inference scheme, the total number of parallel iterations required to complete sampling is at most T.*

C Proofs

This appendix contains the proofs of the following propositions: Proposition B.1 (see Section C.1), Proposition B.6 (see Section C.2), Proposition B.11 (see Section C.3), Proposition B.12 (see Section C.4), Proposition B.15 (see Section C.5), Proposition B.16 (see Section C.6), and Proposition B.18 (see Section C.7).

C.1 Proof of Proposition B.1

Proof. We start from the ParaPC-FM update rule in Eq. (6),

$$x_{t_{i+1}}^{(u+1)} = x_{t_i}^{(u)} + \Delta t_i\, v_\theta(x_{t_i}^{(u)}, t_i) + \sum_{m=1}^{p} a_m D_m^{(u)}, \tag{22}$$

where the velocity difference term is defined as

$$D_m = v_\theta(x_{t_{i-m}}^{(u)}, t_{i-m}) - v_\theta(x_{t_i}^{(u)}, t_i). \tag{23}$$

When $p = 1$, the summation contains a single term,

$$D_1 = v_\theta(x_{t_{i+1}}^{(u)}, t_{i+1}) - v_\theta(x_{t_i}^{(u)}, t_i), \tag{24}$$

and the update reduces to

$$x_{t_{i+1}}^{(u+1)} = x_{t_i}^{(u)} + \Delta t_i \, v_\theta(x_{t_i}^{(u)}, t_i) + a_1 \Big(v_\theta(x_{t_{i+1}}^{(u)}, t_{i+1}) - v_\theta(x_{t_i}^{(u)}, t_i) \Big). \tag{25}$$

For $p = 1$, the moment-matching condition yields

$$a_1 = \frac{\Delta t_i}{2}. \tag{26}$$

Substituting this value gives

$$x_{t_{i+1}}^{(u+1)} = x_{t_i}^{(u)} + \Delta t_i \, v_\theta(x_{t_i}^{(u)}, t_i) + \frac{\Delta t_i}{2} \Big(v_\theta(x_{t_{i+1}}^{(u)}, t_{i+1}) - v_\theta(x_{t_i}^{(u)}, t_i) \Big) \tag{27}$$

$$= x_{t_i}^{(u)} + \frac{\Delta t_i}{2} \Big(v_\theta(x_{t_i}^{(u)}, t_i) + v_\theta(x_{t_{i+1}}^{(u)}, t_{i+1}) \Big). \tag{28}$$

This is exactly the update of Heun's method (explicit trapezoidal rule), which averages the velocity at the current state and the velocity evaluated at the next time step. Therefore, when $p = 1$, the ParaPC-FM update reduces to Heun's method. □

C.2 Proof of Proposition B.6

Proof. By assumption, the coefficients $\{a_m\}_{m=1}^p$ satisfy

$$\sum_{m=1}^{p} a_m \, r_m^k = \frac{\Delta t_i}{k+1}, \qquad k = 1, \ldots, p.$$

Define $c_m = \frac{r_m}{\Delta t_i} \, a_m$. Substituting this definition into the left-hand side of (19), we obtain

$$\sum_{m=1}^{p} c_m \, r_m^{k-1} = \sum_{m=1}^{p} \frac{r_m}{\Delta t_i} a_m \, r_m^{k-1} = \frac{1}{\Delta t_i} \sum_{m=1}^{p} a_m \, r_m^k.$$

Applying the assumed moment condition yields

$$\sum_{m=1}^{p} c_m \, r_m^{k-1} = \frac{1}{\Delta t_i} \cdot \frac{\Delta t_i}{k+1} = \frac{1}{k+1}, \qquad k = 1, \ldots, p,$$

which proves (19). □

C.3 Proof of Proposition B.11

Proof. We analyze the local truncation error of the p-th order one-step update operator Φ_p over the time interval ending at t_{i+1}. Specifically, assuming that the inputs $\{x_{t_n}\}_{n=i-p+1}^{i+1}$ coincide with the exact solution values $\{x(t_n)\}_{n=i-p+1}^{i+1}$, and that $x'_{t_i} = x(t_i)$, we aim to bound

$$x(t_{i+1}) - \Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1}, \, x(t_i)\big) \, . \tag{29}$$

By Taylor expansion around t_i, the exact solution satisfies

$$x(t_{i+1}) = x(t_i) + \sum_{k=0}^{p} \frac{(t_{i+1} - t_i)^{k+1}}{(k+1)!} \frac{d^k}{dt^k} v_\theta(x(t_i), t_i) + \mathcal{O}\big(|t_{i+1} - t_i|^{p+2}\big) \, . \tag{30}$$

Next, consider the numerical update produced by Φ_p when all inputs are exact. For each $n \in \{\, i-p+1, \ldots, i-1, \, i+1 \,\}$, the velocity evaluated at t_n admits the expansion

$$v_\theta(x(t_n), t_n) = v_\theta(x(t_i), t_i) + \sum_{k=1}^{p} \frac{(t_n - t_i)^k}{k!} \frac{d^k}{dt^k} v_\theta(x(t_i), t_i) + \mathcal{O}\big(|t_n - t_i|^{p+1}\big) \, . \tag{31}$$

Then rewrite the coefficients as

$$c_m \;\coloneqq\; \frac{r_m}{\Delta t_i}\, a_m.$$

By Proposition B.6, the coefficients $\{c_m\}_{m=1}^{p}$ satisfy the moment conditions

$$\sum_{m=1}^{p} c_m\, r_m^{k-1} = \frac{1}{k+1}, \qquad k = 1, \dots, p,$$

where

$$r_m = \begin{cases} \frac{t_{i-m}-t_i}{t_{i+1}-t_i}, & m < p, \\ 1, & m = p. \end{cases}$$

Substituting (31) into the definition of the velocity difference terms and then into the operator Φ_p, we obtain

$$\begin{aligned}
&\Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big) \\
&= x(t_i) + (t_{i+1}-t_i)\, v_\theta(x(t_i), t_i) + \sum_{m=1}^{p} a_m \Big[v_\theta(x(t_{i-m}), t_{i-m}) - v_\theta(x(t_i), t_i) \Big] \\
&= x(t_i) + (t_{i+1}-t_i)\, v_\theta(x(t_i), t_i) \\
&\quad + \sum_{m=1}^{p-1} \frac{c_m \Delta t_i}{r_m} \left[\sum_{k=1}^{p} \frac{(t_{i-m}-t_i)^k}{k!} \frac{d^k}{dt^k} v_\theta(x(t_i), t_i) + \mathcal{O}\big(|t_{i-m}-t_i|^{\,p+1}\big) \right] \\
&\quad + c_p (t_{i+1}-t_i) \left[\sum_{k=1}^{p} \frac{(t_{i+1}-t_i)^k}{k!} \frac{d^k}{dt^k} v_\theta(x(t_i), t_i) + \mathcal{O}\big(|t_{i+1}-t_i|^{\,p+1}\big) \right] \\
&= x(t_i) + (t_{i+1}-t_i)\, v_\theta(x(t_i), t_i) \\
&\quad + \sum_{k=1}^{p} \left(\sum_{m=1}^{p} c_m\, r_m^{k-1} \right) \frac{(t_{i+1}-t_i)^{k+1}}{k!} \frac{d^k}{dt^k} v_\theta(x(t_i), t_i) + \mathcal{O}\big(\delta^{\,p+2}\big) \\
&= x(t_i) + (t_{i+1}-t_i)\, v_\theta(x(t_i), t_i) + \sum_{k=1}^{p} \frac{(t_{i+1}-t_i)^{k+1}}{(k+1)!} \frac{d^k}{dt^k} v_\theta(x(t_i), t_i) + \mathcal{O}\big(\delta^{\,p+2}\big), \qquad (32)
\end{aligned}$$

where

$$\delta \;\coloneqq\; \max\left(\max_{1 \le m < p} |t_{i-m} - t_i|,\; |t_{i+1} - t_i| \right).$$

Consequently, all Taylor terms up to order $(t_{i+1}-t_i)^{p+1}$ in (32) exactly match those in the exact expansion (30).

Combining the two expansions yields

$$x(t_{i+1}) - \Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big) = \mathcal{O}\big(\delta^{\,p+2}\big),$$

which shows that Φ_p achieves $(p+1)$-th order local accuracy. □

C.4 Proof of Proposition B.12

Proof. We decompose the error at t_{i+1} as

$$\begin{aligned}
x_{t_{i+1}}^{(u+1)} - x(t_{i+1}) = &\underbrace{\Phi_p\Big(\{x_{t_n}^{(u)}\}_{n=i-p+1}^{i+1},\, x_{t_i}^{(u+1)}\Big) - \Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big)}_{\text{propagated error}} \\
&+ \underbrace{\Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big) - x(t_{i+1})}_{\text{local truncation error}}. \qquad (33)
\end{aligned}$$

By Proposition B.11, the Φ_p satisfies

$$\Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big) - x(t_{i+1}) = \mathcal{O}\Big(\delta_{t_{i+1}}^{p+2}\Big).$$

Next, we bound the propagated error. By Assumption B.8, the velocity field $v_\theta(x,t)$ is Lipschitz continuous with respect to x with constant L, i.e.,

$$\|v_\theta(x,t) - v_\theta(y,t)\| \le L\|x-y\|, \qquad \forall\, x, y,\ \forall\, t.$$

Using the Lipschitz continuity of v_θ, we have

$$\begin{aligned}
&\Big\|\Phi_p\Big(\{x_{t_n}^{(u)}\}_{n=i-p+1}^{i+1},\, x_{t_i}^{(u+1)}\Big) - \Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big)\Big\| \\
&\le \Big\|x_{t_i}^{(u+1)} - x(t_i)\Big\| + \Delta t_i \Big\|v_\theta(x_{t_i}^{(u)}, t_i) - v_\theta(x(t_i), t_i)\Big\| \\
&\quad + \sum_{m=1}^{p-1} |a_m|\Big(\Big\|v_\theta(x_{t_{i-m}}^{(u)}, t_{i-m}) - v_\theta(x(t_{i-m}), t_{i-m})\Big\| + \Big\|v_\theta(x_{t_i}^{(u)}, t_i) - v_\theta(x(t_i), t_i)\Big\| \Big) \\
&\quad + |a_p|\Big(\Big\|v_\theta(x_{t_{i+1}}^{(u)}, t_{i+1}) - v_\theta(x(t_{i+1}), t_{i+1})\Big\| + \Big\|v_\theta(x_{t_i}^{(u)}, t_i) - v_\theta(x(t_i), t_i)\Big\| \Big) \\
&\le \Big\|x_{t_i}^{(u+1)} - x(t_i)\Big\| + \Delta t_i L \Big\|x_{t_i}^{(u)} - x(t_i)\Big\| \\
&\quad + \sum_{m=1}^{p-1} \left|\frac{c_m(t_{i+1}-t_i)}{r_m}\right| L\Big(\Big\|x_{t_{i-m}}^{(u)} - x(t_{i-m})\Big\| + \Big\|x_{t_i}^{(u)} - x(t_i)\Big\| \Big) \\
&\quad + \left|\frac{c_p(t_{i+1}-t_i)}{r_p}\right| L\Big(\Big\|x_{t_{i+1}}^{(u)} - x(t_{i+1})\Big\| + \|x_{t_i} - x(t_i)\| \Big) \\
&\le C_1 \max\left(\max_{i-p+1\le n\le i+1} \Big\|x_{t_n}^{(u)} - x(t_n)\Big\|,\ \Big\|x_{t_i}^{(u+1)} - x(t_i)\Big\| \right),
\end{aligned} \tag{34}$$

where

$$r_m = \begin{cases} \frac{t_{i-m}-t_i}{t_{i+1}-t_i}, & m < p, \\ 1, & m = p. \end{cases}$$

By assumption,

$$\max_{i-p+1\le n\le i+1} \Big\|x_{t_n}^{(u)} - x(t_n)\Big\| \le C \max_{i-p+1\le n\le i+1} \{\delta_{t_n}^{p+2}\}.$$

At the same time, we assume that the starting point of the sliding window and all preceding states have converged, i.e., their values coincide with the exact solution. Under this assumption, the states recursively propagated from the sliding-window start up to x_{t_i} satisfy

$$\Big\|x_{t_n}^{(u+1)} - x(t_n)\Big\|_{n\le i} = \mathcal{O}\Big(\delta_{t_n}^{p+2}\Big).$$

Therefore, we have

$$\Phi_p\Big(\{x_{t_n}^{(u)}\}_{n=i-p+1}^{i+1},\, x_{t_i}^{(u+1)}\Big) - \Phi_p\big(\{x(t_n)\}_{n=i-p+1}^{i+1},\, x(t_i)\big) = \mathcal{O}\Big(\max_{i-p+1\le n\le i+1} \{\delta_{t_n}^{p+2}\}\Big).$$

Combining the propagated error and the local truncation error in (33), we conclude that

$$\Big\|x_{t_{i+1}}^{(u+1)} - x(t_{i+1})\Big\| = \mathcal{O}\big(\delta^{\,p+2}\big),$$

where

$$\delta = \max_{i-p+1\le n\le i+1} \{\delta_{t_n}\}.$$

Therefore, the $(p+1)$-th order accuracy is preserved after one ParaPC-FM update. □

C.5 Proof of Proposition B.15

Proof. We analyze the local truncation error of the q-th order operator Ψ_q over the time interval ending at $t_{\ell+j}$. Specifically, assuming that the inputs $\{x_{t_{\ell-m}}\}_{m=0}^{q-1}$ coincide with the exact solution $\{x(t_{\ell-m})\}_{m=0}^{q-1}$ of the underlying ODE, we aim to bound

$$x(t_{\ell+j}) - \Psi_q\Big(\{x_{t_{\ell-m}}\}_{m=0}^{q-1},\, j\Big)\,. \tag{35}$$

By Taylor expansion around t_ℓ, the exact solution satisfies

$$x(t_{\ell+j}) = x(t_\ell) + \sum_{k=0}^{q-1} \frac{(t_{\ell+j}-t_\ell)^{k+1}}{(k+1)!}\frac{d^k}{dt^k} v_\theta(x(t_\ell),t_\ell) + \mathcal{O}\big(|t_{\ell+j}-t_\ell|^{q+1}\big)\,. \tag{36}$$

Next, consider the numerical update produced by Ψ_q when all inputs are exact. For each $m = 1,\dots,q-1$, the velocity evaluated at $t_{\ell-m}$ admits the expansion

$$v_\theta(x(t_{\ell-m}),t_{\ell-m}) = v_\theta(x(t_\ell),t_\ell) + \sum_{k=1}^{q-1} \frac{(t_{\ell-m}-t_\ell)^k}{k!}\frac{d^k}{dt^k} v_\theta(x(t_\ell),t_\ell) + \mathcal{O}(|t_{\ell-m}-t_\ell|^q)\,. \tag{37}$$

Then rewrite the coefficients as

$$c_m \;\coloneqq\; \frac{r_m}{t_{\ell+j}-t_\ell}\,a_m.$$

By Proposition B.6, the coefficients $\{c_m\}_{m=1}^{q-1}$ satisfy the moment conditions

$$\sum_{m=1}^{q-1} c_m\, r_m^{k-1} = \frac{1}{k+1}, \qquad k = 1,\dots,q-1,$$

where

$$r_m = \frac{t_{\ell-m}-t_\ell}{t_{\ell+j}-t_\ell}.$$

Substituting (37) into the definition of the velocity difference terms and then into the operator Ψ_q, we obtain

$$\begin{aligned}
&\Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big)\\
&= x(t_\ell) + (t_{\ell+j}-t_\ell)\, v_\theta(x(t_\ell),t_\ell) + \sum_{m=1}^{q-1} a_m\Big[v_\theta(x(t_{\ell-m}),t_{\ell-m}) - v_\theta(x(t_\ell),t_\ell)\Big]\\
&= x(t_\ell) + (t_{\ell+j}-t_\ell)\, v_\theta(x(t_\ell),t_\ell)\\
&\quad + \sum_{m=1}^{q-1} \frac{c_m(t_{\ell+j}-t_\ell)}{r_m}\left[\sum_{k=1}^{q-1}\frac{(t_{\ell-m}-t_\ell)^k}{k!}\frac{d^k}{dt^k}v_\theta(x(t_\ell),t_\ell) + \mathcal{O}(|t_{\ell-m}-t_\ell|^{\,q})\right]\\
&= x(t_\ell) + (t_{\ell+j}-t_\ell)\, v_\theta(x(t_\ell),t_\ell)\\
&\quad + \sum_{k=1}^{q-1}\left(\sum_{m=1}^{q-1} c_m\, r_m^{k-1}\right)\frac{(t_{\ell+j}-t_\ell)^{k+1}}{k!}\frac{d^k}{dt^k}v_\theta(x(t_\ell),t_\ell) + \mathcal{O}\big(\delta^{\,q+1}\big)\\
&= x(t_\ell) + (t_{\ell+j}-t_\ell)\, v_\theta(x(t_\ell),t_\ell) + \sum_{k=1}^{q-1}\frac{(t_{\ell+j}-t_\ell)^{k+1}}{(k+1)!}\frac{d^k}{dt^k}v_\theta(x(t_\ell),t_\ell) + \mathcal{O}\big(\delta^{\,q+1}\big)\,,
\end{aligned} \tag{38}$$

where

$$\delta = \max\left(\max_{1\le m<q}|t_{\ell-m}-t_\ell|,\, |t_{\ell+j}-t_\ell|\right).$$

Consequently, all Taylor terms up to order $(t_{\ell+j}-t_\ell)^q$ in (38) exactly match those in the exact expansion (36).

Combining the two expansions yields

$$x(t_{\ell+j}) - \Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big) = \mathcal{O}\big(\delta^{\,q+1}\big)\,,$$

which shows that Ψ_q achieves q-th order local accuracy. □

C.6 Proof of Proposition B.16

Proof. We decompose the error at $t_{\ell+j}$ as

$$x_{t_{\ell+j}}^{(u+1)} - x(t_{\ell+j}) = \underbrace{\Psi_q\Big(\{x_{t_{\ell-m}}^{(u+1)}\}_{m=0}^{q-1},\, j\Big) - \Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big)}_{\text{propagated error}} + \underbrace{\Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big) - x(t_{\ell+j})}_{\text{local truncation error}}. \tag{39}$$

By Proposition B.15, the Ψ_q satisfies

$$\Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big) - x(t_{\ell+j}) = \mathcal{O}\Big(\delta_{t_{\ell+j}}^{q+1}\Big),$$

where

$$\delta_{t_{\ell+j}} = \max\left(\max_{1\le m<q} |t_{\ell-m} - t_\ell|,\, |t_{\ell+j} - t_\ell|\right).$$

Next, we bound the propagated error. By Assumption B.8, the velocity field $v_\theta(x,t)$ is Lipschitz continuous with respect to x with constant L, i.e.,

$$\|v_\theta(x,t) - v_\theta(y,t)\| \le L\|x-y\|, \qquad \forall\, x, y,\ \forall\, t.$$

Using the Lipschitz continuity of v_θ, we have

$$\begin{aligned}
&\Big\|\Psi_q\Big(\{x_{t_{\ell-m}}^{(u+1)}\}_{m=0}^{q-1},\, j\Big) - \Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big)\Big\| \\
&\le \Big\|x_{t_\ell}^{(u+1)} - x(t_\ell)\Big\| + (t_{\ell+j} - t_\ell)\Big\|v_\theta(x_{t_\ell}^{(u+1)}, t_\ell) - v_\theta(x(t_\ell), t_\ell)\Big\| \\
&\quad + \sum_{m=1}^{q-1} |a_m|\Big(\Big\|v_\theta(x_{t_{\ell-m}}^{(u+1)}, t_{\ell-m}) - v_\theta(x(t_{\ell-m}), t_{\ell-m})\Big\| + \Big\|v_\theta(x_{t_\ell}^{(u+1)}, t_\ell) - v_\theta(x(t_\ell), t_\ell)\Big\|\Big) \\
&\le \Big\|x_{t_\ell}^{(u+1)} - x(t_\ell)\Big\| + L(t_{\ell+j} - t_\ell)\Big\|x_{t_\ell}^{(u+1)} - x(t_\ell)\Big\| \\
&\quad + \sum_{m=1}^{q-1} \left|\frac{c_m(t_{\ell+j} - t_\ell)}{r_m}\right| L\Big(\Big\|x_{t_{\ell-m}}^{(u+1)} - x(t_{\ell-m})\Big\| + \Big\|x_{t_\ell}^{(u+1)} - x(t_\ell)\Big\|\Big) \\
&\le C_1 \max_{0\le m\le q-1} \Big\|x_{t_{\ell-m}}^{(u+1)} - x(t_{\ell-m})\Big\|,
\end{aligned} \tag{40}$$

where $r_m = \frac{t_{\ell-m} - t_\ell}{t_{\ell+j} - t_\ell}$.

By assumption,

$$\max_{0\le m\le q-1} \Big\|x_{t_{\ell-m}}^{(u+1)} - x(t_{\ell-m})\Big\| \le C \max_{0\le m\le q-1} \Big\{\delta_{\ell-m}^{p+2}\Big\},$$

which yields

$$\Psi_q\Big(\{x_{t_{\ell-m}}^{(u+1)}\}_{m=0}^{q-1},\, j\Big) - \Psi_q\Big(\{x(t_{\ell-m})\}_{m=0}^{q-1},\, j\Big) = \mathcal{O}\Big(\max_{0\le m\le q-1} \Big\{\delta_{\ell-m}^{p+2}\Big\}\Big).$$

Combining the propagated error and the local truncation error in (39), we conclude that

$$\Big\|x_{t_{\ell+j}}^{(u+1)} - x(t_{\ell+j})\Big\| = \mathcal{O}\left(\max\left[\max_{0\le m\le q-1}\{\delta_{\ell-m}\},\, \delta_{t_{\ell+j}}\right]^{min(p+2,q+1)}\right).$$

Therefore, the ParaPC-FM initialization (11) achieves $min(p+1, q)$-th order accuracy. □

C.7 Proof of Proposition B.18

Proof. ParaPC-FM proceeds by maintaining a sliding window whose left boundary corresponds to the most recent converged time index. Let $i^{(u)}$ denote the reference index at the beginning of the u-th parallel iteration, with $i^{(0)} = 0$.

At each iteration, a non-negative shift $s^{(u)} \geq 1$ is determined according to the convergence criterion in Eq. (14), and the reference index is advanced as

$$i^{(u+1)} = i^{(u)} + s^{(u)}.$$

By construction, $s^{(u)} \geq 1$ whenever the process has not terminated, since at least the reference state itself is always regarded as converged.

The sampling process terminates once the reference index reaches T. Therefore, the total number of parallel iterations Q satisfies

$$\sum_{u=0}^{Q-1} s^{(u)} = T.$$

Since each $s^{(u)} \geq 1$, it follows immediately that

$$Q \leq T.$$

This establishes that ParaPC-FM never requires more parallel iterations than the number of time steps in the corresponding sequential generation process. □

D End-to-End Error Analysis

We now analyze the end-to-end error of ParaPC-FM. The goal is to separate the standard discretization error of the underlying sequential solver from the additional error introduced by adaptive sliding-window advancement.

Let x_N^{para} denote the final output produced by Algorithm 1, let x_N^{seq} denote the output of the corresponding sequential solver on the same time grid $\{t_i\}_{i=0}^N$, and let $x(t_N)$ be the exact ODE solution introduced in Definition B.2. We decompose

$$x_N^{\mathrm{para}} - x(t_N) = \underbrace{x_N^{\mathrm{para}} - x_N^{\mathrm{seq}}}_{\text{parallel early-stopping error}} + \underbrace{x_N^{\mathrm{seq}} - x(t_N)}_{\text{discretization error}}. \tag{41}$$

The second term is the standard discretization error. Therefore, it remains to bound the first term, which measures the deviation caused by terminating the parallel correction according to the tolerance criterion in Eq. (14).

Assumption D.1 (Local Contractivity of Sliding-Window Correction)**.** *For each sliding window, let $X_i^{(u)}$ collect the states in the current window at the u-th parallel correction iteration, and let*

$$X_i^{(u+1)} = \mathcal{T}_i(X_i^{(u)})$$

denote the corresponding ParaPC-FM correction map. We assume that, for sufficiently small step size, $\mathcal{T}_i$ is locally contractive around its fixed point $X_i^\star$, i.e.,

$$\|\mathcal{T}_i(X) - \mathcal{T}_i(Y)\| \leq \rho_i \|X - Y\|, \qquad 0 \leq \rho_i < 1. \tag{42}$$

Moreover, the same a posteriori residual bound holds for each accepted state inside the window:

$$\begin{aligned}
\left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^\star \right\|_2 &\leq \sum_{r=1}^{\infty} \left\| x_{t_{i+k}}^{(u+r+1)} - x_{t_{i+k}}^{(u+r)} \right\|_2 \\
&\leq \sum_{r=1}^{\infty} \rho_i^r \left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{(u)} \right\|_2 \\
&\leq \frac{\left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{(u)} \right\|_2}{1 - \rho_i},
\end{aligned} \tag{43}$$

where $x_{t_{i+k}}^\star$ denotes the fully converged fixed-point value at t_{i+k} within the current window. We write

$$\rho := \max_i \rho_i < 1.$$

Proposition D.2 (End-to-end error bound)**.** *Assume Assumption B.8 and Assumption D.1 hold. Let*

$$h \coloneqq \max_{0 \le k < N} \Delta t_k, \qquad T \coloneqq t_N - t_0.$$

Suppose that the corresponding sequential solver on the same time grid satisfies the standard discretization bound

$$\|x_N^{\mathrm{seq}} - x(t_N)\|_2 \le C_{\mathrm{disc}} h^r, \tag{44}$$

for some constants $C_{\mathrm{disc}} > 0$ and $r > 0$. Then the output of ParaPC-FM satisfies

$$\|x_N^{\mathrm{para}} - x(t_N)\|_2 \le C_{\mathrm{disc}} h^r + \exp(C_\Phi T) \frac{T\,\mathrm{tol}\sqrt{d}}{1-\rho}, \tag{45}$$

where $C_\Phi > 0$ is a stability constant depending on the Lipschitz constant of v_θ and the coefficients of the ParaPC-FM update, but independent of h and tol*.*

Consequently, the total error consists of the usual discretization floor and an additional tolerance-controlled parallelization error:

$$\|x_N^{\mathrm{para}} - x(t_N)\|_2 = \mathcal{O}(h^r) + \mathcal{O}(\mathrm{tol}). \tag{46}$$

Proof. We first bound the error introduced by early acceptance in one sliding window. By Algorithm 1, a state $x_{t_{i+k}}^{(u+1)}$ is accepted only if

$$\left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{(u)} \right\|_2^2 \le \mathrm{tol}^2 \Delta t_{i+k-1}^2 d. \tag{47}$$

Equivalently,

$$\left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{(u)} \right\|_2 \le \mathrm{tol}\, \Delta t_{i+k-1} \sqrt{d}. \tag{48}$$

Applying the componentwise residual estimate in Assumption D.1, we obtain

$$\left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{\star} \right\|_2 \le \frac{\left\| x_{t_{i+k}}^{(u+1)} - x_{t_{i+k}}^{(u)} \right\|_2}{1-\rho_i} \le \frac{\mathrm{tol}\, \Delta t_{i+k-1}\sqrt{d}}{1-\rho_i} \le \frac{\mathrm{tol}\, \Delta t_{i+k-1}\sqrt{d}}{1-\rho}. \tag{49}$$

Thus, each accepted state differs from the fully converged fixed-point value of the same window by at most a quantity proportional to tol.

Next, define the deviation between the ParaPC-FM trajectory and the corresponding sequential trajectory by

$$e_k \coloneqq \left\| x_{t_k}^{\mathrm{para}} - x_{t_k}^{\mathrm{seq}} \right\|_2. \tag{50}$$

Since the ParaPC-FM update operator Φ_p is built from evaluations of v_θ and finite linear combinations of velocity differences, Assumption B.8 implies the stability estimate

$$e_{k+1} \le (1 + C_\Phi \Delta t_k) e_k + \xi_k, \tag{51}$$

where ξ_k denotes the local perturbation caused by accepting a parallel iterate before full fixed-point convergence. By Eq. (49), this perturbation satisfies

$$\xi_k \le \frac{\mathrm{tol}\, \Delta t_k \sqrt{d}}{1-\rho}. \tag{52}$$

For completeness, we briefly justify Eq. (51). Using the definition of Φ_p in Definition B.3, the difference between two updates is bounded by the difference between their input states and the corresponding velocity evaluations. Since v_θ is Lipschitz in x, and since the coefficients $\{a_m\}_{m=1}^p$ are of order $\mathcal{O}(\Delta t_k)$ by the moment-matching construction, there exists a constant $C_\Phi > 0$ such that the propagated deviation over one step is bounded by $(1 + C_\Phi \Delta t_k) e_k$. The remaining discrepancy is exactly the fixed-point residual ξ_k introduced by early acceptance.

Applying the discrete Grönwall inequality to Eq. (51), and using $e_0 = 0$, gives

$$\begin{aligned} e_N &\le \exp(C_\Phi T) \sum_{k=0}^{N-1} \xi_k \\ &\le \exp(C_\Phi T) \frac{\mathrm{tol}\sqrt{d}}{1-\rho} \sum_{k=0}^{N-1} \Delta t_k \\ &= \exp(C_\Phi T) \frac{T\,\mathrm{tol}\sqrt{d}}{1-\rho}. \end{aligned} \tag{53}$$

Table 5: Tolerance settings used in Tables 1 and 3.

Update order	FLUX.1-dev			Stable Diffusion 3	Stable Diffusion 3.5
	25 steps	50 steps	100 steps	50 steps	50 steps
$p = 0$	1.0	0.5	0.4	0.5	1.0
$p = 1$	1.0	0.5	0.4	0.5	1.0
$p = 2$	1.0	0.5	0.5	0.5	1.0

Finally, by the decomposition in Eq. (41) and the triangle inequality,

$$\begin{aligned}\|x_N^{\text{para}} - x(t_N)\|_2 &\le \|x_N^{\text{para}} - x_N^{\text{seq}}\|_2 + \|x_N^{\text{seq}} - x(t_N)\|_2 \\ &\le \exp(C_\Phi T)\frac{T\,\text{tol}\sqrt{d}}{1-\rho} + C_{\text{disc}}h^r. \end{aligned} \tag{54}$$

This proves Eq. (45). □

E Additional Experimental Details

E.1 Implementation Details for ParaDiGMS-style Experiments

Common settings. All ParaDiGMS-style [29] experiments use the same inference codebase, prompt set[3], and random seeds across compared configurations. Images are generated at 512×512 resolution. We use classifier-free guidance [16] with guidance scale 3.5. The sliding-window size is fixed to $n = 8$, matching the number of parallel devices. For the main comparisons in Tables 1 and 3, the convergence tolerance settings are summarized in Table 5. In the tolerance study, we explicitly vary tol to analyze its effect on correction steps, wall-clock time, RMSE, and image quality.

Hardware and timing. The ParaDiGMS-style experiments are run on a server equipped with 8 Ascend 910B3 NPUs, each with 64 GB memory. Wall-clock time is measured for end-to-end image generation under the same hardware setting, excluding one-time model loading overhead.

RMSE computation. For RMSE evaluation, both the generated image and the corresponding sequential reference image are first converted to RGB and normalized to $[0, 1]$ by dividing pixel values by 255. Given the normalized generated image x and reference image x^{ref}, we compute

$$\text{RMSE}(x, x^{\text{ref}}) = \sqrt{\frac{1}{D}\sum_{d=1}^{D}\left(x_d - x_d^{\text{ref}}\right)^2}, \tag{55}$$

where D denotes the total number of pixel values in the image.

E.2 Implementation Details for ParaTAA Transfer Experiments

Purpose. The ParaTAA [33] transfer experiment evaluates whether the proposed truncated initialization can be used as a plug-and-play module for another parallel sampling method. We keep the ParaTAA DDIM inference procedure fixed and only vary the initialization order q.

Model and hardware. We use Stable Diffusion 1.5 as the base model. The experiments are conducted on 8 NVIDIA GeForce RTX 4090 GPUs, each with 24 GB memory.

Sampling configuration. We use the official ParaTAA implementation[4] and incorporate our proposed initialization into its sampling procedure. The parameter settings are fixed across all compared initialization orders:

$$\begin{gathered}\texttt{timesteps} = 50, \quad \texttt{cfg_scale} = 4.0, \quad \texttt{eta} = 0.0, \quad \texttt{seed} = 0, \\ \texttt{num_samples} = 1, \quad \texttt{window_size} = 8, \quad \texttt{order} = 8, \quad \texttt{memory_size} = 2, \\ \texttt{tol1} = 0.001, \quad \texttt{tol2} = 0.001, \quad \texttt{fp16} = \texttt{False}. \end{gathered} \tag{56}$$

[3] https://huggingface.co/datasets/VincentGOURBIN/FluxPrompting

[4] https://github.com/TZW1998/ParaTAA-Diffusion

Table 6: Per-NPU memory usage of sequential and DDP-based parallel inference for 50-step sampling at 512×512 resolution.

Model	Sequential (MiB)	Parallel (MiB)
FLUX.1-dev	33796	33884
Stable Diffusion 3	18634	18654
Stable Diffusion 3.5	28542	28594

Table 7: CPU memory usage of sequential and DDP-based parallel inference. In the parallel setting, the total CPU memory is approximately the memory of one main process plus 8 worker subprocesses.

Model	Process	Sequential (MB)	Parallel (MB)
FLUX.1-dev	Main process	907.18	2.31
	Worker subprocess	–	993.46
Stable Diffusion 3	Main process	934.89	2.34
	Worker subprocess	–	983.93
Stable Diffusion 3.5	Main process	224.70	2.38
	Worker subprocess	–	264.95

Evaluation protocol. We compare $q = 0$, $q = 1$, and $q = 2$ using the same prompt dataset as the main paper. The ParaTAA transfer experiment is evaluated on $10{,}000$ prompts[5]. We report average inference time, average correction steps, ImageReward, and CLIP score in the main paper.

E.3 Computational Resource Usage

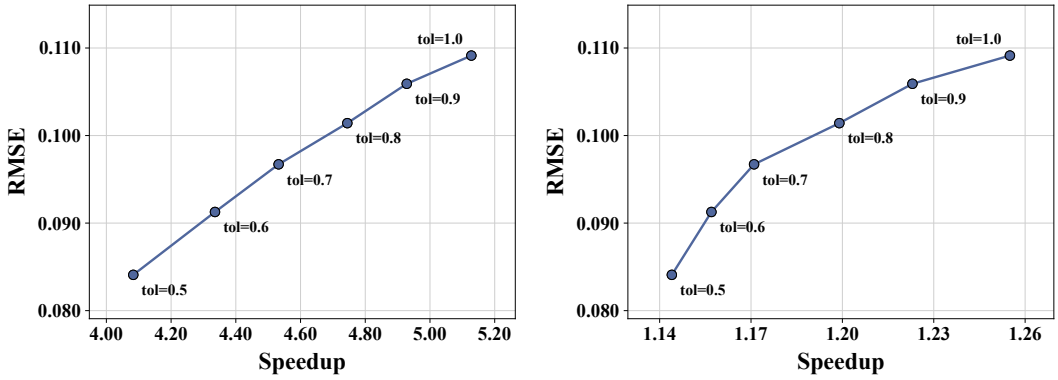
NPU memory usage. We implement ParaPC-FM using DDP-based parallel inference with window size $n = 8$. Under this implementation, ParaPC-FM launches 8 parallel worker processes, each corresponding to one NPU and keeping a separate model replica. As shown in Table 6, the per-NPU memory usage of ParaPC-FM is very close to that of the original sequential inference. For 50-step sampling at 512×512 resolution, the increase in per-worker NPU memory is negligible. Since the model is replicated across 8 NPU workers, the total NPU memory usage is approximately 8 times that of single-NPU sequential sampling.

CPU memory usage. For CPU memory, the parallel setting consists of one lightweight main process and 8 worker subprocesses. Therefore, the total number of CPU-side processes is $8 + 1$. As reported in Table 7, each worker subprocess has a CPU memory footprint comparable to the corresponding sequential process, while the main process is lightweight. As a result, the total CPU memory usage is approximately the memory of the main process plus 8 worker subprocesses.

CPU utilization. Compared with standard single-NPU sequential inference, DDP-based parallel inference uses 8 additional worker subprocesses plus one main process. Therefore, CPU utilization increases with the number of workers. Overall, the main resource overhead of ParaPC-FM comes from model and process replication across the 8 DDP workers, rather than a large increase in the per-worker memory footprint.

Estimated reproduction time. We estimate the total time required to reproduce the reported experiments based on the average per-image inference times reported in Section 4. Specifically, we sum the average inference time of each configuration multiplied by the corresponding number of evaluated prompts, and include the time required for auxiliary experiments and processing. This gives an estimated total reproduction time of approximately 5.82×10^5 seconds, i.e., about 161.6 hours or 6.7 days. The per-image average inference times used in this estimate are reported in the corresponding tables in Section 4.

[5] `https://huggingface.co/datasets/VincentGOURBIN/FluxPrompting`



(a) Overall speedup

(b) Initialization speedup

Figure 6: Speedup versus reference fidelity on FLUX.1-dev with 50 sampling steps. (a) Overall speedup relative to sequential sampling. (b) Initialization speedup relative to $q = 0$ at the same tolerance. Both panels use RMSE against the 50-step sequential Euler output (lower is better). Labels denote tolerance; lines connect observed means.

Table 8: Initialization baselines on FLUX.1-dev with 50 sampling steps, Euler correction ($p = 0$), and window size $n = 8$. Rounds denotes effective sampling/correction rounds. S is speedup relative to the original ParaDiGMS baseline. Estimated NFE is $8 \times \text{Rounds}$ for parallel rows.

Method	Rounds↓	Est. NFE↓	Time (s)↓	$S \uparrow$	ImRe↑	CLIP↑
Sequential	50.000	50.000	22.788	–	0.842	26.0
Original ParaDiGMS ($q = 0$)	15.243	121.944	6.385	1.00	0.846	26.1
ParaPC-FM ($q = 1$)	**13.282**	106.256	**5.581**	**1.14**	0.845	26.1
Finite-difference extrapolation	14.954	119.632	6.296	1.01	0.846	26.1
Least-squares extrapolation	14.552	116.416	6.115	1.04	0.842	26.0

F Speed–Fidelity Trade-off

Figure 6 visualizes the tolerance study in Table 2, with 50 sampling steps, $q = 1$, and the remaining settings fixed. RMSE uses the corresponding 50-step sequential Euler output with RGB values normalized to $[0, 1]$, rather than the 100-step reference in the main FLUX comparison. Panel (a) divides the fixed sequential runtime (22.788 s) by parallel runtime. Panel (b) uses the initialization speedup in Definition 4.1, with the $q = 0$ baseline matched separately at each tolerance.

Increasing tolerance from 0.5 to 1.0 raises overall speedup from $4.08\times$ to $5.13\times$, while RMSE rises from 0.084 to 0.109, showing the cost of faster sampling in reference fidelity. RMSE is not a percentage of perceptual degradation. Panel (b) isolates the incremental initialization gain; its changing baseline differs from the common speed scale in panel (a).

G Additional Experiments

We provide additional FLUX.1-dev experiments with 50 sampling steps on eight NPUs to examine initialization baselines and window size.

G.1 Comparison with Simple Extrapolation Baselines

Table 8 compares initialization strategies under matched sampling settings with $p = 0$ and $n = 8$. Here, $(p, q) = (0, 0)$ is the original ParaDiGMS baseline. Finite-difference extrapolation estimates local acceleration from the last two velocities and linearly extrapolates future velocities. Least-squares extrapolation fits a linear combination of these velocities using trajectories from 1,000 calibration prompts disjoint from evaluation prompts. Both reuse cached velocities without additional model evaluations at inference; our initializer also requires no offline fitting.



(a) Seq., $p = 0$ (50 steps) (b) $p = 0,\ q = 0$ (14 steps) (c) $p = 0,\ q = 1$ (12 steps) (d) $p = 0,\ q = 2$ (14 steps) (e) $p = 1,\ q = 0$ (15 steps)



(f) $p = 1,\ q = 1$ (13 steps) (g) $p = 1,\ q = 2$ (14 steps) (h) $p = 2,\ q = 0$ (14 steps) (i) $p = 2,\ q = 1$ (12 steps) (j) $p = 2,\ q = 2$ (14 steps)

Figure 7: All images are 512×512 samples generated by Stable Diffusion 3 model using the prompt *"Vintage metal lantern"*. The first image is generated sequentially with 50 steps, while the remaining nine images are produced using parallel sampling with different (p, q) update and initialization orders under $\mathrm{tol} = 1.0$. The numbers in parentheses indicate the number of sampling steps for each image.

Table 9: Window-size sensitivity on FLUX.1-dev with 50 sampling steps and $(p, q) = (0, 1)$ on eight NPUs. $S_{\mathrm{seq}} = 22.788/T$ is speedup relative to sequential sampling. Estimated NFE is $n \times \mathrm{Rounds}$ for parallel rows.

Method	Window n	Rounds↓	Est. NFE↓	Time (s)↓	S_{seq} ↑	ImRe↑	CLIP↑
Sequential	1	50.000	50.000	22.788	1.00	0.842	26.0
ParaPC-FM	4	22.439	89.756	9.586	2.38	0.845	26.1
ParaPC-FM	8	13.282	106.256	**5.581**	**4.08**	0.845	26.1
ParaPC-FM	16	**12.757**	204.112	10.199	2.23	0.851	26.2

ParaPC-FM requires fewer correction rounds and less wall-clock time than both extrapolation alternatives, with comparable reported ImRe and CLIP scores.

G.2 Sensitivity to Window Size

Table 9 fixes $(p, q) = (0, 1)$ and varies window size. Larger windows reduce correction rounds, but wall-clock time is not monotonic: $n = 4$ underutilizes the eight devices, while $n = 16$ requires additional scheduling/batches. The matching window size $n = 8$ gives the shortest runtime. Window size should therefore be selected jointly with available device parallelism.

H Additional Qualitative Comparisons of ParaPC-FM

Specifically, we present qualitative comparisons under fixed model and sampling-step settings: 50-step image generation on Stable Diffusion 3 (Figure 7), 50-step image generation on Stable Diffusion 3.5 (Figure 8), and image generation on FLUX.1-dev with 25, 50, and 100 sampling steps (Figures 9, 10, and 11), respectively. In each case, different (p, q) configurations are compared. We also provide qualitative results under different tolerance values in Figure 12, where (p, q) is fixed to $(1, 1)$.



(a) Seq., $p = 0$ (50 steps) (b) $p = 0,\ q = 0$ (16 steps) (c) $p = 0,\ q = 1$ (15 steps) (d) $p = 0,\ q = 2$ (16 steps) (e) $p = 1,\ q = 0$ (17 steps)



(f) $p = 1,\ q = 1$ (16 steps) (g) $p = 1,\ q = 2$ (17 steps) (h) $p = 2,\ q = 0$ (19 steps) (i) $p = 2,\ q = 1$ (18 steps) (j) $p = 2,\ q = 2$ (19 steps)

Figure 8: All images are 512×512 samples generated by Stable Diffusion 3.5 model using the prompt *"A vintage-style poster advertising a classic Route 66 motel or hotel, featuring a cheerful cartoon character."* The first image is generated sequentially with 50 steps, while the remaining nine images are produced using parallel sampling with different (p, q) update and initialization orders under $\mathrm{tol} = 1.0$. The numbers in parentheses indicate the number of sampling steps for each image. This example is selected to illustrate how correction order p affects adherence to the requested cartoon character. At fixed p, outputs remain broadly similar across q, which mainly affects convergence efficiency.

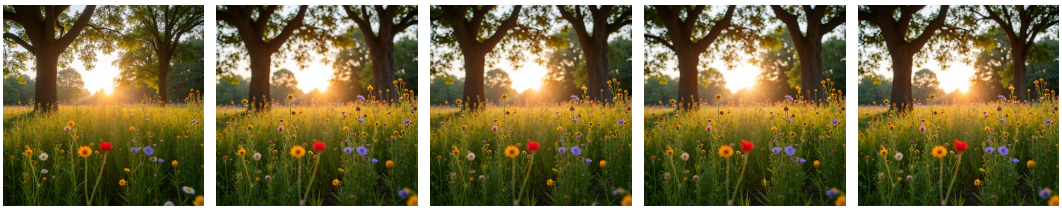


(a) Seq., $p = 0$ (25 steps) (b) $p = 0,\ q = 0$ (9 steps) (c) $p = 0,\ q = 1$ (9 steps) (d) $p = 0,\ q = 2$ (9 steps) (e) $p = 1,\ q = 0$ (9 steps)



(f) $p = 1,\ q = 1$ (8 steps) (g) $p = 1,\ q = 2$ (9 steps) (h) $p = 2,\ q = 0$ (10 steps) (i) $p = 2,\ q = 1$ (10 steps) (j) $p = 2,\ q = 2$ (10 steps)

Figure 9: All images are 512×512 samples generated by FLUX.1-dev using the prompt *"Simple breakfast scene with Muji kitchenware"*. The first image is generated sequentially with 25 steps, while the remaining nine images are produced using parallel sampling with different (p, q) update and initialization orders under $\mathrm{tol} = 1.0$. The numbers in parentheses indicate the number of sampling steps for each image. This example is selected to illustrate how correction order p affects scene details, including food and tableware. At fixed p, outputs remain broadly similar across q, which mainly affects convergence efficiency.



(a) Seq., $p = 0$ (50 steps) (b) $p = 0,\ q = 0$ (13 steps) (c) $p = 0,\ q = 1$ (9 steps) (d) $p = 0,\ q = 2$ (10 steps) (e) $p = 1,\ q = 0$ (13 steps)

(f) $p = 1,\ q = 1$ (10 steps) (g) $p = 1,\ q = 2$ (10 steps) (h) $p = 2,\ q = 0$ (13 steps) (i) $p = 2,\ q = 1$ (10 steps) (j) $p = 2,\ q = 2$ (10 steps)

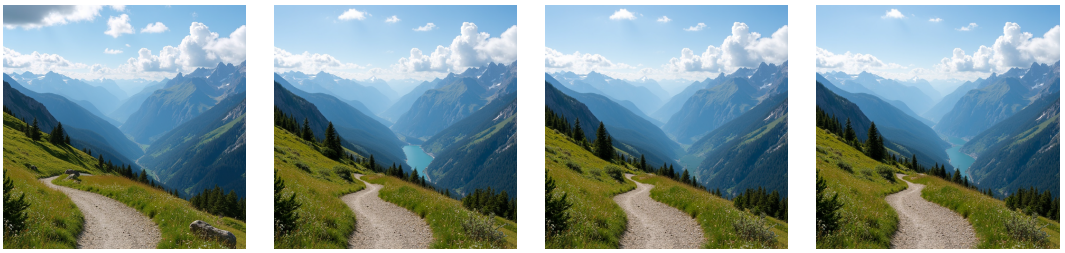
Figure 10: All images are 512×512 samples generated by FLUX.1-dev using the prompt *"A serene meadow at sunrise with wildflowers swaying gently in the breeze, surrounded by tall oak trees and a soft carpet of green grass."* The first image is generated sequentially with 50 steps, while the remaining nine images are produced using parallel sampling with different (p, q) update and initialization orders under $\mathrm{tol} = 1.0$. The numbers in parentheses indicate the number of sampling steps for each image.



(a) Seq., $p = 0$ (100 steps) (b) $p = 0,\ q = 0$ (24 steps) (c) $p = 0,\ q = 1$ (16 steps) (d) $p = 0,\ q = 2$ (18 steps) (e) $p = 1,\ q = 0$ (25 steps)

(f) $p = 1,\ q = 1$ (17 steps) (g) $p = 1,\ q = 2$ (20 steps) (h) $p = 2,\ q = 0$ (24 steps) (i) $p = 2,\ q = 1$ (17 steps) (j) $p = 2,\ q = 2$ (19 steps)

Figure 11: All images are 512×512 samples generated by FLUX.1-dev using the prompt *"An authentic Italian village market filled with fresh produce, local handicrafts, and the scent of freshly baked bread wafting through the air."*. The first image is generated sequentially with 100 steps, while the remaining nine images are produced using parallel sampling with different (p, q) update and initialization orders under $\mathrm{tol} = 1.0$. The numbers in parentheses indicate the number of sampling steps for each image.

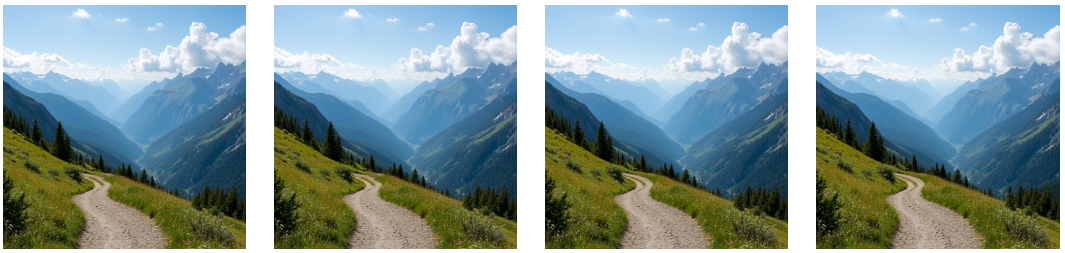


(a) Seq.
(50 steps)

(b) $\mathrm{tol} = 0.4$
(12 steps)

(c) $\mathrm{tol} = 0.5$
(11 steps)

(d) $\mathrm{tol} = 0.6$
(10 steps)



(e) $\mathrm{tol} = 0.7$
(10 steps)

(f) $\mathrm{tol} = 0.8$
(10 steps)

(g) $\mathrm{tol} = 0.9$
(10 steps)

(h) $\mathrm{tol} = 1.0$
(9 steps)

Figure 12: All images are 512×512 samples generated by FLUX.1-dev using the prompt *"A scenic ridge path with breathtaking views of the surrounding mountains."* The first image is generated sequentially, while the remaining seven images are produced using parallel sampling with $(p, q) = (1, 1)$ under different tolerance settings. The first line of each subfigure caption indicates the sequential baseline or the tolerance value, and the second line indicates the number of sampling steps.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work performed by the authors in Section 4.4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For each theoretical result, the paper provides the full set of assumptions and a complete (and correct) proof in Appendix C and Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All information necessary to reproduce the main experimental results relevant to the paper's claims and conclusions is provided in Section 4.1 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and accompanying documentation are available at `https://github.com/libangwei01/ParaPC-FM`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: the paper specifies all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results in Section 4.1 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: All reported results are averaged over multiple samples (either 1,000 or 10,000 prompts, depending on the experiment), rather than repeated random trials. Except for wall-clock time, which may have minor runtime fluctuations, the evaluations are conducted under identical settings and are reproducible in the same environment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: For each experiment, the paper provides sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments in Appendix E.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly credit the creators and original owners of all external assets used in this work, including code, datasets, and pretrained models, through citations and acknowledgements where appropriate. We also respect the corresponding licenses, terms of use, and usage restrictions.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code and accompanying documentation are available at `https://github.com/libangwei01/ParaPC-FM`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.