A Scalable Training-Free Diffusion Model for Uncertainty Quantification

Ali Haisam Muhammad Rafid
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA
haisamrafid@vt.edu

Junqi Yin
National Center of Computational Science
Oak Ridge National Laboratory
Oak Ridge, TN, USA
yinj@ornl.gov

Yuwei Geng
Department of Mathematics
University of South Carolina
Columbia, SC, USA
ygeng@email.sc.edu

Siming Liang
Department of Mathematics
Florida State University
Tallahassee, FL, USA
sliang@fsu.edu

Feng Bao
Department of Mathematics
Florida State University
Tallahassee, FL, USA
fbao@fsu.edu

Lili Ju
Department of Mathematics
University of South Carolina
Columbia, SC, USA
ju@math.sc.edu

Guannan Zhang*
Computer Science and Math Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA
zhangg@ornl.gov

***Abstract*—Generative artificial intelligence extends beyond its success in image/text synthesis, proving itself a powerful uncertainty quantification (UQ) technique through its capability to sample from complex high-dimensional probability distributions. However, existing methods often require a complicated training process, which greatly hinders their applications to real-world UQ problems, especially in dynamic UQ tasks where the target probability distribution evolves rapidly with time. To alleviate this challenge, we have developed a scalable, training-free score-based diffusion model for high-dimensional sampling. We incorporate a parallel-in-time method into our diffusion model to use a large number of GPUs to solve the backward stochastic differential equation and generate new samples of the target distribution. Moreover, we also distribute the computation of the large matrix subtraction used by the training-free score estimator onto multiple GPUs available across all nodes. Compared to existing methods, our approach completely avoids training the score function, making it capable of adapting to rapid changes in the target probability distribution. We showcase the remarkable strong and weak scaling capabilities of the proposed method on the Frontier supercomputer, as well as its uncertainty reduction capability in hurricane predictions when coupled with AI-based foundation models.**

I. Introduction

Generative artificial intelligence (AI) extends beyond its success in image/text synthesis, proving itself a powerful uncertainty quantification (UQ) technique through its capability of learning complex high-dimensional probability distributions. Density estimation is a fundamental problem in UQ, involving approximating a probability density function (PDF) of a given set of observation data and generating an unlimited amount of samples from the target PDF. This process helps to uncover the underlying structure and distribution of the data, which is critical to a variety of UQ tasks including stochastic optimization, Bayesian inference and data assimilation. Generative models, leveraging the expressive power of deep neural networks, have revolutionized various domains, including image synthesis, image denoising, anomaly detection, and natural language processing. Prominent generative modeling techniques include variational auto-encoders (VAEs) [13], generative adversarial networks (GANs) [9], normalizing flows [14], [26], and diffusion models [2], [19], [23]–[25]. Each of these methods offers unique strengths and has contributed to significant advancements in the field.

Despite their success, training generative models for UQ, particularly in an unsupervised manner, presents several challenges. GANs, for instance, often suffer from issues such as mode collapse, vanishing gradients, and training instability. Normalizing flows, on the other hand, face computational inefficiency due to the determinant calculation of the Jacobian matrix. Even though using specifically designed bijective neural network architectures can accelerate the Jacobian computation, the bijective architectures often impose a significant constraint to the expressive power of the resulting normalizing flow model. Score-based diffusion models, while promising, also encounter difficulties related to training the score function needed for solving the backward stochastic process. The performance of diffusion models could dramatically deteriorate when the target PDF is dynamically updated, e.g., in the context of data assimilation. To avoid training the score function, the authors in [17] proposed a novel training-free score estimation scheme that uses Monte Carlo (MC) estimator to directly approximate the closed form of the score function. Avoiding training the score function makes it easier to integrate diffusion models into a general UQ framework, especially for solving dynamic UQ problems where the target PDF evolves over time.

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan.

* Corresponding author

979-8-3503-5554-3/24/$31.00 ©2024 IEEE
DOI 10.1109/SCW63240.2024.00057

However, there are a couple of computational challenges associated with the training-free diffusion model that need to be addressed by exploiting high-performance computing (HPC). First, to ensure the accuracy in UQ, we need to reach the convergence in solving the backward stochastic differential equation (SDE) of the diffusion model with a large number of time steps. The sequential nature of traditional time stepping schemes makes it a very inefficient process. Second, the estimation of the score function requires a large and dense matrix operations. When the dimension of the target random variable is very large (e.g., on the order of $\mathcal{O}(10^7)$ for the example in Figure 4), a single GPU does not have sufficient memory to perform the matrix operations, such that multiple GPUs are needed.

We address the aforementioned challenges by developing a scalable implementation of the training-free diffusion model. Our algorithm has two key components. The first is to implement a parallel-in-time method, specifically the Parareal algorithm [16], to accelerate the time-to-solution in solving the backward SDE of the diffusion model. The Parareal algorithm is a numerical method used to solve time-dependent differential equations by enabling parallel computation across multiple time steps. It achieves this by dividing the overall time domain into smaller subintervals, solving them concurrently using a coarse predictor and iteratively refining with a fine corrector to enhance accuracy and efficiency. With Parareal, we can distribute the computational burden of solving the backward SDE onto many GPUs on a supercomputer. The second component is to distribute the computation of the large matrix subtraction used by the MC-based score estimator onto multiple GPUs available across all nodes. Each GPU receives a corresponding chunk for further calculation. The score is then calculated on each GPU for the chunk assigned to that GPU. After computation, the results are concatenated on the main GPU. We showcase the remarkable strong and weak scaling capabilities of the proposed method on the Frontier supercomputer, as well as its uncertainty reduction capability in hurricane predictions when coupling with AI-based foundation models, e.g., FourCastNet [15].

A. Related work

Generative models, particularly Generative Adversarial Networks (GANs) and Diffusion Models, have significantly advanced the field of machine learning by enabling the generation of realistic data. However, the scalability of these models—defined as their ability to maintain performance while increasing the scale of data and model size—remains a critical research area. As GANs scale, they often suffer from mode collapse, where the generator produces limited varieties of outputs. Larger GANs are harder to train due to issues like vanishing gradients and the need for careful balancing between the generator and discriminator. The iterative nature of diffusion models, which often requires hundreds or thousands of steps, makes them computationally expensive. As the model size and data dimensionality increase, the training and inference time can grow significantly.

Some solutions to the scalability issue have been proposed in the literature. Progressive Growing of GANs (ProGAN) [12] is an approach that starts with low-resolution images and gradually increases the resolution, allowing the model to learn finer details incrementally. BigGAN [4] introduced techniques like class-conditional generation and spectral normalization to stabilize training and improve scalability. Recently, Lada-GAN [20] was proposed that uses novel transformer blocks for both the generator and the discriminator, which reduces the computational complexity and the instability of training GAN. [6] discussed improving diffusion models' architecture, focusing on reducing the number of parameters while maintaining performance. They introduced improvements such as attention mechanisms and residual connections, optimizing the model's efficiency. [23] introduced a score-based generative model framework that leverages continuous-time stochastic differential equations (SDEs) for more efficient sampling. This approach reduces the number of denoising steps, lowering memory and computation requirements.

II. Problem Setting

One central task in UQ is high-dimensional sampling, i.e., drawing samples from a high-dimensional unknown distribution. Specifically, we aim to learn how to generate an unlimited number of samples for a target d-dimensional random variable, denoted as

$$X \in \mathbb{R}^d \text{ and } X \sim p_X(x), \tag{1}$$

where $p_X(x)$ is the PDF of the random variable X. We will achieve this task using generative AI models. In the context of UQ, a generative AI model defines a transport map

$$X = F(Y) \text{ with } Y \in \mathbb{R}^d, \tag{2}$$

from a reference variable Y, following the standard normal distribution, to the target random variable X. Once the transport model is built, we can generate unlimited number of samples of X by firstly sampling Y from the standard normal distribution and then passing these samples through the transport map $F(y)$.

How to construct the transport model F in Eq. (2) has been extensively studied in the machine learning community using normalizing flow models [5], [7], [10], [11], [14], [21], [22], where $F(y)$ is defined as a bijective neural network. The drawback of this approach is the necessity for specially designed reversible architectures for $F(y)$ to efficiently perform back-propagation through the computation of $|\det(\mathbf{D}(F^{-1}(x)))|$. The score-based diffusion models overcome such drawback by using stochastic processes to construct the transport map F in Eq. (2), such that bijective neural networks are no longer needed. However, the score-based diffusion model still requires a fairly complicated training process, i.e., using a neural network to learn the score function. This may be important to the quality of generated images in image synthesis, but not necessary in solving the sampling problem for UQ tasks. Moreover, many UQ tasks, e.g., data assimilation, requires frequent update of the transport map F in Eq. (2) because the target PDF $p_X(x)$ is also time-dependent. In this case, the performance of existing diffusion models will be dramatically hindered by the inefficient training process. We will address this challenge by using a training-free diffusion model that can be scaled on modern GPU-bases supercomputers like Frontier.

III. Methodology

A. Training-free diffusion models

The score-based diffusion model defines the transport between the target random variable X and the standard normal random variable Y in Eq. (2) using two SDEs. The transport from X to Y is defined by the following forward SDE:

$$dZ_t = b(t)Z_t dt + \sigma(t)dW_t, \tag{3}$$

where W_t is the standard Brownian motion. In this work, the coefficients $b(t)$ and $\sigma(t)$ are defined by

$$b(t) = \frac{d(\log \alpha_t)}{dt}, \quad \sigma^2(t) = \frac{d\beta_t^2}{dt} - 2\frac{d(\log \alpha_t)}{dt}\beta_t^2, \tag{4}$$

where the functions $\alpha(t)$ and $\beta(t)$ are defined by

$$\alpha_t = 1 - t \text{ and } \beta_t^2 = t, \tag{5}$$

for $t \in [0, 1]$. This choice of $b(t)$ and $\sigma(t)$ ensures that the forward SDE in Eq. (3) can transport any distribution of X to the standard normal distribution Y when letting $Z_0 = X$, i.e., let the initial state Z_0 be the target random variable [18].

The map F is in the opposite direction from the forward SDE, which is defined by the backward SDE of the form

$$dZ_t = [b(t)Z_t - \sigma^2(t)S(Z_t, t)]dt + \sigma(t)d\overleftarrow{W}_t \tag{6}$$

where $Z_1 = Y$, $Z_0 = X$, $\overleftarrow{W}_t$ is the backward Brownian motion and $S(Z_t, t)$ is the score function of the form

$$S(Z_t, t) = \nabla \log(Q(Z_t, t)), \tag{7}$$

with $Q(Z_t, t)$ being the PDF of the state Z_t in Eq. (3). When the score function is known, we can generate samples of the target random variable X by first generating samples from the standard normal distribution and pushing the samples through the backward SDE. Therefore, the key task in constructing a diffusion model is to approximate the score function in Eq. (7).

Instead of using a neural network to learn the score function, we use the training-free score approximation based on the following closed-form representation of the score function:

$$\begin{aligned} S(Z_t, t) =& \nabla_z \log \left(\int Q(Z_t|Z_0)Q(Z_0)dZ_0 \right) \\ =& \frac{\int -\frac{Z_t - \alpha_t Z_0}{\beta_t^2} Q(Z_t|Z_0)Q(Z_0)dZ_0}{\int Q(Z_t|Z_0')Q(Z_0')dZ_0'} \\ =& -\int \frac{Z_t - \alpha_t Z_0}{\beta_t^2} w(Z_t, Z_0)Q(Z_0)dZ_0, \end{aligned} \tag{8}$$

where the weight function $w(Z_t, Z_0)$ is defined by $w(Z_t, Z_0) = Q(Z_t|Z_0)/\int Q(Z_t|Z_0')Q(Z_0')dZ_0'$ satisfying the condition $\int w(Z_t, Z_0)Q(Z_0)dZ_0 = 1$. Then, the integrals in Eq. (8) can be approximated by MC estimation with the samples from the target PDF, i.e., $Q(Z_0)$ in Eq. (8) (See [17] for detailed derivation of the MC estimator). To speedup the solution of backward SDE in Eq. (6), we need to overcome two challenges:

- ***Sequential time-stepping solver for the SDE in Eq.*** (6)**.** To ensure the accuracy in UQ, we need to reach the convergence in solving the backward SDE with a large number of time steps. The sequential nature of traditional time stepping scheme makes it a very inefficient process.
- ***Large dense-matrix operation in score estimation.*** The estimation of the score function requires a large matrix-matrix subtraction, i.e., $Z_t - \alpha_t Z_0$ in Eq. (8). When the dimension of the target random variable X is very large (e.g., on the order of $\mathcal{O}(10^7)$ for the example in Figure 4), a single GPU does not have sufficient memory to perform the matrix operation.

The above two challenges will be addressed using HPC in the following two subsections.

B. The parallel-in-time scheme for the diffusion model

This subsection describes the parallel-in-time scheme for efficiently solving the diffusion model. Specifically, we employ the Parareal algorithm [16], which has been demonstrated to be a scalable discretization scheme for solving dynamical systems. Because the Parareal algorithm cannot be applied to SDEs, we first convert the backward SDE in Eq. (6) to an ordinary differential equation (ODE), i.e.,

$$dZ_t = \left[b(t)Z_t - \frac{1}{2}\sigma^2(t)S(Z_t, t) \right] dt, \tag{9}$$

where $b(t)$, $\sigma(t)$ and $S(Z_t, t)$ are the same as in Eq. (6). According to [17], [24], the trajectories of the backward ODE is different from the backward SDE, but the distribution of the state Z_t remains the same for any $t \in [0, 1]$. Thus, the backward ODE model can provide us the desired transport map F to generate samples from the target distribution.

The Parareal algorithm is a numerical method designed for solving ODEs on parallel computing architectures. It achieves this by decomposing the temporal domain into several subintervals, which can be processed simultaneously across multiple processors. Initially, the algorithm generates a coarse approximation of the solution over the entire time domain using a computationally inexpensive method. Then, it refines this approximation in each subinterval using a more accurate method, but crucially, it does this in parallel, which significantly speeds up the process. The corrections from the fine solver are used to update the solution iteratively until convergence is achieved. This combination of coarse predictions and fine corrections allows the parareal algorithm to leverage parallel processing capabilities effectively. The illustration of the Parareal algorithm is given in Figure 1.

The parallel-in-time ODE solver for the diffuson model is summarized in Algorithm 1. n_c specifies the number of time intervals that will be used for coarse solving step. For example, if $n_c = 4$ and number of time steps is 100, then we will

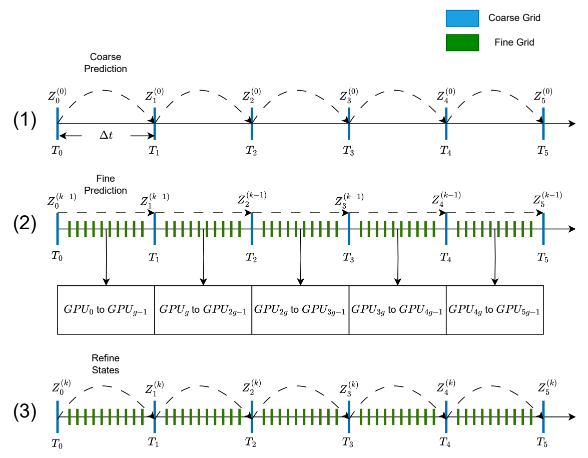
Coarse Grid
Fine Grid
Coarse Prediction
(1)
$Z_0^{(0)}$ $Z_1^{(0)}$ $Z_2^{(0)}$ $Z_3^{(0)}$ $Z_4^{(0)}$ $Z_5^{(0)}$
Δt
T_0 T_1 T_2 T_3 T_4 T_5
Fine Prediction
(2)
$Z_0^{(k-1)}$ $Z_1^{(k-1)}$ $Z_2^{(k-1)}$ $Z_3^{(k-1)}$ $Z_4^{(k-1)}$ $Z_5^{(k-1)}$
T_0 T_1 T_2 T_3 T_4 T_5
GPU_0 to GPU_{g-1}
GPU_g to GPU_{2g-1}
GPU_{2g} to GPU_{3g-1}
GPU_{3g} to GPU_{4g-1}
GPU_{4g} to GPU_{5g-1}
Refine States
(3)
$Z_0^{(k)}$ $Z_1^{(k)}$ $Z_2^{(k)}$ $Z_3^{(k)}$ $Z_4^{(k)}$ $Z_5^{(k)}$
T_0 T_1 T_2 T_3 T_4 T_5

Fig. 1. Illustration of the parallel-in-time (i.e., Parareal) algorithm. For the time domain $[0, T]$, we first split the time range into an arbitrary number of time slices (5 time slices in the figure). In (1), we run the ODE solver sequentially to get the states at times T_1 to T_5. We are running the ODE solver coarsely, as we are using a large time step to go directly from the state Z_{i-1} to Z_i. This sacrifices accuracy but quickly determines the initial states for the next step. In (2), we run the fine solver over each time slices in parallel. For each time slice, we assign g available GPUs to run the ODE solver. In this step, we use small time steps to get a more accurate ODE solve. In (3), the states are refined from the solutions of the coarse and fine solvers. Steps (2) and (3) are repeated until the solutions converge.

use four coarse solvers serially with $t_{\text{vec}}^{(c)}$ values of $[0, 0.25]$, $[0.25, 0.50]$, $[0.50, 0.75]$ and $[0.75, 1]$. This means, instead of using the time step values 0, 0.01, 0.02 gradually up to 1; we go from 0 to 0.25, 0.25 to 0.5, 0.5 to 0.75 and 0.75 to 1. This gives us 4 initial values to use for the fine solvers. We then run the fine solvers for these four intervals in parallel. If the total number of GPUs available is $size$, then we assign $\lfloor size/n_c \rfloor$ GPUs to each fine solver. By fine solving, we mean instead of jumping from time step value 0 to 0.25 for the interval $[0, 0.25]$, we use the time step values $[0, 0.01, \ldots, 0.25]$ to get a more accurate result. For several iterations, the solution values are updated using the update formula described in [16], until the convergence criteria is met.

C. Speeding up the computation of the score function

The computational burden in score estimation lies in the computing matrix subtraction $Z_t - \alpha_t Z_0$ in Eq. (8), where $Z_t \in \mathbb{R}^{N \times d}$ and $Z_0^\top \in \mathbb{R}^{M \times d}$. When the dimension d of the target random variable X in Eq. (1) is extremely large, e.g., d is on the order of $\mathcal{O}(10^7)$ in the example in Figure 4, a single GPU does not have sufficient memory to hold such large matrices. In this work, we split both $Z_t \in \mathbb{R}^{N \times d}$ and $Z_0^\top \in \mathbb{R}^{M \times d}$ into multiple GPUs along the direction of the dimension d. Each GPU receives a corresponding chunk for further calculation. The score function is then calculated on each GPU for the chunk assigned to that GPU. After computation, the results are concatenated on the main GPU along the direction of size d. Our approach ensures that the matrix operations for score computation are independent of the chunks sent to other GPUs. Our algorithm is illustrated in

Algorithm 1 The parallel-in-time solver for diffusion models

```
Require: $Z_t \in \mathbb{R}^{N \times d}$, $Y \in \mathbb{R}^{M \times d}$, $t_{\text{vec}}$ which is an array containing time step values between 0 and 1 depending on the number of time steps, $n_c$ the number of coarse time intervals for the parareal algorithm, $\varepsilon$ tolerance
1: $size \leftarrow$ number of GPUs available
2: Number of GPUs to use for each time slice for the fine solver: $g \leftarrow \lfloor size/n_c \rfloor$
3: Initialize $zt_{\text{coarse}}$, $zt_{\text{fine}}$, and $zt_{\text{parareal}}$ tensors with appropriate dimensions
4: $T \leftarrow$ length of $t_{\text{vec}}$
5: $ts \leftarrow T/n_c$
6: for $i = 0$ to $n_c - 1$ do
7:     $t_{\text{vec}}^{(c)} \leftarrow [t_{\text{vec}}[i * ts], t_{\text{vec}}[(i+1) * ts]]$
8:     $zt_{\text{coarse}}[i+1] \leftarrow ODE\_solver(zt_{\text{coarse}}[i], Y, t_{\text{vec}}^{(c)})$
9:     Broadcast $zt_{\text{coarse}}[i+1]$ from rank 0 to all processes
10: end for
11: $zt_{\text{parareal}}[:] \leftarrow zt_{\text{coarse}}[:]$
12: for $k = 0$ to $n_c - 1$ do
13:     for $i = k$ to $n_c - 1$ do
14:         $t_{\text{vec}}^{(f)} \leftarrow t_{\text{vec}}[i * ts : (i+1) * ts + 1]$
15:         Assign $g$ available GPUs for the fine-solver in the following step
16:         $zt_{\text{fine}}[i+1] \leftarrow ODE\_solver(zt_{\text{parareal}}[i], Y, t_{\text{vec}}^{(f)})$
17:         Broadcast $zt_{\text{fine}}[i+1]$ to all processes
18:     end for
19:     $zt_{\text{parareal}}^{\text{old}} \leftarrow zt_{\text{parareal}}$
20:     for $i = k$ to $n_c - 1$ do
21:         $t_{\text{vec}}^{(c)} \leftarrow [t_{\text{vec}}[i * ts], t_{\text{vec}}[(i+1) * ts]]$
22:         $znc \leftarrow ODE\_solver(zt_{\text{parareal}}[i], Y, t_{\text{vec}}^{(c)})$
23:         Broadcast $znc$ to all processes
24:         $zt_{\text{parareal}}[i+1] \leftarrow znc + (zt_{\text{fine}}[i+1] - zt_{\text{coarse}}[i+1])$
25:         $zt_{\text{coarse}}[i+1] \leftarrow znc$
26:     end for
27:     if $\max(|zt_{\text{parareal}} - zt_{\text{parareal}}^{\text{old}}|) < \varepsilon$ then
28:         break
29:     end if
30: end for
31: return $zt_{\text{parareal}}$
```

Figure 2 and detailed in Algorithm 2. Note that Algorithm 2 will be used by the ODE solver in Algorithm 1.

IV. Experiments

A. Scalability demonstration

Hardware and software. We perform the experiments on the first Exascale supercomputer, Frontier. Each Frontier node is equipped with four AMD Instinct MI250X GPUs with dual Graphics Compute Dies (GCDs) and one third-generation EPYC CPU. A GCD is viewed as an effective GPU, and we use GCD and GPU interchangeably in the following discussion. All four MI250Xs (eight effective GPUs) are connected using 100 GB/s Infinity Fabric (200 GB/s between 2 GCDs of MI250X), and the nodes are connected via a Slingshot-11

Algorithm 2 Score function computation

```
Require: $Z_t \in \mathbb{R}^{N\times d}$, $Z_0^\top \in \mathbb{R}^{M\times d}$, $\alpha_t$, $\beta_t^2$
 1: $size \leftarrow$ number of GPUs available
 2: Split $Z_t$ and $Y$ into $size$ number of along the 2nd dimension
 3: chunks1 $\leftarrow$ chunks of $Z_t$
 4: chunks2 $\leftarrow$ chunks of $Z_0^\top$
 5: if rank == 0 then
 6:     for each process $i$ in 1 to $size - 1$ do
 7:         Send chunks1[$i$] and chunks2[$i$] to process $i$
 8:     end for
 9: else
10:     local_chunk1 $\leftarrow$ corresponding chunk of $Z_t$ from rank 0
11:     local_chunk2 $\leftarrow$ corresponding chunk of $Z_0^\top$ from rank 0
12: end if
13: $S \leftarrow \left(-\frac{1}{\beta_t^2}\right)$(local_chunk1[:, None, :] $- \alpha_t \cdot$ local_chunk2)
14: Compute the final score:
15: score $\leftarrow$ mean value of $S$ across 2nd dimension
16: if rank == 0 then
17:     Gather the score from all processes
18:     Concatenate the gathered scores along the 2nd dimension
19: end if
20: return  Final concatenated score
```

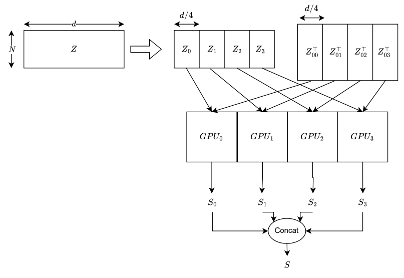

Fig. 2. This figure illustrates how to compute the score function in parallel on GPUs. The data $Z_t \in \mathbb{R}^{N\times d}$ and the samples $Z_0^\top \in \mathbb{R}^{M\times d}$ are split into chunks along the second dimension, depending on the number of available GPUs (4 in this figure). Then, chunks Z_0^i and Z_t^i are sent to corresponding GPU_i. Scores are computed for the chunks in parallel using Eq. (8). The computed scores S_i for each GPU are then concatenated along the 2nd dimension to get the final score.

interconnect with 100 GB/s of bandwidth. Our evaluation is based on PyTorch v2.2.1 and ROCm v5.7.0. We use AWS OFI RCCL plugin for communication, which is built against libfabric v1.15.2.0.

Scaling test. We study both the strong and weak scaling of our parallel ODE solver on Frontier, as shown in Figure 3. For strong scaling, we fix the feature dimension to be 10^6, and evaluate the time-to-solution from 16 to 128 GPUs. For weak scaling, each GPU has a fixed workload with the feature dimension size 12500. The time-to-solution is evaluated up to

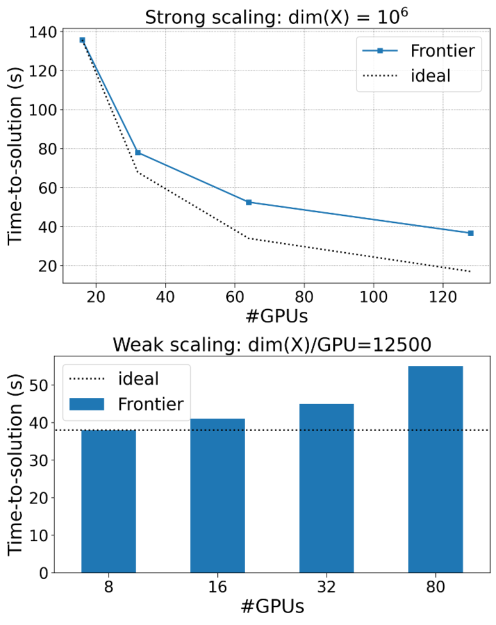

Fig. 3. Strong and weak scaling of our parallel ODE Solver on Frontier .

80 GPUs, i.e., a total feature dimension of 10^6.

B. Scientific demonstration

We demonstrate the performance of our UQ method by applying it to quantify uncertainties of an AI-driven model, FourCastNet [15], in predicting hurricane paths. The AI-driven models are more computationally efficient than traditional physics-based models. However, the performance of the AI models is significantly affected by uncertainties. In this work, we focus on reducing the uncertainty arising from the initial condition. Because FourCastNet was trained on the Fifth Generation of ECMWF Atmospheric Reanalysis of the Global Climate (ERA5), produced by the European Centre for Medium-Range Weather Forecasts (ECMWF), we treat the ERA5 data as the ground truth.

We use Hurricane Lee (2023) as an example to discuss the effect of uncertainty in initial conditions on the hurricane path prediction. We choose September 8, 2023 as the "current time", i.e., Day 0, and use FourCastNet to predict the path of the hurricane's center from Day 1 to Day 8. The initial condition is constructed using the historical observation data. Here, the observation data of the atmosphere state variables, e.g., temperature, humidity, wind speed, geo-potential, etc., for Day 0, Day -1, Day -2 and Day -3 are generated synthetically by adding random perturbations to the ground truth state from ERA5. The correlation structures of the random perturbation is defined according to the information about observation data collection given in the National Hurricane Center's Tropical Cyclone Report [8]. The dimension of FourCastNet is on the order of $\mathcal{O}(10^7)$. To illustrate the performance of our method, we conduct experiments in the following two cases:

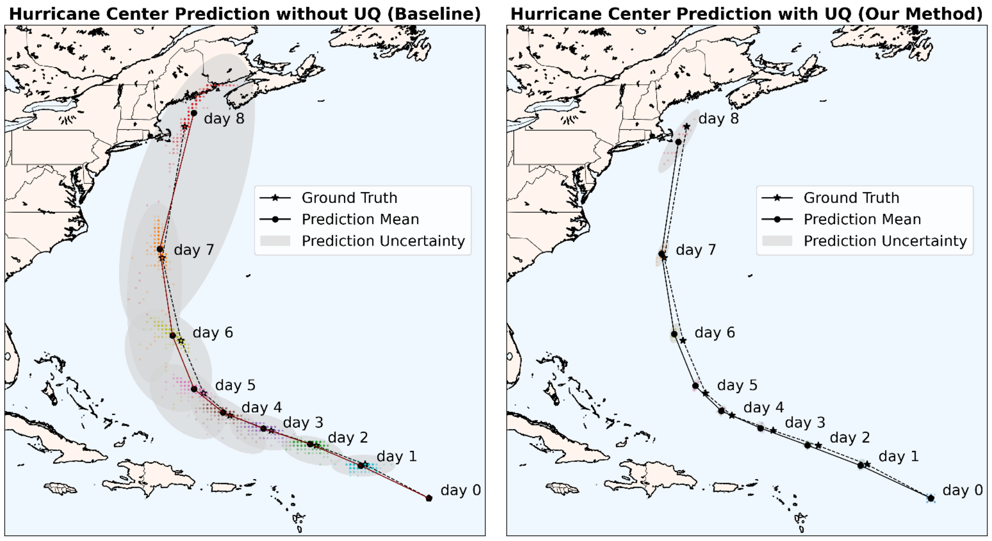
Hurricane Center Prediction without UQ (Baseline)
Hurricane Center Prediction with UQ (Our Method)
Ground Truth
Prediction Mean
Prediction Uncertainty
day 0
day 1
day 2
day 3
day 4
day 5
day 6
day 7
day 8

Fig. 4. (Left) The grey ellipse represents the prediction uncertainty without UQ (Baseline). With 20 perturbed initial conditions, the uncertainty of hurricane center predictions made by FourCastNet expands significantly over time. By day 8, the predicted hurricane centers span a broad area near the East Coast, rendering the predictions too uncertain to be informative. (Right) The grey ellipse depicts the reduced uncertainty in predictions enhanced by UQ (Our method). This UQ approach mitigates the effects of observational noise, enhances the accuracy of initial conditions, and consequently improves the AI model's predictive accuracy. We observe that the prediction uncertainty with UQ is substantially smaller than that without UQ, and it accurately encompasses the ground truth, thereby enhancing the reliability of the predictions.

- **Case I (Baseline): prediction without UQ.** The noisy observation data is directly used as the initial condition for FourCastNet to make 8-day prediction of the hurricane.
- **Case II (Our method): Prediction assisted by UQ.** We apply the proposed UQ method in the context of data assimilation [1], [3], which combines the observation data (from Day -3 to Day 0) and the FourCastNet model to generate the initial conditions for FourCastNet to make the 8-day prediction of the hurricane.

Figure 4 shows the comparison results. When the FourCastNet is used without UQ, the prediction uncertainty grows very quickly, such that the uncertainty of the Day 8 prediction is too large to be informative, as shown in the left subfigure in Figure 4. Even so, it is hard to make a decision under such large uncertainty in practice when the ground truth is unknown. In comparison, Figure 4 (right) shows that after applying our UQ method, the uncertainty in FourCastNet's prediction is much smaller, significantly improving the reliability of the prediction results. This demonstrates that a UQ procedure is critical to reduce uncertainty in the initial condition of the AI model.

We emphasize that this experiment is set up in a relatively ideal situation, as it assumes that all state variables can be observed with added noise. Even in this ideal scenario, the prediction uncertainty without proper UQ is too large to be informative. In practice, the observation data typically exhibit even greater uncertainty due to factors such as sparse observations and inconsistencies from various devices like aircraft, Dvorak, and SATCON [8]. This significantly increases the uncertainty in the initial conditions for FourCastNet, presenting more challenging problems. Therefore, despite the promising results shown in Figure 4, the need for rigorous UQ in AI-based models must be emphasized for future studies.

V. Conclusion

We introduce a scalable, training-free score-based diffusion model designed for high-dimensional sampling tasks, particularly useful in dynamic uncertainty quantification (UQ) where target distributions evolve rapidly. By integrating the Parareal algorithm, our method utilizes multiple GPUs to parallelize and accelerate the solution of backward ODEs, effectively distributing computational loads. We also optimize resource use by splitting large matrix calculations across available GPUs, allowing our model to adapt to changes in probability distributions swiftly without the need for training the score function. Demonstrated on the Frontier supercomputer, our approach showcases remarkable scalability and reduces uncertainty in AI-based hurricane prediction models like FourCastNet, emphasizing its potential in real-world applications. Despite the promising results, several aspects of the proposed method need to be improved in order to be used in operation. For example, observation data, i.e., the data used to estimate the score function of the diffusion model, is the direct observation of the target random variable with added noise. In practice, the observation data is usually indirect, such that the computation of a likelihood function is needed to incorporate the indirect observation data. Another module needs to be added to Algorithm 2 to handle indirect observation.

Acknowledgement

This work is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program, under the contract ERKJ443. ORNL is operated by UT-Battelle, LLC., for the U.S. Department of Energy under Contract DE-AC05-00OR22725. Feng Bao would also like to acknowledge the support from U.S. National Science Foundation through project DMS-2142672 and the support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Grant DE-SC0022297. Lili Ju would also like to acknowledge the support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Grant DE-SC0025527. This research used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility at the Oak Ridge National Laboratory supported by the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Bao, F., Chipilski, H. G., Liang, S., Zhang, G., and Whitaker, J. S. Nonlinear ensemble filtering with diffusion models: Application to the surface quasi-geostrophic dynamics. *arXiv preprint arXiv:2404.00844* (2024).

[2] Bao, F., Zhang, Z., and Zhang, G. An ensemble score filter for tracking high-dimensional nonlinear dynamical systems. *ArXiv:2309.00983* (2024).

[3] Bao, F., Zhang, Z., and Zhang, G. A score-based filter for nonlinear data assimilation. *Journal of Computational Physics* (2024), 113207.

[4] Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019).

[5] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. Generative adversarial networks: An overview. *IEEE signal processing magazine 35*, 1 (2018), 53–65.

[6] Dhariwal, P., and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems 34* (2021), 8780–8794.

[7] Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (2017).

[8] Eric, B., and Heather, N. National hurricane center tropical cyclone report - hurricane lee. Tech. rep., National Hurricane Center, 2024.

[9] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems 27* (2014).

[10] Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019), OpenReview.net.

[11] Guo, L., Wu, H., and Zhou, T. Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models. *Journal of Computational Physics 461* (2022), 111202.

[12] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018).

[13] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (2014), Y. Bengio and Y. LeCun, Eds.

[14] Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence 43*, 11 (2020), 3964–3979.

[15] Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., and Anandkumar, A. FourCastNet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (New York, NY, USA, 2023), PASC '23, Association for Computing Machinery.

[16] Lions, J.-L., Maday, Y., and Turinici, G. A "parareal" in time discretization of pde's. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics 332*, 7 (2001), 661–668.

[17] Liu, Y., Yang, M., Zhang, Z., Bao, F., Cao, Y., and Zhang, G. Diffusion-model-assisted supervised learning of generative models for density estimation. *Journal of Machine Learning for Modeling and Computing 5*, 1 (2024), 25–38.

[18] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems* (2022), A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds.

[19] Lu, D., Liu, Y., Zhang, Z., Bao, F., and Zhang, G. A diffusion-based uncertainty quantification method to advance e3sm land model calibration. *Journal of Geophysical Research: Machine Learning and Computation 1*, 3 (2024), e2024JH000234.

[20] Morales-Juarez, E., and Pineda, G. F. Efficient generative adversarial networks using linear additive-attention transformers. *CoRR abs/2401.09596* (2024).

[21] Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in neural information processing systems 30* (2017), 2338–2347.

[22] Rezende, D., and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning* (2015), PMLR, pp. 1530–1538.

[23] Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems 34* (2021), 1415–1428.

[24] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations* (2021).

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

[26] Yang, M., Wang, P., del-Castillo-Negrete, D., Cao, Y., and Zhang, G. A pseudo-reversible normalizing flow for stochastic dynamical systems with various initial conditions. *SIAM Journal on Scientific Computing 46*, 4 (2024), C508–C533.