Appendix

Table of Contents

A Implementation

Our implementation is available here[1]. The implementation is based on Katherine Crowson's guided-diffusion [2], which is inspired by OpenAI's guided-diffusion[3]. All of the pre-trained diffusion and classifier models are available here[4]. For evaluation, we use OpenAI's measurement implementation with their reference image batch, which can be found here[5].

B Improving PLMS

Initial points are required for using high-order PLMS. The fourth-order formulation, for example, requires three initial points. The original paper (Liu et al., 2022) employs the Runge-Kutta method to compute the initial points. However, Runge-Kutta's method has high computational costs and

[1] https://github.com/sWizad/split-diffusion

[2] https://github.com/crowsonkb/guided-diffusion

[3] https://github.com/openai/guided-diffusion

[4] https://github.com/openai/guided-diffusion/blob/main/model-card.md

[5] https://github.com/openai/guided-diffusion/tree/main/evaluations

is inconvenient to use when the number of steps is small. To reduce the computation costs, we compute the starting points of the higher-order PLMS using lower-order PLMS. Our PLMS can be summarized using Algorithm 3.

Algorithm 3: PLMS

input: $\bar{x}_n$ (previous result), σ_{n+1}, σ_n,
$\{e_i\}_{i<n}$ (evaluation buffer), r (method order) ;
  $e_n = \bar{\epsilon}_\sigma(\bar{x}_n)$;
  $c = \min(r, n)$;
  if $c == 1$ **then**
    $\hat{e} = e_n$;
  else if $c == 2$ **then**
    $\hat{e} = (3e_n - e_{n-1})/2$;
  else if $c == 3$ **then**
    $\hat{e} = (23e_n - 16e_{n-1} + 5e_{n-2})/12$;
  else
    $\hat{e} = (55e_n - 59e_{n-1} + 37e_{n-3} - 9e_{n-4})/24$;
  Result: $\bar{x}_n + (\sigma_{n+1} - \sigma_n)\hat{e}$

In Algorithm 3, the PLMS formulation is obtained by assuming a constant $\Delta\sigma$ for each step. The results in our experiments and previous work (e.g., Liu et al. (2022); Zhang & Chen (2022)) are still reasonable when this assumption is not strictly satisfied, i.e., when $\Delta\sigma$ is not constant. Inspired by Zhang & Chen (2022), we also show how to derive another linear multi-step formulation for non-constant $\Delta\sigma$. Let us first define a dummy variable τ in which $\Delta\tau$ is a constant in each time step. To make it simple, let the value $\tau = 0$ when $t = T$ and when $t = 0$, $\tau = N$, the total number of steps. As a result, the discretization of τ can be defined by $\tau_n = n$ and $\Delta\tau = \tau_n - \tau_{n-1} = 1$ is a constant. Next, we want to extrapolate the value of ϵ_θ using a polynomial $P_\epsilon(\tau)$.

For example, the first order formulation can be produced by integrating a constant polynomial $P_\epsilon(\tau) = e_n$:

$$\begin{aligned}\frac{d\bar{x}}{d\sigma} = \bar{\epsilon}(\bar{x}) \approx & P_\epsilon(\tau) = e_n,\\ \int_{\bar{x}_n}^{\bar{x}_{n+1}} d\bar{x} = & \int_{\sigma_n}^{\sigma_{n+1}} e_n \, d\sigma,\\ \bar{x}_{n+1} - \bar{x}_n = & e_n \Delta\sigma\end{aligned}$$

which leads us to Euler's formulation. Rather than using Lagrange's polynomial like Zhang & Chen (2022), we use Newton's polynomial, which gives a nicer final formulation. However, both are the same polynomial but have different expressions. For the 2nd order formulation, we interpolate between (τ_n, e_n) and (τ_{n-1}, e_{n-1}) using a Newton's polynomial $P_\epsilon(\tau) = e_n + (e_n - e_{n-1})(\tau - \tau_n)$.

$$\int_{\bar{x}_n}^{\bar{x}_{n+1}} d\bar{x} = \int_{\sigma_n}^{\sigma_{n+1}} e_n + (e_n - e_{n-1})(\tau - \tau_n) d\sigma.$$

Every term is the same as in the 1st order formulation, except for one term that is $\int_{\sigma_n}^{\sigma_{n+1}} (\tau - \tau_n) d\sigma$. Let us use separable integration to approximate this term.

$$\int_{\sigma_n}^{\sigma_{n+1}} (\tau - \tau_n) d\sigma = \int_{\tau_n}^{\tau_{n+1}} (\tau - \tau_n) \frac{d\sigma}{d\tau} d\tau \approx \int_{\tau_n}^{\tau_{n+1}} (\tau - \tau_n) d\tau \int_{\tau_n}^{\tau_{n+1}} \frac{d\sigma}{d\tau} d\tau = \frac{\Delta\sigma}{2}$$

The result is $\bar{x}_{n+1} - \bar{x}_n = e_n \Delta\sigma + (e_n - e_{n+1}) \frac{\Delta\sigma}{2}$, which is the PLMS2 formulation. To compute this term more precisely, we need to know the derivation $d\sigma/d\tau$. For this example, let us define σ by

$$\sigma(\tau) = \exp\left(\ln \sigma_{\text{max}} + \frac{\tau}{N}(\ln \sigma_{\text{min}} - \ln \sigma_{\text{max}})\right) = \exp(a + \tau b), \tag{16}$$

where $a = \ln \sigma_{\text{max}}$ and $b = (\ln \sigma_{\text{min}} - \ln \sigma_{\text{max}})/N$. Now, we have

$$\int_{\tau_n}^{\tau_{n+1}} (\tau - \tau_n) \frac{d\sigma}{d\tau} d\tau = \Delta\sigma \left(\frac{\exp(b)}{\exp(b) - 1} - \frac{1}{b} \right).$$

Consider $\frac{\exp(b)}{\exp(b)-1} - \frac{1}{b}$ when limit $N \to \infty$ (or $b \to 0$), this term is equal to $\frac{1}{2}$, which turns the formulation back to PLMS2. When we set $N = 30$ (or $b = -0.33$), the term $\frac{\exp(b)}{\exp(b)-1} - \frac{1}{b} = 0.4723$, which is also close to $\frac{1}{2}$ in PLMS2 formulation.

We can continue using higher-order Newton's polynomials to obtain higher-order formulations. We call this method GLMS (Generalized Linear Multi-Step) and summarize it with Algorithm 4. In our comparison in Figure 6, both PLMS and GLMS produce comparable results. In the fourth order, GLMS4 performs slightly better than PLMS4. However, the GLMS formulation is dependent on the σ schedule, and the formulation must be revised if the σ schedule changes. As a result, we decided to use PLMS as part of our main algorithm for more flexibility.

Algorithm 4: GLMS

input: $\bar{x}_n$ (previous result), σ_{n+1}, σ_n, $\{e_i\}_{i<n}$ (evaluation buffer), r (method order) ;
$b = \ln(\sigma_{n+1}) - \ln(\sigma_n)$ $\quad e_n = \bar{\epsilon}_\sigma(\bar{x}_n)$; $\quad \{e_i\}$.append(e_n) ; $\quad c = \min(r, n)$;
if $c \geq 1$ **then**
$\hat{e} = e_n$;
if $c \geq 2$ **then**
$\hat{e} = \hat{e} + (e_n - e_{n-1}) \left(\frac{\exp(b)}{\exp(b)-1} - \frac{1}{b} \right)$;
if $c \geq 3$ **then**
$\hat{e} = \hat{e} + (e_n - 2e_{n-1} + e_{n-2}) \left(\frac{\exp(b)}{\exp(b)-1}(2 - \frac{2}{b}) - \frac{1}{b} + \frac{2}{b^2} \right)$;
if $c \geq 4$ **then**
$\hat{e} = \hat{e} + (e_n - 3e_{n-1} + 3e_{n-2} - e_{n-3}) \left(\frac{\exp(b)}{\exp(b)-1}(6 - \frac{9}{b} + \frac{6}{b^2}) - \frac{2}{b} + \frac{6}{b^2} - \frac{6}{b^3} \right)$;
Result: $\bar{x}_n + (\sigma_{n+1} - \sigma_n)\hat{e}$

C Numerical methods for unguided diffusion

This experiment evaluates the numerical methods used in our paper on *unguided* sampling to see whether they may behave differently. We perform a similar experiment as in Section 4.1, 4.2 except on an unguided, unconditional diffusion model and use samples from 1,000-step DDIM as reference images. For each method, we use the same initial noise maps as the DDIM and evaluate the image similarity between its generated images and the reference images based on LPIPS. Figure 6 reports LPIPS vs sampling time for ImageNet128 using the σ schedule in Equation 16.

We found that RK2 and RK4 perform better than DDIM, but in our main papers these methods perform worse than DDIM when used on the split sub-problems for guided sampling. Linear multi-step methods continue to be the best performers. The graph also shows that higher order is generally better, especially when the number of steps is large.

D Comparing PLMS and DEIS

In Tables 3 and 4, we compare the coefficients e_n, e_{n-1}, e_{n-2}, and e_{n-3} of PLMS in Algorithm 3 and the original implementation DEIS Zhang & Chen (2022) using the default linear schedule and 10 sampling steps. This comparison demonstrates that DEIS and PLMS are similar methods. However, there are some differences. DEIS uses non-fixed coefficients that can change depending on the number of steps and the noise schedule, whereas PLMS has fixed coefficients. Although the coefficients from both methods are close, it should be noted that the DEIS coefficients will converge to the PLMS coefficients over time. Since DEIS's implementation relies heavily on the noise schedule, if the schedule changes, we must re-implement the method. Therefore, for practical purposes, we prefer PLMS over DEIS.

E LPIPS vs. the number of sampling step

We report additional LPIPS results of the experiment in Section 4.1 but with respect to the number of sampling steps. The result shows that methods in the RK family can outperform other methods

	coefficient of			
n$^{\text{th}}$ step	e_n	e_{n-1}	e_{n-2}	e_{n-3}
1	1.0	0.	0.	0.
2	1.5	-0.5	0.	0.
3	1.5	-0.5	0.	0.
4	1.5	-0.5	0.	0.
5	1.5	-0.5	0.	0.
6	1.5	-0.5	0.	0.
7	1.5	-0.5	0.	0.
8	1.5	-0.5	0.	0.
9	1.5	-0.5	0.	0.
10	1.5	-0.5	0.	0.

(a) PLMS

	coefficient of			
n$^{\text{th}}$ step	e_n	e_{n-1}	e_{n-2}	e_{n-3}
1	1.00	0.	0.	0.
2	1.42	-0.42	0.	0.
3	1.43	-0.43	0.	0.
4	1.43	-0.43	0.	0.
5	1.44	-0.44	0.	0.
6	1.45	-0.45	0.	0.
7	1.46	-0.46	0.	0.
8	1.47	-0.47	0.	0.
9	1.48	-0.48	0.	0.
10	1.50	-0.50	0.	0.

(b) DEIS

Table 3: Comparison of second-order DEIS and PLMS coefficients

	coefficient of			
n$^{\text{th}}$ step	e_n	e_{n-1}	e_{n-2}	e_{n-3}
1	1.0	0.	0.	0.
2	1.5	-0.5	0.	0.
3	1.92	-1.33	0.41	0.
4	2.29	-2.46	1.54	-0.38
5	2.29	-2.46	1.54	-0.38
6	2.29	-2.46	1.54	-0.38
7	2.29	-2.46	1.54	-0.38
8	2.29	-2.46	1.54	-0.38
9	2.29	-2.46	1.54	-0.38
10	2.29	-2.46	1.54	-0.38

(a) PLMS

	coefficient of			
n$^{\text{th}}$ step	e_n	e_{n-1}	e_{n-2}	e_{n-3}
1	1.0	0.	0.	0.
2	1.42	-0.42	0.	0.
3	1.77	-1.12	0.34	0.
4	2.09	-2.07	1.28	-0.34
5	2.11	-2.11	1.31	-0.32
6	2.14	-2.16	1.35	-0.33
7	2.16	-2.22	1.38	-0.34
8	2.20	-2.28	1.42	-0.35
9	2.23	-2.35	1.47	-0.35
10	2.23	-2.28	1.55	-0.38

(b) DEIS

Table 4: Comparison of fourth-order DEIS and PLMS coefficients

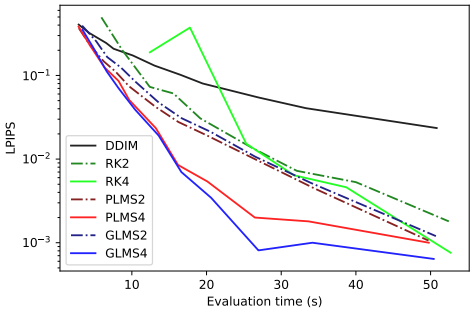
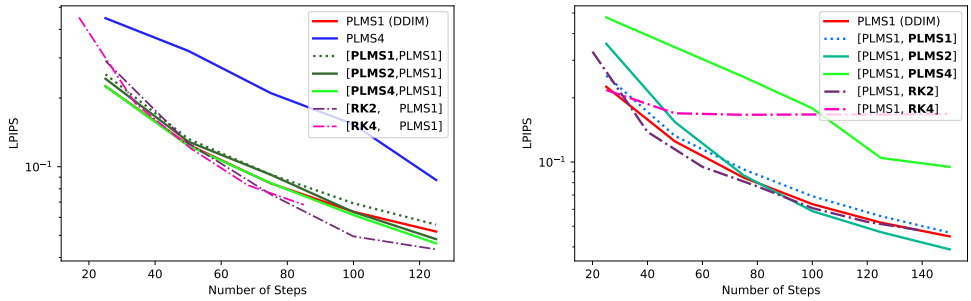


Figure 6: Comparison of different numerical methods on an unguided diffusion model. Using samples from a 1000-step DDIM as reference solutions, we measure average LPIPS scores and plot them against the average sampling time.



(a) Vary method for diffusion subproblem (b) Vary method for condition subproblem

Figure 7: In addition to Figure 4.1, we plot LPIPS against their **sampling steps**. RK family can outperform other methods in many situations. However, methods in RK family took a longer time per diffusion step.

	Sampling time within			
	5 sec.	10 sec.	15 sec.	20 sec.
DDIM	$0.111 \pm .078$	$0.062 \pm .065$	$0.042 \pm .056$	$0.031 \pm .048$
PLMS4	$0.240 \pm .131$	$0.085 \pm .112$	$0.039 \pm .072$	$0.018 \pm .040$
RK2	$0.152 \pm .090$	$0.048 \pm .051$	$0.030 \pm .037$	$0.026 \pm .037$
RK4	$0.190 \pm .106$	$0.044 \pm .022$	$0.033 \pm .038$	$0.019 \pm .032$
LTSP4	$0.111 \pm .092$	$0.056 \pm .060$	$0.040 \pm .054$	$0.028 \pm .046$
STSP4	$\mathbf{0.072} \pm .065$	$\mathbf{0.033} \pm .044$	$\mathbf{0.018} \pm .028$	$\mathbf{0.012} \pm .022$

Table 5: Our STSP4 has low average LPIPS scores and low standard deviations of LPIPS (N=120 samples).

given the same number of sampling steps. However, once taken into account the slower evaluation time per step, these methods are overall slower to reach the same level of LPIPS of other methods.

F More statistics for Experiment 4.2

We report the mean and standard deviation of each LPIPS score of the experiment in Section 4.2 in Table 5. In Table 6, we report the p-value for the null hypothesis that our STSP4 performs worse than other methods.

G STSP4 vs. additional numerical method combinations

We extend the experiment from Section 4.2 and compare our STSP4 and DDIM baselines with more numerical method combinations. Here we report LPIPS vs sampling time in Figure 8. The results

	Sampling time within			
	5 sec.	10 sec.	15 sec.	20 sec.
DDIM	9.1×10^{-11}	9.0×10^{-10}	7.1×10^{-9}	9.2×10^{-8}
PLMS4	3.3×10^{-41}	2.1×10^{-13}	8.1×10^{-4}	3.7×10^{-2}
RK2	1.2×10^{-30}	6.4×10^{-5}	6.1×10^{-5}	2.5×10^{-7}
RK4	5.0×10^{-31}	6.2×10^{-3}	6.1×10^{-5}	1.6×10^{-3}
LTSP4	5.6×10^{-11}	9.1×10^{-11}	2.8×10^{-10}	1.7×10^{-7}

Table 6: p-value for STSP4 has lower average LPIPS scores compare to other methods at approximately the same sampling time.

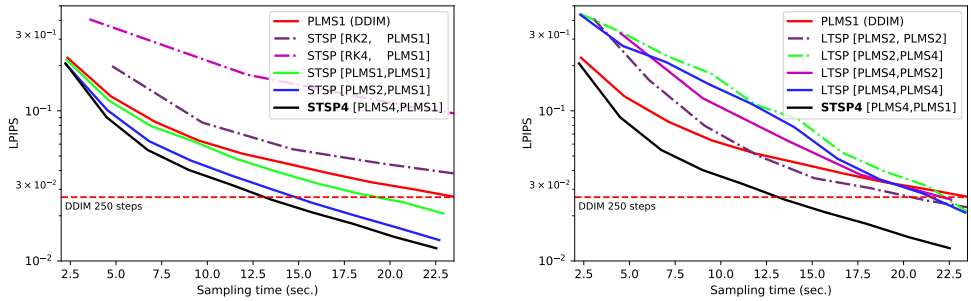


Figure 8: We compare Strang splitting with different numerical methods on the diffusion subproblem.

	Sampling time within			
	5 sec.	10 sec.	15 sec.	20 sec.
DPM-Solver-1 (DDIM)	0.333	0.125	0.080	0.045
DPM-Solver-2	0.565	0.188	0.078	0.045
DPM-Solver-3	0.540	0.233	0.087	0.043
LTSP4	0.185	0.105	0.071	0.048
STSP4	**0.169**	**0.062**	**0.061**	**0.037**

Table 7: The comparison with DPM-solver Lu et al. (2022). Average LPIPS when the sampling time is limited to be under 5-20 seconds.

show that Strang splitting method with [PLMS4, PLMS1] (our STSP4) is still the best combination, similar to the finding in Section 4.2.

H Comparison with DEIS and DPM-solver

We compare our splitting method to DPM-solver Lu et al. (2022) and DEIS Zhang & Chen (2022). Since both methods have different implementation details, such as using different noise schedules, which affect the ODE solution, these methods do not converge to the same exact solution. Thus, it is not sensible to directly compare them, and we instead implement our method in their official implementations of DPM-solver Lu et al. (2022) and DEIS Zhang & Chen (2022). We then compare the results using LPIPS on a classifier-guided diffusion model pretrained on ImageNet256, which is the same model used in Table 1 in Section 3. The results are shown in Table 7 and 8. Our splitting methods perform better than both DPM-solver Lu et al. (2022) and DEIS Zhang & Chen (2022).

	Sampling time within			
	3 sec.	6 sec.	9 sec.	12 sec.
0-DEIS (DDIM)	0.333	0.125	0.080	0.045
1-DEIS	0.466	0.193	0.092	0.044
3-DEIS	0.625	0.511	0.433	0.345
LTSP4	0.321	0.120	0.080	0.048
STSP4	**0.212**	**0.080**	**0.046**	**0.031**

Table 8: Comparison with DEIS Zhang & Chen (2022). We report the average LPIPS scores when the sampling time is limited to be under 3-12 seconds.

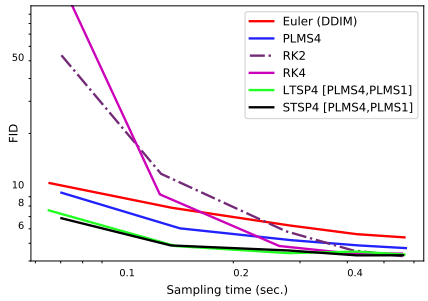


Figure 9: FID vs. Sampling time measured on ImageNet128

I EXPERIMENT ON FID VS. SAMPLING TIME

In this section, we demonstrate how splitting methods can accelerate guided diffusion sampling. We vary the number of sampling steps, generate 20k samples, and compare the FID scores of the splitting methods (LTSP4 and STSP4) to many non-splitting methods, including DDIM, PLMS4, RK2, and RK4. Figure 9 shows a comparison of FID vs. sampling time for each method. This experiment uses a classifier-guided diffusion model that was pretrained on the ImageNet128 dataset by Dhariwal & Nichol (2021). Our method can generate good sample quality, especially when the average sampling is limited to under 0.3 seconds (or about 50 steps of DDIM for 128×128 resolution), while other methods show large FID scores at lower sampling steps or require more time to generate similar high-quality samples.

J TEXT-GUIDED IMAGE GENERATION

For text-to-image generation or text-guided image generation, our implementation is based on Disco-Diffusion v3.1[6], which also relies on Crowson (2021). We use v3.1 that does not contain unrelated features; however, our method can be applied to any of the versions. We use the fine-tuned 512x512 diffusion model from Katherine Crowson [7] and use pre-trained OpenAI's CLIP models [8] including RN50, ViTB32, and ViTB16. For other model and conditional function configurations, we use Disco-Diffusion v3.1's default settings.

K CONTROLLABLE GENERATION

We provide implementation details of our sampling algorithm for other conditional generation tasks.

Image inpainting: Given a target masked image, y_0, and a mask P which is a matrix of $0, 1$ values, we want to sample x_0 so that $y_0 = Px_0$. The key concept from Song et al. (2020b) is to use reverse diffusion in the unmasked area and forward diffusion in the masked area through an additional step called the impose step. Let x'_{t-1} be an unconditional diffusion sample from x_t and y_{t-1} be a forward diffusion sample from y_0. The impose step can be summarized as follows:

$$x_{t-1} = (I - P^T P)x'_{t-1} + P^T y_{t-1}. \tag{17}$$

To sample a corrupted target image y_{t-1}, we use the sampling formulation from Song et al. (2020a) to sample from y_0 by

$$y_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}y_0 + \sqrt{1 - \bar{\alpha}_t - \eta_t^2}\epsilon_\theta(x_t, t), \eta_t^2 I) \tag{18}$$

where $\eta_t = \sqrt{(1 - \bar{\alpha}_{t-1})/(1 - \bar{\alpha}_t)}\sqrt{1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}}$. This formulation works more effectively with numerical methods than the original sampling formulation from Song et al. (2020b), which is $y_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}y_0, (1 - \bar{\alpha}_t)I)$.

[6] https://colab.research.google.com/drive/1bItz4NdhAPHg5-u87KcH-MmJZjK-XqHN

[7] http://batbot.tv/ai/models/guided-diffusion/512x512_diffusion_uncond_finetune_008100.pt

[8] https://github.com/openai/CLIP

Rather than sampling x'_{t-1} unconditionally, we follow Chung et al. (2022a) and use guided diffusion sampling with the conditional function defined by:

$$f(x_t) = \frac{1}{2\gamma}||y_0 - P\hat{x}_0(x_t)||_2^2, \quad \hat{x}_0(x_t) = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \tag{19}$$

where γ is a control parameter. Since directly computing $\epsilon_\theta(x_t, t)$ with the diffusion network in Equation 18 and 19 is time consuming, we use a secondary-model method from Katherine Crowson[9] to speed it up.

Image colorization: The idea behind colorization is very similar to inpainting. We convert images from the RGB format to the HSV format, with the grayscale value in the first channel, using an orthogonal matrix:

$$C = \begin{bmatrix} 0.577 & -0.816 & 0 \\ 0.577 & 0.408 & 0.707 \\ 0.577 & 0.408 & -0.707 \end{bmatrix}.$$

To mask out other channels and keep only the first grayscale channel, we define the mask matrix P as 1 in the grayscale channel and 0 in the other two channels. The impose step and conditional function can be defined as

$$x_{t-1} = (I - C^T P^T PC)x'_{t-1} + C^T P^T PCy_{t-1}, \quad f(x_t) = \frac{1}{2\gamma}||y_0 - PC\hat{x}_0(x_t)||^2.$$

Image super-resolution: Let us denote D as a down-sampling matrix. The impose step and conditional function is given by

$$x_{t-1} = (I - D^T D)x'_{t-1} + D^T y_{t-1}, \quad f(x_t) = \frac{1}{2\gamma}||y_0 - D\hat{x}_0(x_t)||^2.$$

In our implementation, we replace the D and D^T operations with the ILVR (Choi et al., 2021) down-sampling and up-sampling functions.

Figure 10, 11, and 12 show additional qualitative results on the three tasks. We evaluate LPIPS and PSNR between the generated images and their original images for the three tasks in Table 9. Our test set consists of 200 input images, and each test image will be used to produce 6 samples for each task for the evaluation.

Note that LPIPS and PSNR are not the ideal measurements for this evaluation, but they are a de facto standard for benchmarking. Our goal here is to demonstrate how our technique can generalize to other conditional generation tasks. It is not appropriate to compare the acceleration effect of our diffusion technique with non-diffusion techniques since they are based on fundamentally different principles. However, for a comparison of the quality of diffusion and non-diffusion techniques for these tasks, refer to Chung et al. (2022a).

To achieve state-of-the-art performance on these tasks in terms of quality and speed, other recent techniques may be more suitable, such as improved initialization (Chung et al., 2022b) and forward-backward repetition (Meng et al., 2021). However, our contributions are orthogonal to these investigations.

	Inpainting		Colorization		Super-resolution	
Methods	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR
DDIM	0.17	19.52	0.28	20.40	0.46	17.88
PLMS4	0.25	15.01	0.47	13.50	0.73	7.85
LTSP4	0.17	19.61	0.31	19.40	0.51	16.07
STSP4	**0.16**	**20.03**	**0.26**	**21.27**	**0.42**	**19.34**

Table 9: Average LPIPS and PSNR scores from different methods on three different tasks.

[9] https://colab.research.google.com/drive/1mpkrhOjoyzPeSWy2r7T8EYRaU7amYOOi

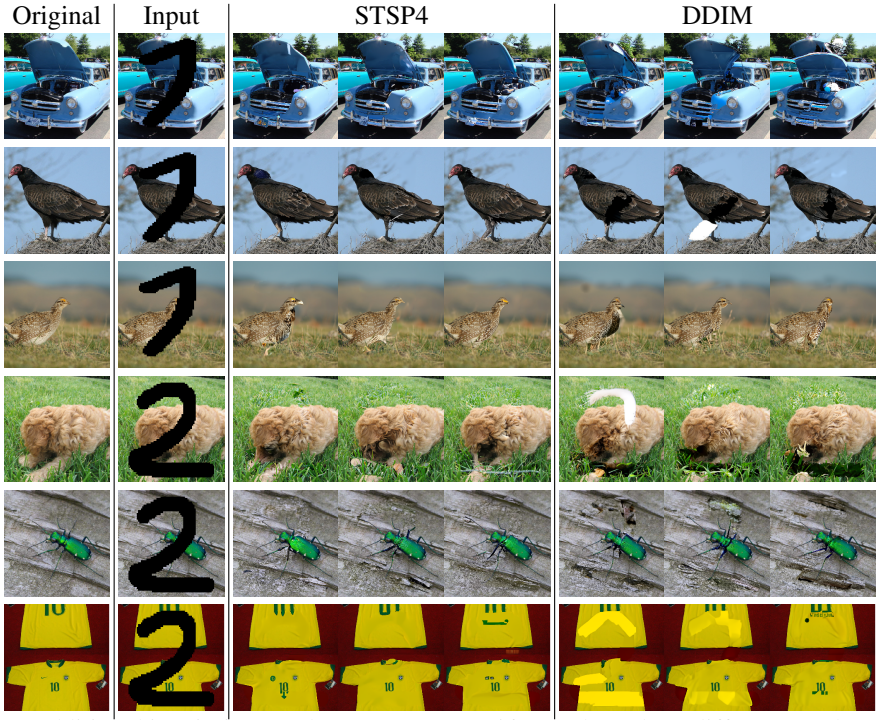


Figure 10: Additional inpainting results on ImageNet256. We show three different results generated by each method for each input. All results are generated using the same 5-second sampling time.

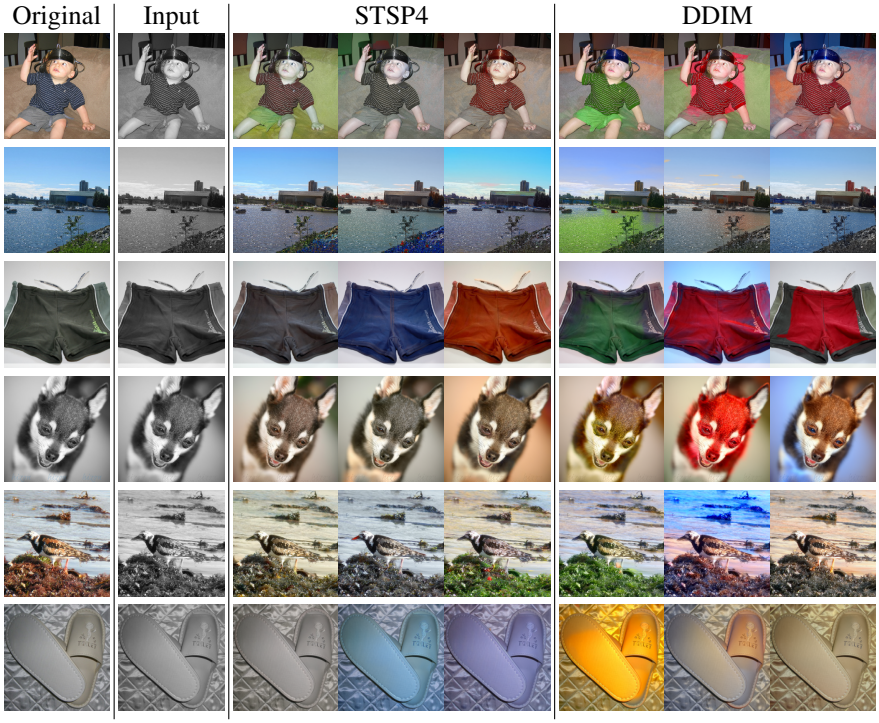


Figure 11: Additional colorization results on ImageNet256. We show three different results generated by each method for each input. All results are generated using the same 5-second sampling time.

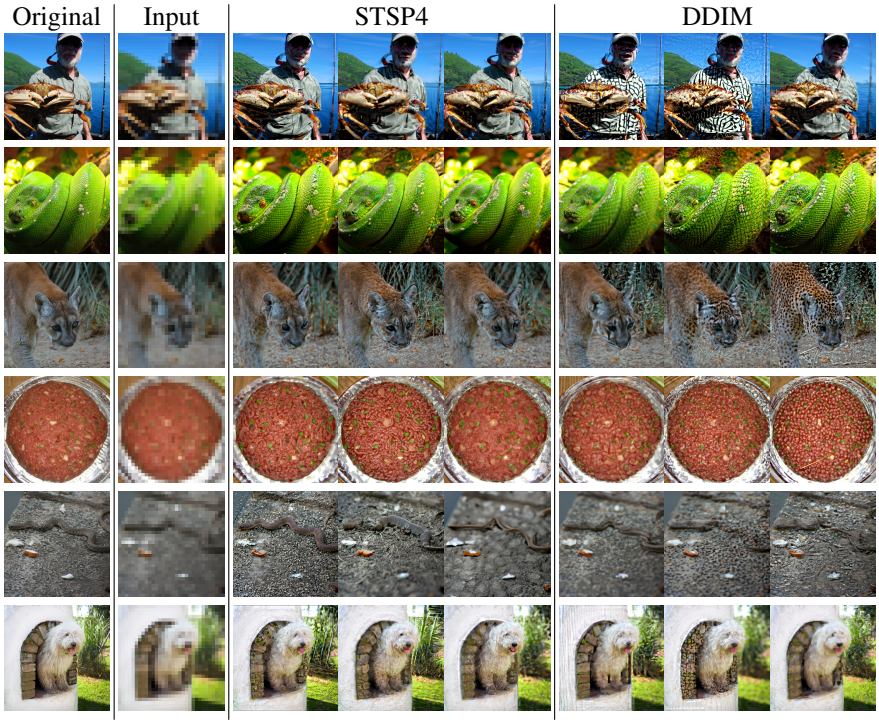


Figure 12: Additional 8x super-resolution results on ImageNet256. We show three different results generated by each method for each input. All results are generated using the same 5-second sampling time.

L DREAMBOOTH STABLE DIFFUSION

Dreambooth (Ruiz et al., 2022) is a technique for fine-tuning a pretrained text-to-image diffusion model on a given set of images. We discover that, similar to guided diffusion models, Dreambooth on Stable Diffusion (a pretrained text-guided latent-space diffusion) sometimes cannot be used with high-order methods but can be accelerated by our proposed method. This example demonstrates that our splitting method is effective not only on classifier-guided models but also classifier-free diffusion models.

The guided ODE of a classifier-free model is given by

$$\frac{d\bar{x}}{d\sigma} = \bar{\epsilon}_\sigma(\bar{x}|\phi) + s(\bar{\epsilon}_\sigma(\bar{x}|c) - \bar{\epsilon}_\sigma(\bar{x}|\phi)), \tag{20}$$

where c is the input prompt, $\bar{\epsilon}_\sigma(\bar{x}|c)$ is the network output conditioned on the input prompt, and $\bar{\epsilon}_\sigma(\bar{x}|\phi)$ is the network output conditioned on a null label ϕ. We can split the guided ODE into two subproblems as follows:

$$\frac{dy}{d\sigma} = \bar{\epsilon}_\sigma(y|\phi), \quad \frac{dz}{d\sigma} = s(\bar{\epsilon}_\sigma(z|c) - \bar{\epsilon}_\sigma(z|\phi)). \tag{21}$$

We test on "mo-di-diffusion[10]," a Dreambooth Stable Diffusion model that was fine-tuned on a dataset of screenshots from Disney studio. We use the prompt "a girl face in modern Disney style." The result is shown in Figure 13. However, we believe that the underlying problems of Dreambooth diffusion models may differ from those of classifier-guided diffusion models, and therefore require their own comprehensive study.

M CLIP-GUIDED STABLE DIFFUSION

UPainting (Li et al., 2022) suggests that incorporating gradients from CLIP models can improve the quality of text-to-image Stable Diffusion results. This approach is an example of combining

[10]https://huggingface.co/nitrosocke/mo-di-diffusion

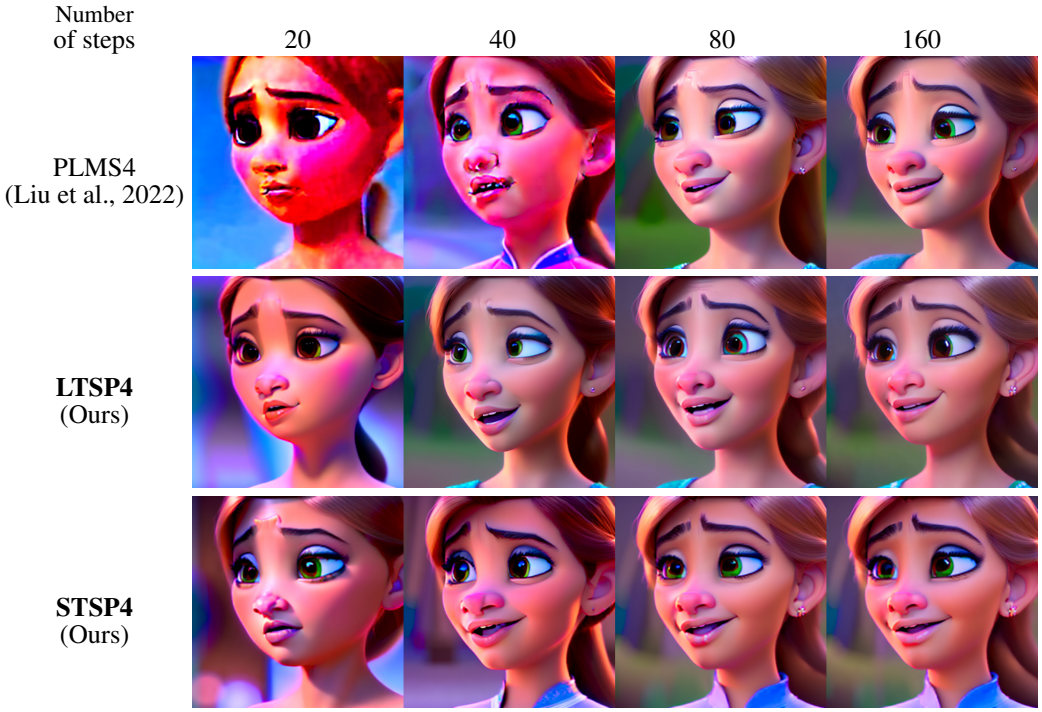


Figure 13: Generated samples from a text-guided Stable Diffusion model fine-tuned on a dataset of screenshots from Disney studio using 20-160 sampling steps. Our splitting technique produces high-quality results in fewer sampling steps. Prompt: "a girl face in modern Disney style."

classifier-free and gradient-guided techniques to achieve better outcomes. The CLIP-guided ODE 8 can be defined as follows:

$$\frac{d\bar{x}}{d\sigma} = \bar{\epsilon}_\sigma(\bar{x}|\phi) + s(\bar{\epsilon}_\sigma(\bar{x}|c) - \bar{\epsilon}_\sigma(\bar{x}|\phi)) - \lambda\nabla_{\bar{x}}(f_{\text{img}}(\bar{x}) \cdot f_{\text{txt}}(a)), \tag{22}$$

where $f_{\text{img}}(\bar{x})$ represents the output from CLIP's image encoder and $f_{\text{txt}}(a)$ represents the output of CLIP's text encoder. As we have shown in our paper, adding a gradient term to the diffusion model can hinder acceleration by high-order methods. To address this problem, we apply our methods and split Equation 22 into two subproblems, given by

$$\frac{dy}{d\sigma} = \bar{\epsilon}_\sigma(y|\phi) + s(\bar{\epsilon}_\sigma(y|c) - \bar{\epsilon}_\sigma(y|\phi)), \quad \frac{dz}{d\sigma} = -\lambda\nabla_z(f_{\text{img}}(z) \cdot f_{\text{txt}}(a)). \tag{23}$$

We illustrate sample images generated using different numerical methods in Figure 14. Our proposed method produces high-quality results in fewer sampling steps.

N Convergence orders of numerical methods

In this section, we establish the convergence orders of Lie-Trotter and Strang splitting methods for solving the differential equation

$$\frac{dx}{dt} = f_0(x) = f_1(x) + f_2(x), \tag{24}$$

where f_1 and f_2 are differentiable functions.

We define $\Phi_{\Delta t, f_i}$ as the mapping solution of the ODE $\frac{dx}{dt} = f_i(x,t)$ in the interval $[t_0, t_0 + \Delta t]$. Note that $\Phi_{0,f_i}(x) = x$ and $\frac{d}{dt}\Phi_{\Delta t,f_i}(x) = f_i(x)$.

Note that by performing a Taylor expansion, we can express the mapping solution $\Phi_{\Delta t,f_i}$ as

$$\Phi_{\Delta t,f_i}(x) = x + \Delta t f_i(x) + \frac{(\Delta t)^2}{2} f_i'(x) f_i(x) + \mathcal{O}(\Delta t^3). \tag{25}$$

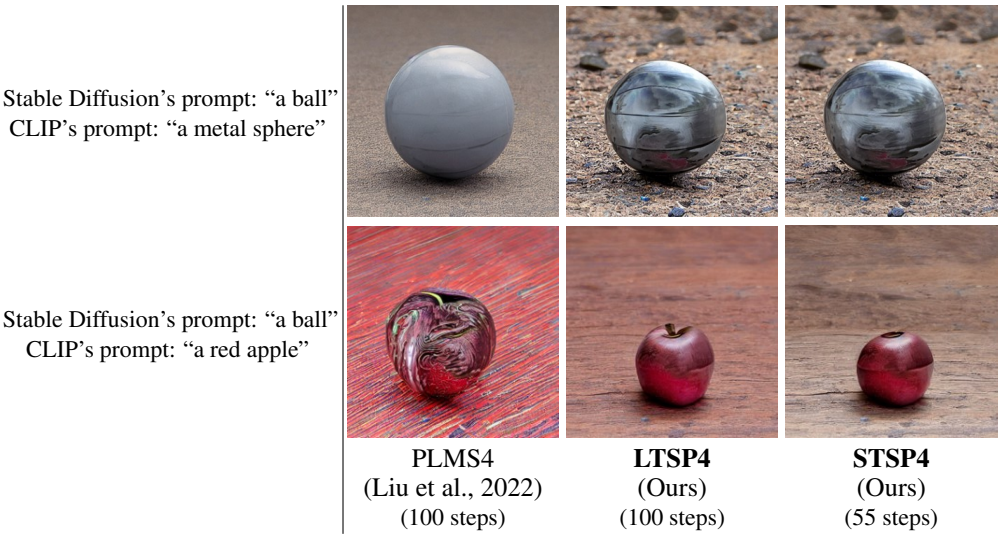


Figure 14: Text-to-image generation using CLIP-guided Stable Diffusion from different high-order sampling methods with approximately the same sampling time.

We will use this expansion in the proofs of the convergence orders for both Lie-Trotter and Strang splitting methods.

Proposition 1. *The Lie-Trotter splitting method converges at a first-order rate.*

Proof. We can express a single step of the Lie-Trotter splitting method as $\Phi_{\Delta t,f_2}(\Phi_{\Delta t,f_1}(x))$. By applying the expansion of Equation 25, we get:

$$\begin{aligned}\Phi_{\Delta t,f_2}(\Phi_{\Delta t,f_1}(x)) &= [x+\Delta t f_1(x)+\mathcal{O}(\Delta t^2)]+\Delta t f_2[x+\Delta t f_1(x)+\mathcal{O}(\Delta t^2)]+\mathcal{O}(\Delta t^2)\\ &= x+\Delta t f_1(x)+\Delta t f_2(x)+\mathcal{O}(\Delta t^2)\\ &= \Phi_{\Delta t,f_0}(x)+\mathcal{O}(\Delta t^2). \end{aligned} \tag{26}$$

This shows that the error in each step is of order $\mathcal{O}(\Delta t^2)$, which is a first-order convergence rate. □

Proposition 2. *The Strang splitting method has a convergence rate of second-order.*

Proof. Consider a single step of the Strang splitting, which is $\Phi_{\Delta t/2,f_2}(\Phi_{\Delta t,f_1}(\Phi_{\Delta t/2,f_2}(x)))$. To show the second-order accuracy of the Strang splitting, we first expand the two inner operators. Using the expansion of Equation 25, we have

$$\begin{aligned}\Phi_{\Delta t,f_1}(\Phi_{\Delta t/2,f_2}(x)) &= x+\frac{\Delta t}{2}f_2(x)+\frac{(\Delta t)^2}{2^2 2!}f'2(x)f_2(x)+\mathcal{O}(\Delta t^3)\\ &\quad+(\Delta t)f_1[x+\frac{\Delta t}{2}f_2(x)+\mathcal{O}(\Delta t^2)]\\ &\quad+\frac{(\Delta t)^2}{2}f'1[x+\mathcal{O}(\Delta t)]f_1[x+\mathcal{O}(\Delta t)]+\mathcal{O}(\Delta t^3) \qquad (27)\\ &= x+\frac{\Delta t}{2}f_2(x)+\frac{(\Delta t)^2}{2^2 2!}f'2(x)f_2(x)\\ &\quad+(\Delta t)f_1(x)+\frac{(\Delta t)^2}{2}f'1(x)f_2(x)\\ &\quad+\frac{(\Delta t)^2}{2}f'1(x)f_1(x)+\mathcal{O}(\Delta t^3) \qquad (28)\end{aligned}$$

Then, we apply the same expansion to the outer operator $\Phi_{\frac{\Delta t}{2},f_2}$:

$$\begin{aligned}\Phi_{\frac{\Delta t}{2},f_2}(\Phi_{\Delta t,f_1}(\Phi_{\frac{\Delta t}{2},f_2}(x))) = x &+ (\Delta t)f_1(x) + \frac{\Delta t}{2}f_2(x) \\ &+ \frac{(\Delta t)^2}{2^2 2!}f_2'(x)f_2(x) + \frac{(\Delta t)^2}{2}f_1'(x)f_2(x) + \frac{(\Delta t)^2}{2!}f_1'(x)f_1(x) \\ &+ \frac{\Delta t}{2}f_2[x + (\Delta t)f_1(x) + \frac{\Delta t}{2}f_2(x) + \mathcal{O}(\Delta t^2)] \\ &+ \frac{(\Delta t)^2}{2^2 2!}f_2'[x + \mathcal{O}(\Delta t)]f_2[x + \mathcal{O}(\Delta t)] + \mathcal{O}(\Delta t^3) \end{aligned} \tag{29}$$

$$\begin{aligned}\Phi_{\frac{\Delta t}{2},f_2}(\Phi_{\Delta t,f_1}(\Phi_{\frac{\Delta t}{2},f_2}(x))) &= x + (\Delta t)f_1(x) + (\Delta t)f_2(x) \\ &+ \frac{(\Delta t)^2}{2!}[f_1'(x)f_1(x) + f_1'(x)f_2(x) + f_2'(x)f_1(x) + f_2'(x)f_2(x)] \\ &+ \mathcal{O}(\Delta t^3) \end{aligned} \tag{30}$$

$$= x + (\Delta t)f_0(x) + \frac{(\Delta t)^2}{2!}f_0'(x)f_0(x) + \mathcal{O}(\Delta t^3) \tag{31}$$

$$= \Phi_{\Delta t,f_0}(x) + \mathcal{O}(\Delta t^3) \tag{32}$$

Therefore, the Strang splitting method's convergence rate is of second-order.

□

O TOY EXAMPLE WHERE HIGH-ORDER METHODS BECOME UNSTABLE

In this section, we create a toy example to demonstrate how high-order numerical methods can become unstable on a certain class of ODE problems (Stiff equation) despite involving no neural networks. Let us define the following ODE:

$$\frac{dx}{dt} = \epsilon(x) + s \cdot g(x), \qquad x(0) = \begin{bmatrix}1\\0\end{bmatrix}, \tag{33}$$

where s is a scaling parameter and

$$\epsilon\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \begin{bmatrix}0 & 1\\-1 & -2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}, \quad g\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \begin{bmatrix}0 & 0\\-1 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}. \tag{34}$$

Figure 15 depicts solution trajectories of various numerical methods. We can observe that the PLMS4's trajectory situates farther from the exact solution than the Euler's method or the splitting methods, LSTP4 and STSP4. The exact solution of Equation 33 is

$$x(t) = \frac{1}{s}\begin{bmatrix}-1\\s+1\end{bmatrix}e^{-(s+1)t} + \frac{1}{s}\begin{bmatrix}s+1\\-s-1\end{bmatrix}e^{-t}. \tag{35}$$

When s increases, the term $e^{-(s+1)t}$ decays to zero more rapidly than the term e^{-t}. When the two terms behave differently, classical high-order numerical methods tend to perform poorly unless they employ a very small step size.

P STABILITY ANALYSIS

In this section, we analyze the stability of numerical methods by computing the lowest number of steps before their numerical solutions are guaranteed to diverge by theory. We visualize the solution

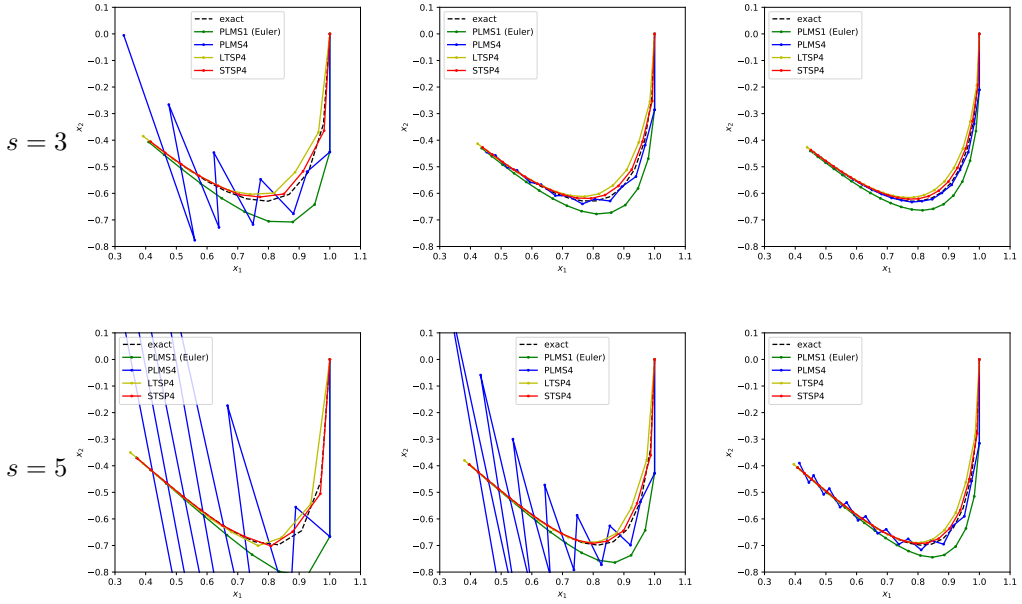


Figure 15: Solution trajectories of different numerical methods on a toy ODE problem using different numbers of steps. Non-splitting methods, especially PLMS4, are more likely to fail to converge to the exact solution when the number of steps is reduced.

trajectories using different numbers of steps to empirically support the theory. We only focus on Euler and other second-order numerical methods in this part.

One way to analyze numerical methods that solve Equation 33 is to evaluate them with a test equation that leads to the solution $y(t) = e^{-(s+1)t}$, such as

$$y' = -(s+1)y. \tag{36}$$

Note that the solution $y(t) \to 0$ as $t \to \infty$.

Euler's Method: Applying the Euler's method to Equation 36 yields:

$$y_{n+1} = y_n - \Delta t(s+1)y_n = (1 - \Delta t(s+1))y_n.$$

After solving this recurrence relation, we have $y_n = (1 - \Delta t(s+1))^n y_0$. The condition for the numerical solution $y_n \to 0$ as $n \to \infty$ is equivalent to $|1 - \Delta t(s+1)| < 1$ or

$$\begin{aligned} -1 < 1 - \Delta t(s+1) < 1, \\ 2 > \Delta t(s+1) > 0. \end{aligned}$$

Let us substitute $\Delta t = 1/N$, where N is the number of steps. Now, we can conclude that if N is lower than $\frac{s+1}{2}$, the solution of Euler's method in Equation 36 diverges from the exact solution.

PLMS2: Consider a second-order linear multistep method on the same test Equation 36:

$$y_{n+1} = y_n + \Delta t \left(-\frac{3}{2}(s+1)y_n + \frac{1}{2}y_{n-1}(s+1) \right) \tag{37}$$

$$= \left(1 - \Delta t \frac{3}{2}(s+1) \right) y_n + \Delta t \frac{1}{2}(s+1). \tag{38}$$

After solving the linear recurrence relation, we obtain

$$y_n = a_1 r_1^n + a_2 r_2^n, \tag{39}$$

$$\text{where } r_1 = \frac{1}{2}\left(1 - \frac{3}{2}\Delta t(s+1) + \sqrt{1 - \Delta t(s+1) + \frac{9}{4}(\Delta t)^2(s+1)^2}\right), \tag{40}$$

$$\text{and } r_2 = \frac{1}{2}\left(1 - \frac{3}{2}\Delta t(s+1) - \sqrt{1 - \Delta t(s+1) + \frac{9}{4}(\Delta t)^2(s+1)^2}\right). \tag{41}$$

The numerical solution $y_n \to 0$ as $n \to \infty$ when both $|r_1| < 1$ and $|r_2| < 1$, which is equivalent to

$$\left|\frac{1}{2}\left(1 - \frac{3}{2}\frac{(s+1)}{N} \pm \sqrt{1 - \frac{(s+1)}{N} + \frac{9}{4}\left(\frac{(s+1)}{N}\right)^2}\right)\right| < 1. \tag{42}$$

In Table 10, we report the lowest number N for each s before Inequality 42 is not satisfied. In other words, if the number of steps is below the lowest number N in the table, the solution of the method in Equation 36 is guaranteed to diverge from the exact solution. The analysis of the higher-order methods can be done in a similar fashion.

LTSP2: We analyze the Lie-Trotter splitting method similarly. In this case, the test Equation 36 needs to also be split into

$$\hat{y}' = -\hat{y}, \tag{43}$$

$$\tilde{y}' = -s\tilde{y}. \tag{44}$$

Let us apply the second order linear multistep method (PLMS2) to Equaiton 43 and Euler's method (PLMS1) to Equation 44. We have

$$\hat{y}_{n+1} = \hat{y}_n - \Delta t\left(\frac{3}{2}\hat{y}_n - \frac{1}{2}\hat{y}_{n-1}\right), \qquad \tilde{y}_{n+1} = \tilde{y}_n - \Delta t s\tilde{y}_n. \tag{45}$$

Thus, a single combining step of LTSP2 can be formulated by

$$y_{n+1} = (1 - s\Delta t)\left(\left(1 - \frac{3}{2}\Delta t\right)y_n + \frac{\Delta t}{2}y_{n-1}\right). \tag{46}$$

Similarly to the above, we solve the linear recurrence relation and obtain the following condition

$$\left|\frac{1}{2}\left(\left(1 - \frac{s}{N}\right)\left(1 - \frac{3}{2}\frac{s}{N}\right) \pm \sqrt{\left(1 - \frac{s}{N}\right)^2\left(1 - \frac{3}{2}\frac{s}{N}\right)^2 + \frac{2}{N}\left(1 - \frac{s}{N}\right)}\right)\right| < 1. \tag{47}$$

We report the lowest integer number N for each s before Inequality 47 is not satisfied in Table 10.

STSP2: We analyze the Strang splitting method by splitting the test Equation 36 into

$$\bar{y}' = -s\bar{y} \tag{48}$$

$$\hat{y}' = -\hat{y} \tag{49}$$

$$\tilde{y}' = -s\tilde{y} \tag{50}$$

We apply the second-order linear multistep method (PLMS2) on Equation 49 and Euler's method on Equation 48 and 50.

$$\bar{y}_{n+1} = \left(1 - \frac{\Delta t}{2}s\right)\bar{y}_n \tag{51}$$

$$\hat{y}_{n+1} = \left(1 - \frac{3}{2}\Delta t\right)\hat{y}_n + \frac{\Delta t}{2}\hat{y}_{n-1} \tag{52}$$

$$\tilde{y}_{n+1} = \left(1 - \frac{\Delta t}{2}s\right)\tilde{y}_n \tag{53}$$

	$s = 5$	$s = 10$	$s = 15$	$s = 20$	$s = 30$	$s = 40$	$s = 60$	$s = 80$
Euler	4	6	9	11	16	21	31	41
PLMS2	6	11	16	22	32	42	63	83
LTSP2	2	3	7	9	14	19	29	39
STSP2	2	3	4	5	8	10	15	20

Table 10: The lowest number of steps before each numerical method will fail to solve Equation 36. Notice that LTSP2 and STSP2 have lower numbers, which indicate that they are less likely to fail when the number of steps is reduced, as compared to Euler and PLMS2.

We combine Equation 51-53 into

$$y_{n+1} = \left(1 - \frac{s}{2N}\right)^2 \left(1 - \frac{3}{2N}\right) y_n + \frac{1}{2N}\left(1 - \frac{s}{2N}\right)^2 y_{n-1}. \tag{54}$$

After solving the linear recurrence relation, we obtain the following condition

$$\left|\frac{1}{2}\left(b \pm \sqrt{b^2 + \frac{2}{N}c}\right)\right| < 1, \tag{55}$$

where $b = \left(1 - \frac{s}{2N}\right)^2 \left(1 - \frac{3}{2N}\right)$ and $c = \left(1 - \frac{s}{2N}\right)^2$. In Table 10, we report the lowest number of steps N for each s before Inequality 55 is not satisfied.

In Table 10, we compare the lowest number of steps N before each method is guaranteed to diverge from our analysis. We also show numerical solutions of our toy example in Figure 16 and compare them with our analysis. It is important to note that if the number of steps exceeds Table 10, we cannot presume that the numerical solution will function properly.

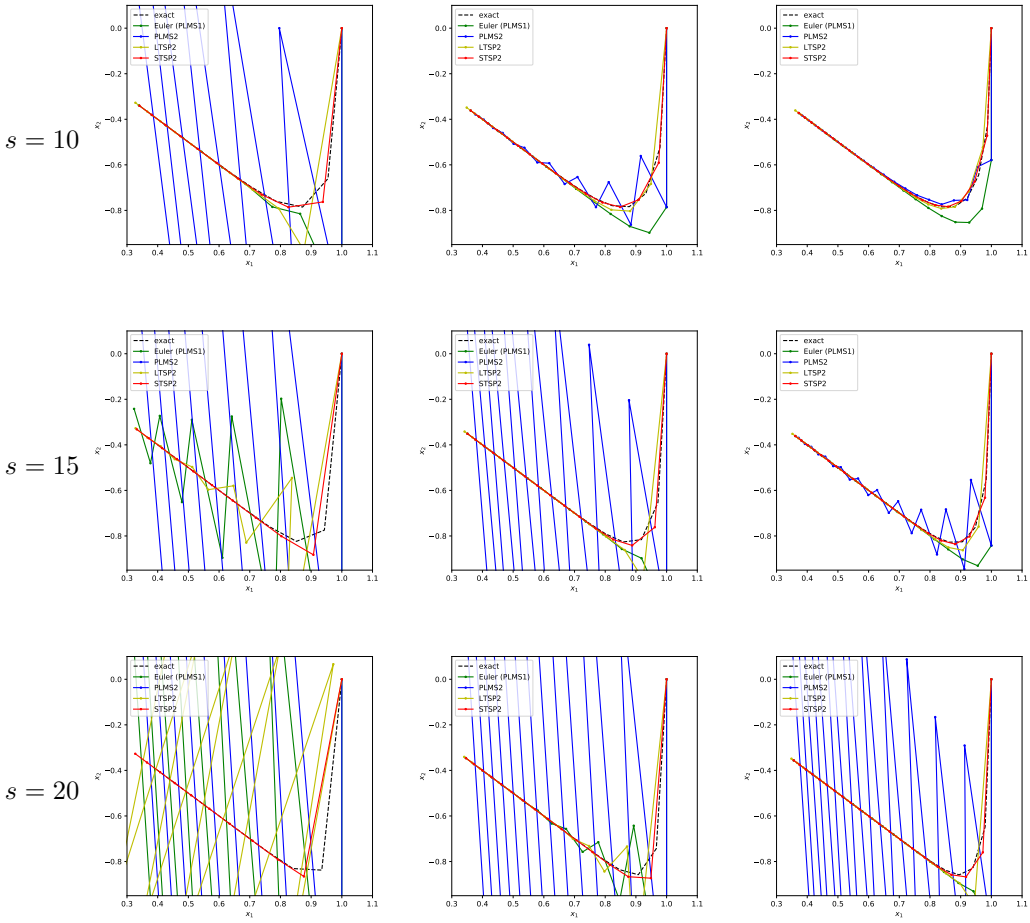


Figure 16: Solution trajectories of different numerical methods when their numbers of steps are close to those in Table 10.