Analysis and review of the DPM-Solver

Giovanni BENEDETTI DA ROSA
Cristian Alejandro CHÁVEZ BECERRA
Yann Fabio NTSAMA

Master 2 Data Science
Institut Polytechnique de Paris-École Polytechnique
Introduction to Generative Models

1 Introduction

Diffusion Probabilistic Models (DPMs) [1] are generative models that have shown outstanding results in many tasks, such as image generation [3], video generation [4]. However, to generate high-quality samples, DPMs typically require hundreds or even thousands(as in the original paper) of sequential steps involving huge neural network evaluations. The goal of the DPM-solver method, presented in the paper *DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps*, is to achieve high-quality samplings in around 10 steps of sequential function evaluations [5]. To achieve such result, the authors cleverly derived a new strategy of solving DPM equations based on the semi-linear structure of Diffusion ODES, that the previous works have previously neglected.

In this work the mathematical framework behind the DPM method will be exposed as well as some experiments related to the task. Notice that the full-derivation of the expressions showed in the paper are attached in the auxiliary document.

2 Methodology and Theoretical Analysis

2.1 Objective

Although black-box ODE solvers had been used in the literature, they require huge amounts of steps. After discretizing the stochastic differential equations of the diffusion models and find their equivalent probability flow ordinary differential equations, the idea here is these ODEs can be simplified to an easier formulation due to its semi-linearity, which makes them easier to be approximated.

2.2 Forward Process and Diffusion SDEs

Let $x_0 \in \mathbb{R}^D$ a random variable with unknown distribuition $q_0(x_0)$. A diffusion probabilistic model (DPM) [5] defines a forward process $\{x_t\}_{t\in[0,T]}$ with $T > 0$ starting with x_0, such that for any $t \in [0, T]$. The distribution of x_t conditioned on x_0 satisfies:

$$q_{0t}(x_t|x_0) = \mathcal{N}(x_t|\alpha(t)x_0, \sigma^2(t)I) \tag{1}$$

where α_t, $\sigma_t \in \mathbb{R}^+$, are differentiable functions of t with bounded derivatives, and are common known as noise schedule of the DPM.

Kingma et al. prove [6] that the following stochastic differential equation(SDE) has the same distribution as stated in Eq. 1, $\forall t \in [0, T]$:

$$dx_t = f(t)x_t\,dt + g(t)\,dw_t, \quad x_0 \sim q_0(x_0), \tag{2}$$

$$f(t) = \frac{d\log\alpha_t}{dt}, \quad g^2(t) = \frac{d\sigma_t^2}{dt} - 2\frac{d\log\alpha_t}{dt}\sigma_t^2. \tag{3}$$

with w_t represents the standard Wiener process.

Now, we will focus on the reverse of the diffusion process. Song et al.[2], based on the results of Anderson [7], prove that the reverse process of 2, stated from T to 0, with marginal distribution $q_T(x_T)$, under some regularity conditions, has an equivalent formulation:

$$dx_t = \left[f(t)x_t - g^2(t)\nabla_x \log q(x_t, t)\right] dt + g(t)d\bar{\mathbf{w}}_t, \quad x_t \sim q_T(x_T) \tag{4}$$

where $\bar{w}_t$ is a standard Wiener process in the reverse time. The only unknown term in Eq. 4 is the score function $\nabla_x \log q(x_t, t)$ at each time t. In practice this value is estimated by a parametrized neural network $\epsilon_\theta(x_t, t)$ that is obtained by minimizing a objective function to estimate a scaled score function $\sigma_t\nabla_x \log q(x_t, t)$ that is, in fact, its groung truth value. So DPMs replace the score function by the parameterized neural network:

$$dx_t = \left[f(t)x_t + \frac{g^2(t)}{\sigma_t}\epsilon_\theta(x_t, t)\right] dt + g(t)d\bar{\mathbf{w}}_t, \quad x_T \sim \mathcal{N}(0, \tilde{\sigma}^2 I). \tag{5}$$

Now, for faster sampling, the associated probability flow ODE of the Diffusion SDE is going to be considered. The associated probability flow ODE was proved by Song et al.[2] to be:

$$dx_t = f(t)x_t + \frac{1}{2}g^2(t)\nabla_x \log q(x_t, t), \quad x_t \sim q_T(x_T) \tag{6}$$

Then, replacing the score function with the noise prediction model (the neural network)[5]:

$$\frac{dx_t}{dt} = h_\theta(x_t, t) := \underbrace{f(t)x_t}_{\text{Linear part}} + \underbrace{\frac{g^2(t)}{2\sigma_t}\epsilon_\theta(x_t, t)}_{\text{Nonlinear part}}, \quad x_T \sim \mathcal{N}(0, \tilde{\sigma}^2\mathbf{I}). \tag{7}$$

The probability flow ODE differs from Langevin dynamics only in the nature of particle evolution: the former is deterministic, while the latter is stochastic [15]. With these formulas, now we will be able the analytical solutions given in the paper to this ODE [**?**].

2.3 Customized Fast Solvers for Diffusion ODEs

As was already cited the equation 7 is semi-linear. So, we can apply variation of constants method:

$$x_t = e^{\int_s^t f(\tau)d\tau}x_s + \int_s^t e^{\int_s^\tau f(\xi)d\xi}\frac{g^2(\tau)}{2\sigma(\tau)}\epsilon_\theta(x_\tau, \tau)\, d\tau \tag{8}$$

The next idea is to introduce a strictly decreasing function(λ_t) defined as $\lambda_t := \log\left(\frac{\alpha_t}{\sigma_t}\right)$. This definition of strictly decreasing comes by construction in a diffusion process. Thus we can rewrite, $g(t)$, based on this new function $g^2(t) = \frac{d\sigma_t^2}{dt} - 2\frac{dlog\alpha_t}{dt} = 2\sigma_t\frac{d\sigma_t}{dt} - 2\frac{dlog\alpha_t}{dt}$. Using eq. 3 and eq. 8, we arrive to:

Proposition 1 Given an initial value x_s at time $s > 0$, the solution x_t at time $t \in [0, s]$ of diffusion ODEs in Equation 7;

$$x_t = \frac{\alpha_t}{\alpha_s}x_s - \alpha_t\int_{\lambda_s}^{\lambda_t} e^{-\lambda}\epsilon_\theta(x_\lambda, \lambda)\, d\lambda \tag{9}$$

The integral in 9 the exponentially weighted integral of ϵ_θ,which is very special and highly related to the exponential integrators in the literature of ODE solvers, as mentioned in the original paper.

Given an initial value x_T at time T and $M+1$ time steps $\{t_i\}_{i=0}^M$ decreasing from $t_0 = T$ to $t_M = 0$, let $\tilde{x}_{t_0} = x_T$ be the initial value. The proposed solvers use M steps to iteratively compute a sequence $\{\tilde{x}_{t_i}\}_{i=0}^M$ that approximates the true solutions at the time steps $\{t_i\}_{i=0}^M$. In particular, the final iterate $\tilde{x}_{t_M}$ approximates the true solution at time 0.

Starting with the previous value $\tilde{x}_{t_{i-1}}$ at time t_i-1, according to Eq. 9, the exact solution $x_{t_{i-1}\to t_i}$ at time t_i is given by:

$$x_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{x}_{t_{i-1}} - \alpha_{t_i}\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\hat{\epsilon}_\theta(\tilde{x}_\lambda, \lambda)\, d\lambda. \tag{10}$$

To compute the value $\tilde{x}_{t_i}$ for approximating $x_{t_{i-1}\to t_i}$, we need to approximate the exponentially weighted integral of $\hat{\epsilon}_\theta$ from $\lambda_{t_{i-1}}$ to λ_{t_i}. Expanding $\hat{\epsilon}_\theta$ using taylor series w.r.t. , it's possible to solve a new integral analytically, computed by repeatedly applying n times of integration-by-parts, as can be seen in the auxiliary document. In this sense, considering h to be the difference between two subsequent λ_t, it's possible to show that by dropping the $O\big(h^{k+1}\big)$ error term and approximating the first $(k-1)$-the total derivatives, we can derive k-th-order ODE solvers for diffusion ODEs.

For instance, let's take $k = 1$:

DPM-Solver-1. Given an initial value x_T and $M+1$ time steps $\{t_i\}_{i=0}^M$ decreasing from $t_0 = T$ to $t_M = 0$. Starting with $\tilde{x}_{t_0} = x_T$, the sequence $\{\tilde{x}_{t_i}\}_{i=1}^M$ is computed iteratively as follows:

$$\tilde{x}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{x}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right)\epsilon_\theta(\tilde{x}_{t_{i-1}}, t_{i-1}),$$

where $h_i = \lambda_{t_i} - \lambda_{t_{i-1}}$.

For $k \geq 2$, approximating the first k terms of the Taylor expansion needs additional intermediate points between t and s. Given these considerations and that for solvers with $k \geq 4$ need much more intermediate points, the authors state the following theorem.

Theorem 1 (DPM-Solver-k as a k-th-order solver). *[5]Assume $\epsilon_\theta(\mathbf{x}_t, t)$ follows the following assumptions:*

- *The total derivatives $\frac{d^j\epsilon_\theta(\mathbf{x}_\lambda, \lambda)}{d\lambda^j}$ (as a function of λ) exist and are continuous for $0 \leq j \leq k+1$.*
- *The function $\epsilon_\theta(\mathbf{x}, s)$ is Lipschitz with respect to its first parameter $\mathbf{x}$.*

– *The maximum step size* $h_{max} = O(1/M)$*, where* M *is the total number of steps.*

, then for $k = 1, 2, 3$, DPM-Solver-k *is a* k*-th order solver for diffusion ODEs, i.e., for the sequence* $\{\tilde{\mathbf{x}}_{t_i}\}_{i=1}^{M}$ *computed by* DPM-Solver-k*, the approximation error at time* 0 *satisfies*

$$\tilde{\mathbf{x}}_{t_M} - \mathbf{x}_0 = \mathcal{O}(h_{max}^k), \quad \textit{where } h_{max} = \max_{1 \le i \le M} (\lambda_{t_i} - \lambda_{t_{i-1}}).$$

The proof of the theorem can be reviewed in the auxiliary document. In figure 1 is possible to visualize the pseudocode of the method of order 1. The complete 1,2,3 versions of the DPM-k methods are available in the suplementary document.

Algorithm 1 DPM-Solver-1

Require: Initial value x_T, time steps $\{t_i\}_{i=0}^{M}$, model ϵ_θ
procedure DPM-SOLVER-1$(\tilde{x}_{t_{i-1}}, t_{i-1}, t_i)$
  $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$
  $\tilde{x}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \tilde{x}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right) \epsilon_\theta\left(\tilde{x}_{t_{i-1}}, t_{i-1}\right)$
  return $\tilde{x}_{t_i}$
end procedure

$\tilde{x}_{t_0} \leftarrow x_T$
for $i \leftarrow 1$ to M **do**
  $\tilde{x}_{t_i} \leftarrow$ DPM-SOLVER-1$\left(\tilde{x}_{t_{i-1}}, t_{i-1}, t_i\right)$
end for
return $\tilde{x}_{t_M}$

Fig. 1: Pseudocode for DPM-Solver-1.

3 Experimental Evaluation

3.1 Methodology

To validate the methods that were described in the paper, we try to reproduce some experiments of denoising a diffusion probabilistic model[11] that were proposed in the paper while proposing other simpler evaluations. To evaluate the solver method, as in the original paper[5], we vary the number of function evaluations (NFE), the number of calls to the noise prediction model $\epsilon_\theta(x_t, t)$, and compare the sample quality using Fréchet Inception Distance (FID) produced by DPM-Solver with that of other methods(DDIM)[12] and DDPM[11], across cat 256 dataset[9], CIFAR 10[8], and a toy model make-moons[10]. Also, empirical visualizations of the images comparing the sampler methods were applied. For comparison purposes and to avoid numerical instabilities, the majority of experiments were conducted using the Diffusers library.

3.2 Results

Makemoons: The makemoons dataset is a simple toy model widely used for machine learning tests that is basically composed of two interleaving half-circles. As a starting point, we used the idea from Denoising Diffusion Probability Model(DDPM)[11] and trained a simple MLP as the denoising model. It is possible to visualize the values of FID and Wasserstein distances versus NFE, respectively in figures 2 and . We can clearly see that there was no huge difference in the behavior of the three solvers applied and that it fastly converged to a low value. We suppose that is related to the fact the dataset is too simple and it can be quickly denoised by these models that are able to look to more complex distributions patterns.

CIFAR: A similar computation as in the previous section was done with CIFAR10 32 x 32, using google pre-trained model available in the *diffusers* [**?**] library, using the DPM-2 solver, with a linear β scheduler. Due to the lack of computational power, to evaluate the FID scores we were able to generate 100 (image 4) and 1k (image5) randomly taken to compare with the real distributions of CIFAR dataset as can be seen in figures 4 and 5.

As we can observe, the FID values remain high due to the limited number of generated images compared to the 60,000 CIFAR dataset samples used in the original paper. However, FID decreases as the sample size increases, from 100 to 1,000, closer to the real distribution. Besides, DPM-Solver starts with a lower FID than DDIM, which is consistent with the original paper, and achieves similar results in subsequent evaluations with much faster sampling than the original DDPM. At this point, it's good to notice that as reported in the paper the DDIM can be written in the form of a DPM-Solver1 solver[5] which explains its similar values with the DPM-Solver.

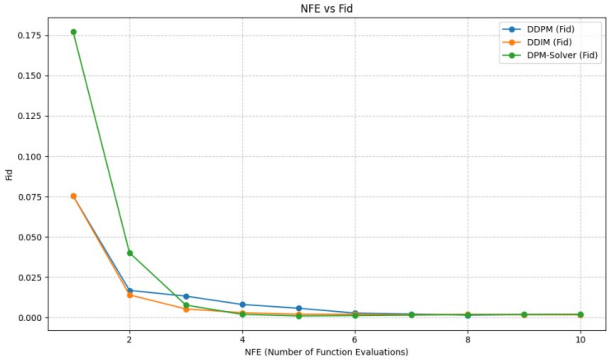

Fig. 2: FID vs NFE make-moons

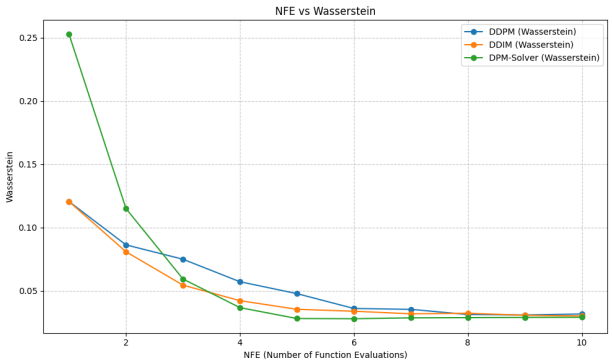

Fig. 3: Wassertein vs NFE make-moons

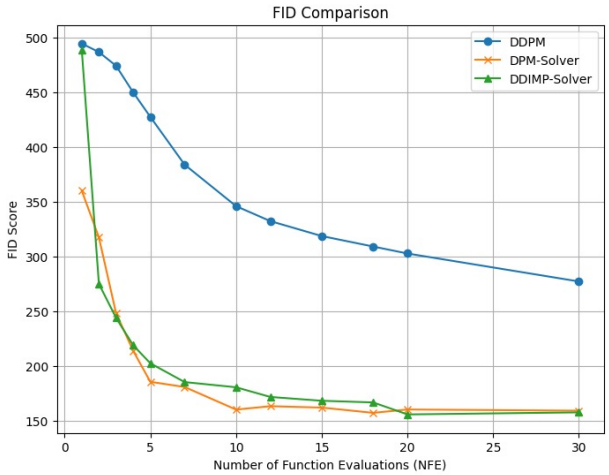

Fig. 4: FID vs NFE 100 images CIFAR

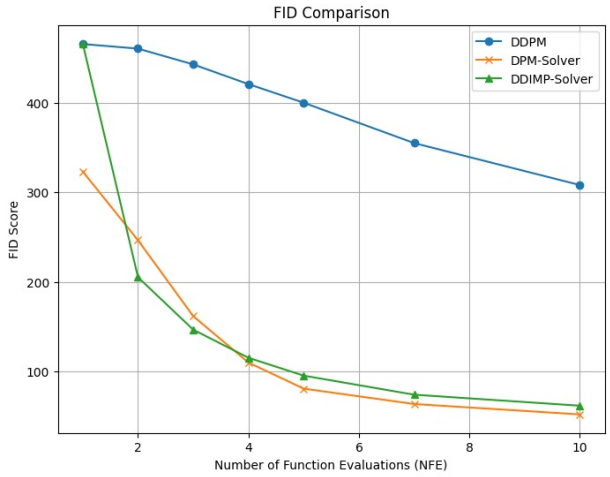

Fig. 5: FID vs NFE 1000 images CIFAR

DDIM and DPM-Solver samples comparisons: The image 6 shows the results of the sampling of a car (from cifar10 dataset) with 10, 15, 20 and 100 NFE's for the DDIM and the sample of the same car with just 10 NFE's in DPM-Solver2. We can appreciate that the wheel and the front of the car start to be visible with 100 iterations in DDIM, something that is achieved with just 10 iterations using DPM-Solver.

Fig. 6: DDIM vs DPM Cifar dataset

A similar image comparison for DDIM and DPM-2 was generated using the LSUN cat dataset and a pre-trained model from diffusers. As we can inspect visually in image 7, the samples generated from DPM with 10 NFEs are much clearer than DDIM for 10 NFEs, as reported in the original paper.

This shows that the algorithm needs only a few function evaluations to converge (Typically between 10 and 20).

Results from our DPM-Solver implementation: Our implementation of the algorithms proposed by the paper was influenced by the authors original implementation, however, the main difference is that we try to keep the most simple algorithms and hyperparameter values in order to have a more understandable code and focus on the essential parts. This entails putting the emphasis on discrete time noise prediction models with simple uniform time schedules. The results and the proof of the sampling improvement of the algorithms can be seen in figures 8, 9. In image 10,

Fig. 7: DDIM vs DPM Cat dataset

we can see that after some steps the image is degenerated. This can be due to a numerical instability problem of our implementation our that is not good enough to denoise all images.

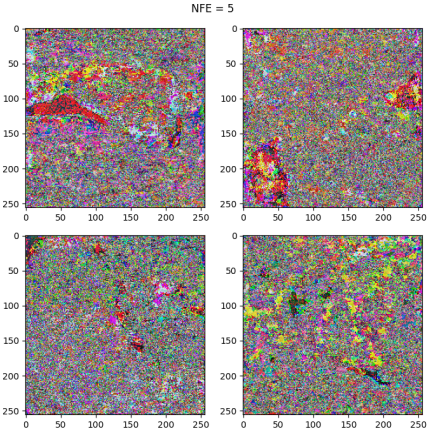

Fig. 8: NFE 5

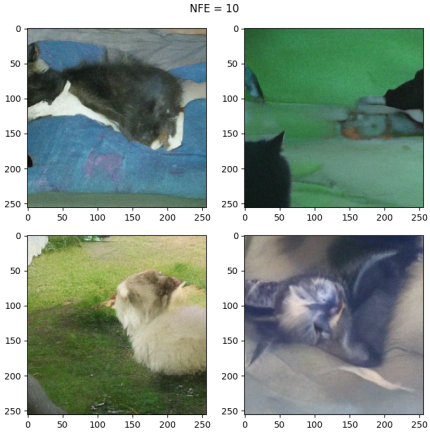

Fig. 9: NFE 10

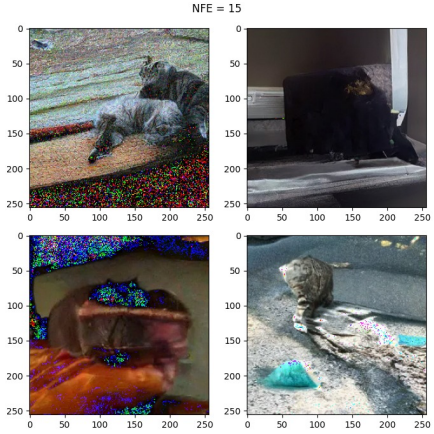

Fig. 10: Denoising instability

Fig. 11: Our implementation

3.3 Discussion

Although the quantitative evaluations of FID vs NFE does not provide the same improvement in sampling to DPMs as described in the original paper [5], it could be seen that the DPM-solver starts from a lower FID value, having slightly better results than DDIM. In this sense, the qualitative confirm the fact that DPM can have higher quality images faster than the already tested methods like the solver from DDPM and DDIM. It was also possible to see that for simpler toy model datasets all the existent solvers converge really fast. Thus, the results from both complex and simple datasets support the hypothesis that DPM-Solver is more efficient than traditional methods.

DPM-Solver effectively reduces the computational cost of sampling from DPMs by leveraging the semi-linearity of diffusion ODEs, making it a promising tool for practical applications that require fast sampling from Diffusion Models. In spite of its promising sampling speed-up performance, our various experimentations proved that the parameter-heavy nature of the algorithm makes it prone to huge performance cuts in terms of the quality of sampling or the stability of the denoising after a large number of steps. One should then carefully tune it before adapting it to its use cases.

4 Conclusion

In this work, we reviewed the DPM-solver method[5], exploring the mathematical framework and reproducing experiments about it. We show how the authors explore the semi-linearity and a change in the integration domain to solve DPMs ODE in a more efficient way than previous methods. By the results that we have with the FID metric (specially with the results from the authors in the original paper) and qualitatively, we could check method is faster than DDIM, despite some denoising instabilities. We could also see that for really simple datasets the velocity of the methods is similar. As shown in Theorem 1, that focus on the 3 first orders DPM-Solver, the higher the order, the smaller the error rate for the same number of NFE's. Future could focus on implementing the 4-th order solver, that, as proved by work [13][14] it would require around 12 calls to the noise prediction model for one step of the algorithm.

References

1. J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6840–6851.
2. Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in *International Conference on Learning Representations*, 2021.
3. P. Dhariwal and A. Q. Nichol, “Diffusion models beat GANs on image synthesis,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 8780–8794.
4. J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, “Video diffusion models,” *arXiv preprint arXiv:2204.03458*, 2022.
5. C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, ”DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps,” *arXiv preprint arXiv:2206.00927*, 2022.
6. D. P. Kingma, T. Salimans, B. Poole, and J. Ho, ”Variational diffusion models,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 21696–21707.
7. B. D. O. Anderson, ”Reverse-time diffusion equation models,” *Stochastic Processes and Their Applications*, vol. 12, no. 3, pp. 313–326, May 1982.
8. A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.
9. F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, “LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop,” in arXiv preprint arXiv:1506.03365, 2015.
10. F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830. [Online]. Available: https://scikit-learn.org.
11. J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 6840–6851.
12. J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” in International Conference on Learning Representations, 2021.
13. M. Hochbruck and A. Ostermann, “Explicit exponential Runge-Kutta methods for semilinear parabolic problems,” SIAM Journal on Numerical Analysis, vol. 43, no. 3, pp. 1069–1090, 2005.
14. V. T. Luan, “Efficient exponential Runge-Kutta methods of high order: Construction and implementation,” BIT Numerical Mathematics, vol. 61, no. 2, pp. 535–560, 2021.
15. M. Yi, “Probability flow ODEs and diffusion probabilistic models,” *Personal Blog*, Feb. 2023. [Online]. Available: https://mingxuan-yi.github.io/blog/2023/prob-flow-ode/.