Uncertainty propagation for random field outputs using denoising diffusion models

Pavankumar Koratikere[1 [0009−0007−1491−4654]], Leifur Leifsson[1 [0000−0001−5134−870X]], Slawomir Koziel[2,3 [0000−0002−9063−2647]], and Anna Pietrenko-Dabrowska[3 [0000−0003−2319−6782]]

[1] School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana 47907, USA
{pkoratik@purdue.edu, leifur@purdue.edu}
[2] Engineering Optimization & Modeling Center, Department of Engineering, Reykjavík University, Menntavegur 1, 102 Reykjavík, Iceland
koziel@ru.is
[3] Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
anna.dabrowska@pg.edu.pl

Abstract. Uncertainty propagation (UP) is a popular method for estimating randomness in model outputs by propagating input uncertainties through the model. Several methods have been proposed for UP through an expensive-to-evaluate model, including surrogate-based approaches. These methods rely on approximating input-output relation to create a fast-to-evaluate model which can then be used for UP. However, these approaches are limited by the curse of dimensionality and are not easily scalable to UP problems where the model output is a random field. To alleviate these challenges with quantifying random field outputs, this work proposes to utilize diffusion models to perform UP. Specifically, denoising diffusion probabilistic models (DDPMs) are used for tackling UP for random field outputs. DDPM is an unsupervised model, and hence, does not learn input-output mapping. Moreover, DDPMs have been shown to work well for complex high-dimensional distributions. The DDPM-based UP is demonstrated on two physics-based problems and is evaluated against direct Monte Carlo (MC) simulations. The DDPM-based approach is able to accurately learn the underlying distribution using small number of samples, while the MC simulation requires thousands of samples. These results indicate that DDPM can be an efficient tool for quantifying uncertainty in random output fields.

Keywords: Uncertainty propagation · random output fields · denoising diffusion probabilistic models · deep neural networks · diffusion models

1 Introduction

Uncertainty quantification (UQ) is a widely used tool in various science and engineering applications [13,9]. The primary aim of UQ is to determine the impact

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

on model outputs due to the randomness in model parameters. It consists of three steps: (i) identifying the type of uncertainty in model parameters, (ii) using a mathematical framework to quantify this uncertainty, and (iii) uncertainty propagation (UP) (also known as forward UQ). The UP consists of propagating randomness in parameters through the model to quantify uncertainty in model output. This work focuses on UP, especially for the models that consists of random field outputs.

Monte Carlo (MC) methods are one of the most popular approaches for UP. It consists of sampling from the parameter uncertainty distribution and evaluating the model on the generated samples to obtain distribution for the model outputs [9]. This approach is not feasible when model is computationally expensive to evaluate. Alternatively, a fast and cheap-to-evaluate model can be constructed for evaluating a sample. Common models include Gaussian process regression [12], polynomial chaos expansion [15] (PCE), and neural networks [7] (NN). These methods approximate the mapping between input and output, and hence, are limited by the curse of dimensionality. Moreover, these methods are primarily designed for finite number of inputs and outputs, and may not always scale to random fields. However, dimensionality reduction approaches using PCE [6] and NNs [2] have been proposed for UP to alleviate the challenges posed by uncertain random field.

Recently, denoising diffusion probabilistic models (DDPMs) have gained a lot of interest, owing to their effectiveness in approximating complex high-dimensional probability distributions [4,1]. Specifically, DDPMs are a class of generative models that learn an underlying data distribution by training on samples from that same distribution. Note that DDPM is not modeling input-output relation, rather it approximates the unknown distribution. As a result, it is not limited by the number of uncertain parameters and can circumvent the input curse of dimensionality.

Owing to these advantages, this work proposes a novel method for utilizing DDPM for UP tasks, where the model output is a random field. The proposed method is demonstrated on UP task for two physics-based problems and is benchmarked against direct MC simulations. The results indicate that DDPM is able to accurately learn the underlying data distribution with a small number of samples.

The remainder of this paper is organized as follows: Section 2 provides an overview of DDPM and outlines the proposed method. It also provides a description of the deep neural network architecture used within DDPM. Section 3 presents the result of performing UP using DDPM. Lastly, Section 4 concludes this work, along with some recommendations for future work.

2 Methods

This section briefly introduces denoising diffusion models, followed by the proposed algorithm for performing UP using diffusion models. Lastly, it provides a detailed description of the deep neural network model used in this work.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

2.1 Denoising diffusion probabilistic models

Denoising diffusion probabilistic model (DDPM) is an unsupervised generative model that learns a target probability distribution $p(\mathbf{x})$ using the samples from the same distribution. Specifically, DDPM approximates $p(\mathbf{x})$ with a parametric distribution $p_\theta(\mathbf{x}_0)$, where $\mathbf{x}_0$ is a sample from the original distribution [4]. Mathematically, it can be written as

$$p(\mathbf{x}) \approx p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}, \tag{1}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_T$ represent latent variables of the same dimension as $\mathbf{x}_0$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is defined via a Markov chain and is written as

$$p_\theta(\mathbf{x}_{0:T}) = p_T(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{2}$$

where transition probability distribution follows a Gaussian and is given by

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}). \tag{3}$$

The mean μ_θ and σ_t are written as

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t), \text{ and } \sigma_t^2 = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}. \tag{4}$$

As shown in Figure 1, DDPM starts with a sample $\mathbf{x}_T$ from a standard gaussian distribution which is then progressively denoised to obtain a sample $\mathbf{x}_0$ from target distribution. Now, the primary task is to learn a noise predictor $\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)$.

To learn an appropriate noise predictor for this reverse process, a Markov chain is used to fully define a forward process that progressively corrupts $\mathbf{x}_0$ with Gaussian noise to obtain $\mathbf{x}_T$ [4,11]. This forward process is written as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{5}$$

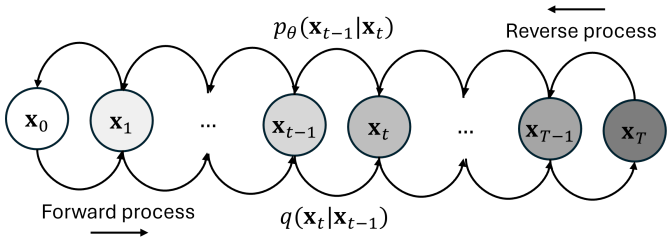

Fig. 1: Forward and reverse process within a diffusion model.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

where the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is assumed to be Gaussian so that it is of the same form as $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Mathematically, it is written as

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{6}$$

where $\beta_1, \ldots, \beta_T$ represents variance scheduler that controls how much Gaussian noise is added at each time step t. Based on this forward process definition, α_t and $\bar{\alpha}_t$ can be computed as

$$\alpha_t = 1 - \beta_t, \text{ and } \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s. \tag{7}$$

Using the properties of Markov chain and reparameterization trick [5], $\mathbf{x}_t$ can be directly computed from $\mathbf{x}_0$ using

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{8}$$

For any given t and $\mathbf{x}_t$, the goal is to train $\hat{\boldsymbol{\epsilon}}$ so that the difference between forward and reverse process is minimized. Based on Ho et al. [4], a simplified version of negative log-likelihood of evidence lower bound is used as the loss function, which is written as

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}), t \sim U[1,T], \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta[\mathbf{x}_t(\mathbf{x}_0, t)] \right\|^2 \right]. \tag{9}$$

Note that, in above equation, $\mathbf{x}_t$ is computed using (8). A deep neural network model, parametrized with θ, is used as a noise predictor. The loss function $\mathcal{L}$ is minimized with respect to θ to obtain the parameters of the network. In this work, adaptive moment (ADAM) optimization method is used for training the network with a learning rate of 0.0001. The network is trained for 5000 epochs with a batch size of 64. The total number of time steps T is set to 200. These

Algorithm 1 Training process for a diffusion model [10]

input: training data $\mathbf{x}_i, i = 1, \ldots, N$
repeat
  for $j \in \mathcal{B}$ **do** ▷ for each sample index in batch
    $t \sim U[1, \ldots, T]$ ▷ sample t randomly
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ sample noise
    $\mathcal{L}_j(\theta) = \left\| \boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\sqrt{\alpha_t}\mathbf{x}_j + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}) \right\|^2$ ▷ compute loss
  end for
  $\mathcal{L}(\theta) = \sum_j \mathcal{L}_j(\theta)$ ▷ accumulate losses
  take gradient of $\mathcal{L}$ with respect to θ
  update model parameters θ using gradient
until convergence
output: model parameters θ

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

Algorithm 2 Sampling process using diffusion model [10]

input: trained model $\hat{\boldsymbol{\epsilon}}_\theta$
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 2$ **do**
  $\boldsymbol{\mu}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{\beta_1}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t)$ ▷ compute mean for next step
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_{t-1} = \boldsymbol{\mu}_{t-1} + \sigma_t \boldsymbol{\epsilon}$ ▷ next sample
end for
$\mathbf{x} = \frac{1}{\sqrt{\alpha_1}}\mathbf{x}_1 - \frac{\beta_1}{\sqrt{\alpha_1}\sqrt{1-\bar{\alpha}_1}}\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_1)$ ▷ final sample
output: sample $\mathbf{x}$

optimization parameters are determined using manual trial and error process. Algorithm 1 and 2 summarize the training and sampling process used in this work, respectively. Refer to Ho et al. [4], Sohl-Dickstein at el. [11], and Lai C et al. [8] for a detail review of diffusion models.

2.2 Uncertainty propagation using DDPM

Consider a computationally expensive function $\mathcal{F} : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^m$, where d is the number of uncertain variables, n is the number of deterministic variables, and m is the number of outputs. The function $\mathcal{F}$ can be written as

$$\mathbf{Y} = \mathcal{F}(\mathbf{X}, \boldsymbol{\varphi}), \tag{10}$$

where $\mathbf{X} \in \mathbb{R}^d$ is a random vector representing uncertain variables and $\boldsymbol{\varphi} \in \mathbb{R}^n$ denotes deterministic variables. In this work, $\boldsymbol{\varphi}$ is held constant and the focus is only on uncertain variables. Since input to the function $\mathcal{F}$ is uncertain, the output $\mathbf{Y} \in \mathbb{R}^m$ is also a random vector.

The UP problem consists of propagating the uncertainty in $\mathbf{X} \sim P_{\mathbf{X}}(\mathbf{x})$ through $\mathcal{F}$ to quantify the uncertainty in $\mathbf{Y} \sim P_{\mathbf{Y}}(\mathbf{y})$. Hence, the goal is to find the distribution $P_{\mathbf{Y}}(\mathbf{y})$ or some of the statistics of $\mathbf{Y}$, such as mean or standard deviation, induced by input uncertainty. This paper specifically deals with problems that have uncertain field outputs. Moreover, $\mathcal{F}$ can be computationally expensive and only a limited number of evaluations can be made.

DDPM is shown to approximate complex high-dimensional probability distributions [4,1]. Since the output of $\mathcal{F}$ is also a random field, this work proposes a method for approximating $P_{\mathbf{Y}}$ using DDPM. Algorithm 3 outlines the proposed method. The core idea is straight-forward: generate samples from the target distribution $P_{\mathbf{Y}}$ and use DDPM to approximate it. This is a fundamentally different approach to UP problem as compared to other methods. DDPM is not modeling an input-output relation, instead it learns the underlying distribution $P_{\mathbf{Y}}$. Moreover, this method is independent of the number of input variables (uncertain or deterministic), and hence, circumvents the input curse of dimensionality. DDPM

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

Algorithm 3 Uncertainty propagation using DDPM (this work)

1: **input**: $P_{\mathbf{X}}$, $\mathcal{F}$, $\boldsymbol{\varphi}$, N
2: $\mathbf{x}_i \leftarrow$ generate N samples from $P_{\mathbf{X}}$
3: $\mathbf{y}_i \leftarrow$ compute output at $\mathbf{x}_i$ using $\mathcal{F}$
4: $\hat{\boldsymbol{\epsilon}}_\theta \leftarrow$ train DDPM using $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ ▷ c.f. Algorithm 1
5: generate samples from $P_{\mathbf{Y}}$ using $\hat{\boldsymbol{\epsilon}}_\theta$ ▷ c.f. Algorithm 2
6: compute mean and standard deviation using sampled points
7: **output**: samples from $P_{\mathbf{Y}}$, mean and standard deviation of $\mathbf{Y}$

also has a strong theoretical foundation via the Fokker-plank equation and the stochastic differential equation [8].

Meanwhile, there are certain challenges with the proposed approach. Firstly, it is assumed that sampling from $P_{\mathbf{X}}$ and then evaluating it using $\mathcal{F}$ will generate diverse sample set from $P_{\mathbf{Y}}$. This may not be true always. Secondly, even though training a DDPM is more stable than other unsupervised generative models, it is still difficult, especially given the number of hyperparameters. Lastly, depending on the complexity of $P_{\mathbf{Y}}$, DDPM might require a lot of samples to learn the underlying distribution which is not always feasible. However, it is worthwhile exploring DDPM for UP problems, given its effectiveness in approximating complex high-dimensional distribution.

2.3 Network architecture

This section describes the architecture of the deep neural network model used in this work for noise prediction. As shown in Figure 2a, a deep convolutional neural network is used. Note that this architecture is determined using manual trial and error process for the problems demonstrated in Section 3.

The model consists of a series of blocks that perform computation on the given input sample $\mathbf{x}$ at a time step t. A sinusoidal embedding method [1] is used to convert discrete time step t to a continuous representation that can then be used by the model. In this work, time step t is embedded into 128 dimensions. The input $\mathbf{x}$ is first processed by a 1D convolutional unit, followed by a nonlinear activation function. This convolution unit converts one input channel to 64 output channels. This is followed by 5 individual computation blocks, whose architecture is described in Figure 2b.

Each block first applies a convolution operation to the given input, followed by adding time step information. The embedded time is processed through a one layer fully connected network (FCNN) before adding it to output channels of the convolution unit. This addition of time step information is crucial for the model to correctly predict noise $\boldsymbol{\epsilon}$ at given t. The weights of FCNN are initialized using the He method [3]. Note that each block has its own FCNN. Next, the combined output is passed through a group normalization unit before passing it through a nonlinear activation function. This normalization prevents exploding gradient issues for the activation function and is shown to be better

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

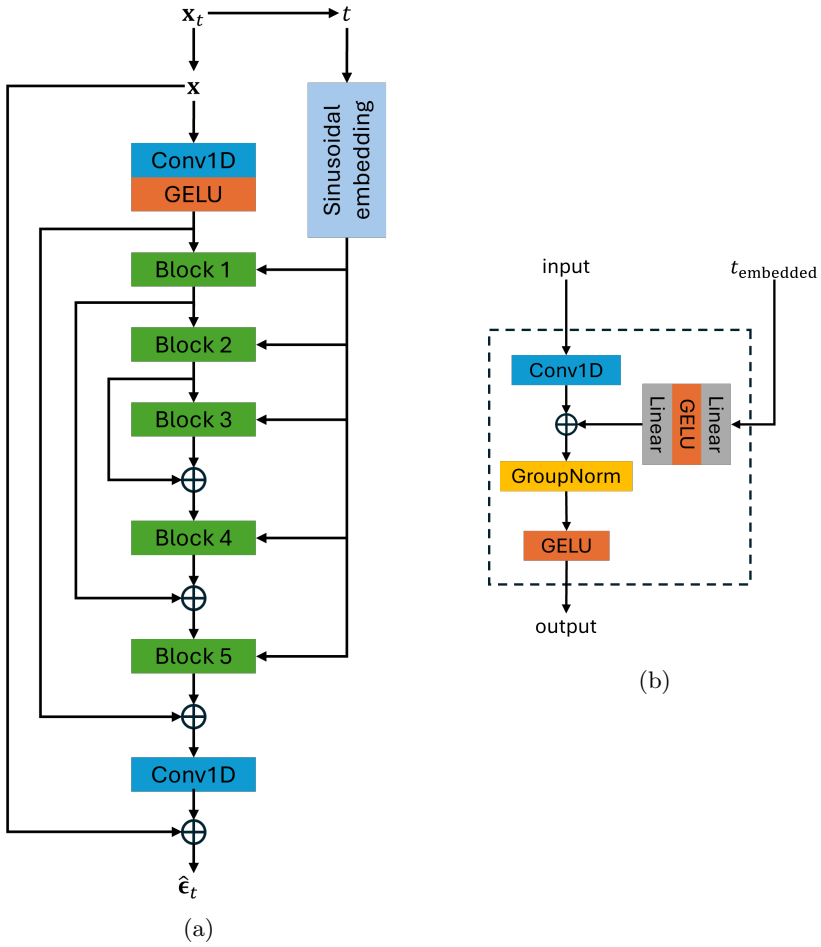

Fig. 2: Deep learning model used as noise predictor $\hat{\boldsymbol{\epsilon}}_\theta$ within DDPM (a): full architecture and (b): single computation block used within the architecture.

than batch normalization [14]. In this work, channels are divided into groups of eight for normalization.

Table 1 outlines the size of convolution and fully connected units across each of the five blocks. The block 1 and 2 expand the number of channels to 256 while block 4 and 5 contract the number of channels back to 64. After processing the input through each of the five blocks, a final convolution unit is used to further reduce the number of channels to one. There are also four connections that enable skipping some of the intermediate blocks to improve training efficiency and to prevent vanishing gradient issues, especially when using deep networks. Each convolution block used in this network has a kernel size of 15, padding of

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

Table 1: Architecture details for the individual blocks used within noise predictor $\hat{\boldsymbol{\epsilon}}_\theta$ model.

Block	Convolution unit		Fully connected unit	
	in channels	out channels	1^{st} linear	2^{nd} linear
Block 1	64	128	128×128	128×128
Block 2	128	256	128×256	256×256
Block 3	256	256	128×256	256×256
Block 4	256	128	128×128	128×128
Block 5	128	64	128×64	64×64

7 and stride of 1. The model is built and trained using PyTorch framework on NVIDIA 3070.

3 Results

This section demonstrates UP using DDPM on two physics-based problems. The training set for DDPM consists of 250 samples for both problems, while 10000 points are sampled from the trained DDPM model. The direct MC simulation is also performed using 10,000 samples. The mean and standard deviation of the random field computed using DDPM samples are compared against the values obtained from direct MC simulation. It is important to note that the problems illustrated in this section are only for demonstrating the proposed method. The added value of DDPM for UP will be more clear in the scenarios that involve large number of uncertain input variables, such as uncertain boundary conditions.

3.1 Flow in a circular pipe

Consider fluid flow in a circular pipe as shown in Figure 3. Under the assumption of incompressible, laminar flow and Newtonian fluid, an analytical expression can be obtained for this 1D flow. The fully developed velocity field can be computed as

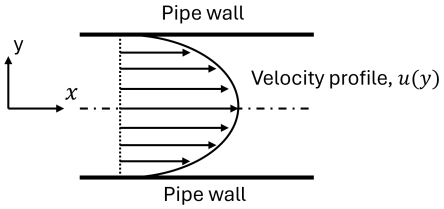

Fig. 3: A fully developed flow in a circular pipe.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

$$u(y) = \frac{G}{2\nu}\left(\frac{d^2}{4} - y^2\right), \tag{11}$$

where $-d/2 \leq y \leq y/2$, G is the pressure gradient applied in the direction of flow, ν is the fluid viscosity, and d is the diameter of the pipe. Since the flow is 1D, the velocity u only varies in y direction. In this work, the fluid viscosity ν is considered to be uncertain, which follows a uniform distribution given by

$$\nu \sim U[7.5 \times 10^{-4}, 2.5 \times 10^{-3}]. \tag{12}$$

This results in an uncertain $u(y)$ field that needs to be quantified. Hence, the goal is to train a DDPM to approximate the distribution of this uncertain field. For this problem, $\mathcal{F}$ is given by (11), $\mathbf{X}$ consists of viscosity ν and $\mathbf{Y}$ is the random velocity field $u(y)$. The $\boldsymbol{\varphi}$ consists of parameters G and d, which are fixed to 0.1 Pa/m and 0.1 m, respectively. For this problem, the random field $u(y)$ consists of 100 linearly spaced collocation points between $-d/2$ and $d/2$.

Figure 4 compares ten $u(y)$ samples generated from the true distribution and the trained DDPM. It should be noted that samples generated from DDPM has similar flow field features as the true samples. Specifically, the $u(y)$ field starts and ends at 0 m/s, which is physically consistent with no-slip wall condition. Moreover, the $u(y)$ samples generated by DDPM are also symmetric about the center line. This points to the fact that DDPM is generating samples that are physically valid.

Figure 5 compares flow field statistics obtained from DDPM samples to that of direct MC samples. Specifically, 10^{th}, 25^{th}, 75^{th}, and 90^{th} percentile are compared, along with mean $u(y)$ field. It can be noted that samples from DDPM yield almost identical statistics as the direct MC samples. This indicates that DDPM is able to learn the true underlying distribution of $u(y)$ field.

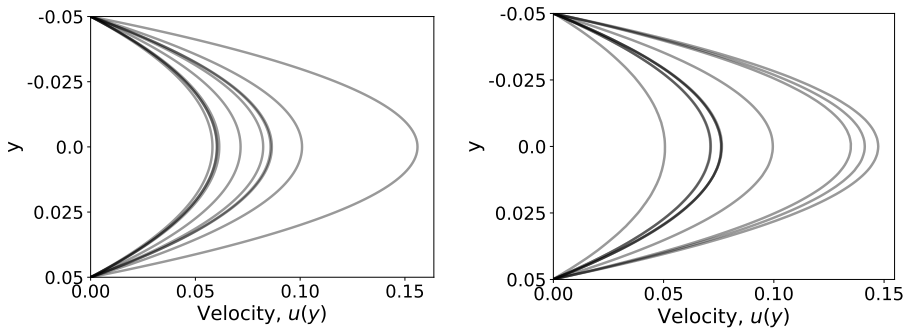

Fig. 4: Comparison of ten randomly generated velocity profile samples (a) from true distribution, and (b) using DDPM.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

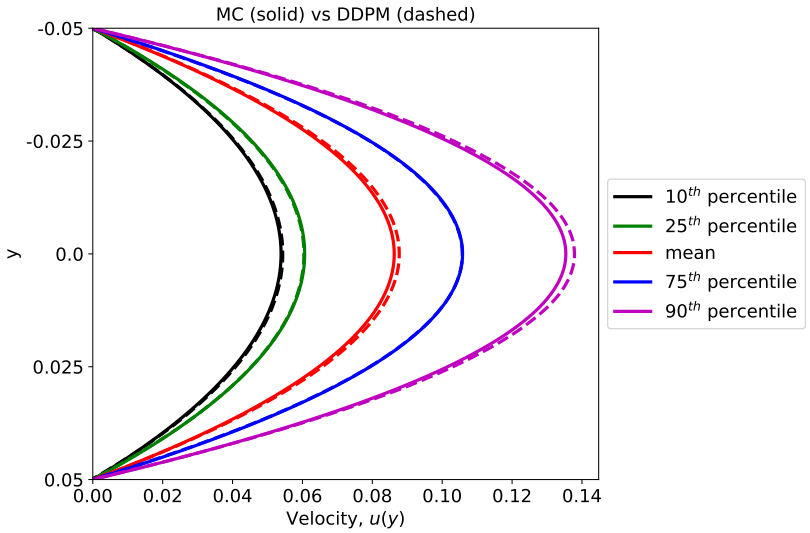

Fig. 5: Comparison of mean and 10^{th}, 25^{th}, 75^{th}, and 90^{th} percentile computed using Monte Carlo and DDPM-based approach

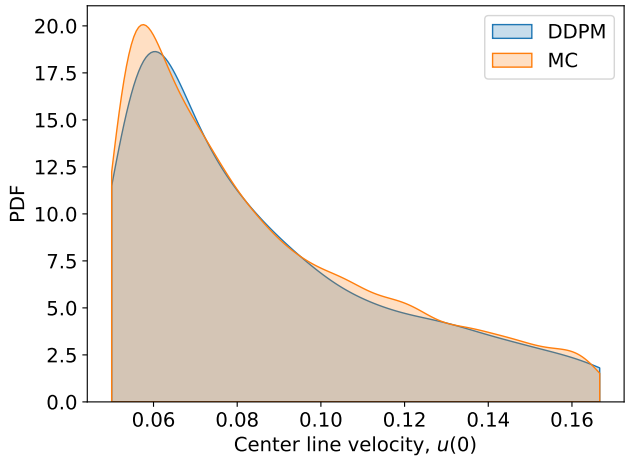

Fig. 6: Comparison of probability density function of the center line velocity, $u(0)$.

Figure 6 compares probability density function (PDF) of the center line velocity $u(0)$ obtained from DDPM and direct MC simulation. There is strong agreement between the PDFs with a slight mismatch around the peak. This shows that DDPM is able to capture the randomness in important flow field features, such as maximum velocity $u_{\max}$. Lastly, Figure 7 illustrates the evolution of a Gaussian noise to $u(y)$ sample within DDPM. At the start of reverse process

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

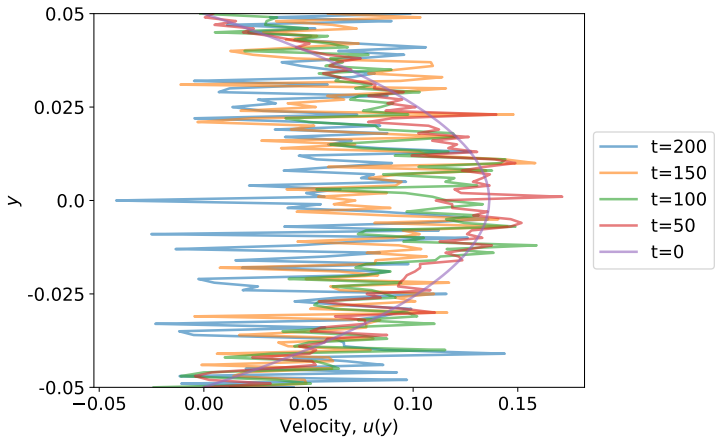

Fig. 7: Evolution of $u(y)$ within DDPM

($t = 200$), the sample is just a Gaussian noise (blue line). As $t \to 0$, the noise is progressively removed to yield a smooth $u(y)$ field. These results indicate that DDPM is able to learn the distribution of $u(y)$ with just 250 samples, and hence, can be used to quantify uncertainty in $u(y)$.

3.2 Converging-diverging nozzle

This problem deals with compressible flow within a conical converging-diverging (CD) nozzle shown in Figure 8. The geometry of the nozzle is governed by the inlet, outlet and throat radius, while the flow is driven by the back pressure p_b. There is no flow in the nozzle when p_b is same as the stagnation pressure p_0.

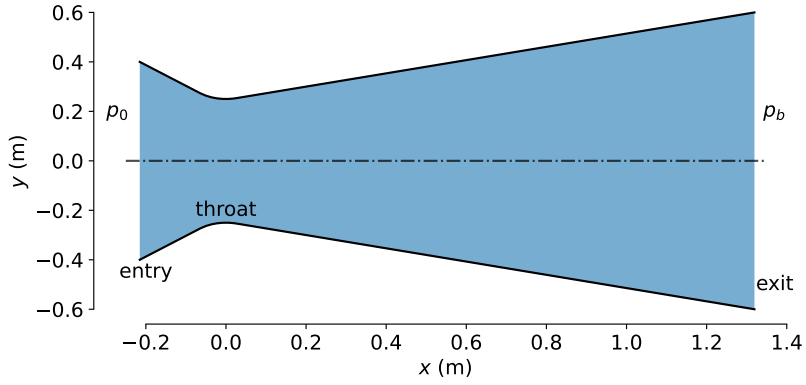

Fig. 8: Converging-diverging nozzle used in this work.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

As p_b decreases beyond a threshold, the flow accelerates to supersonic speeds through the throat and encounters a normal shock-wave in diverging part of the nozzle. This shock wave decelerates the flow to subsonic speeds as it exits the nozzle.

In this work, the ratio of back pressure to stagnation pressure (p_b/p_0) and the ratio of specific heats (γ) are considered to be uncertain. Both these variables are distributed as

$$p_b/p_0 \sim \mathcal{N}(0.4, 0.02^2) \quad \text{and} \quad \gamma \sim U[1.15, 1.35]. \tag{13}$$

This results in an uncertain Mach number $M(x)$ field along the length of the nozzle. The goal is to use the DDPM model to quantify this uncertain distribution of $M(x)$. Note that $M(x)$ is a discontinuous field due to the presence of a shock wave. Moreover, there are regions where $M(x)$ is not affected by the variations in p_b/p_0. This makes it a challenging distribution for the DDPM to model.

The $M(x)$ is governed by isentropic flow relations and normal shock wave equations, which are solved numerically to obtain $M(x)$ for any given condition. This work uses pygasflow[4] package, an open-source Python library for solving various compressible flow problems. The solver uses 114 linearly spaced collocation points from the nozzle inlet till the exit.

In this problem, $\mathcal{F}$ is the solver described earlier. The $\mathbf{X}$ consists of uncertain p_b/p_0 and γ, and $\mathbf{Y}$ is the random Mach field $M(x)$. The $\boldsymbol{\varphi}$ consists of various fixed parameters such as nozzle geometry variables. The inlet, outlet and throat radius is set to 0.4 m, 0.6 m, and 0.25 m, respectively.

Figure 9 compares ten $M(x)$ samples generated from the true distribution and the trained DDPM. It should be noted that samples generated from DDPM has similar flow field features as the true samples. Specifically, the $M(x) = 1$ at

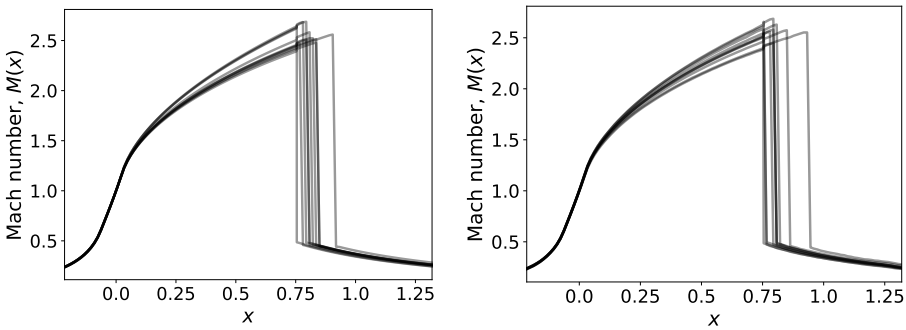

Fig. 9: Comparison of ten randomly generated Mach profile samples (a) from true distribution, and (b) using DDPM.

[4] `https://pygasflow.readthedocs.io`

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

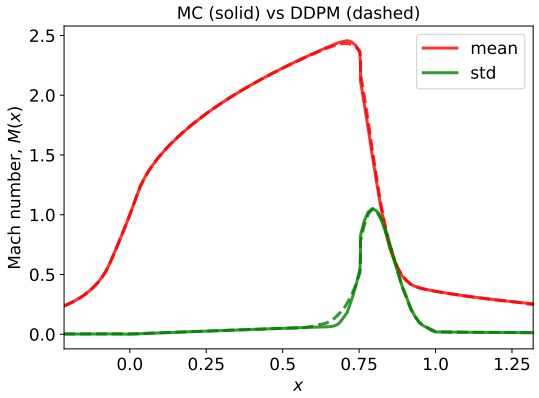

Fig. 10: Comparison of mean and standard deviation of random Mach field computed using Monte Carlo and DDPM.

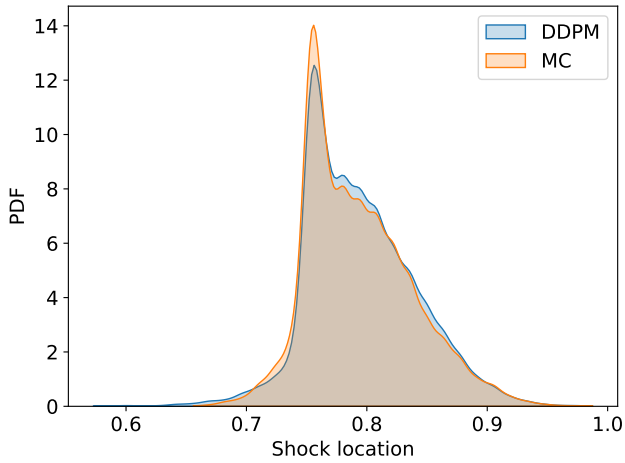

Fig. 11: Probability density function of shock location in the CD nozzle.

the throat of the nozzle, which is physically consistent as Mach number is equal to 1 at throat. Moreover, DDPM is able to generate shock waves without any smoothing, and the flow speed after the shock wave is subsonic. All these points indicate that DDPM is generating samples that are physically valid.

Figure 10 compares flow field statistics from DDPM samples to that of direct MC samples. The mean and standard deviation of $M(x)$ field obtained from DDPM are almost same as the one obtained from direct MC simulation. Moreover, DDPM is able to identify regions in the field that have near-zero variability. This indicates that DDPM is able to learn the discontinuous random flow features of $M(x)$, while taking into account zero-uncertainty regions.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

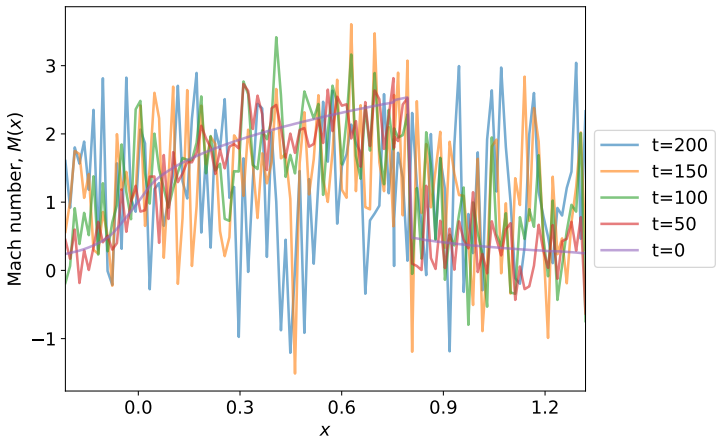

Fig. 12: Generation of a sample from random Mach field using DDPM.

Figure 11 compares PDF of the shock wave location in the nozzle obtained from DDPM and direct MC simulation. The PDF obtained from DDPM is almost identical as the one obtained from MC, but with a slightly lower peak value. This indicates that DDPM is able to capture the randomness in important flow field features, such as the shock wave location.

Similar to the previous problem, Figure 12 illustrates the evolution of a Gaussian noise to a $M(x)$ sample within DDPM. At the start of reverse process ($t = 200$), the sample is just a Gaussian noise (blue line). As $t \to 0$, the sample is progressively denoised to yield a valid $M(x)$ sample. These results indicate that DDPM is able to learn the distribution of $M(x)$ with just 250 samples, and hence, can be used to generate samples from it.

4 Conclusion

This work proposes a novel method for tackling uncertainty propagation (UP) problems and to quantify randomness in output fields using diffusion models. Specifically, the proposed method consists of training a denoising diffusion probabilistic model (DDPM) to learn the underlying distribution governing the output random field. To demonstrate DDPM for UP, this work utilizes two physics-based fluid flow problems that consists of random field outputs induced by uncertain flow parameters. The results highlight that DDPM can learn the distribution of random field, while utilizing only a small number of samples. Hence, it can be a potent tool for UP through computationally expensive scientific models. Going forward, DDPM-based UP approach needs to be tested rigorously on problems involving 2D random fields. Moreover, latest state-of-the-art diffusion models, such as score-matching or flow-matching, can be utilized to further improve the sample efficiency.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10

5 Acknowledgements

This work is supported in part by the Icelandic Research Fund Grant 239858.

References

1. Dhariwal, P., Nichol, A.: Diffusion Models Beat GANs on Image Synthesis. In: Advances in Neural Information Processing Systems. vol. 34, pp. 8780–8794 (2021)
2. Dikshit, A., Koratikere, P., Leifsson, L.T., He, P.: Surrogate-based uncertainty propagation in aerostructural analysis of a wing in subsonic flow. In: AIAA SCITECH 2026 Forum. p. 2398. Orlando, Florida (2026)
3. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). p. 1026–1034 (2015)
4. Ho, J., Jain, A., Abbeel, P.: Denoising Diffusion Probabilistic Models. In: Advances in Neural Information Processing Systems. vol. 33, pp. 6840–6851 (2020)
5. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
6. Kontolati, K., Loukrezis, D., Giovanis, D.G., Vandanapu, L., Shields, M.D.: A survey of unsupervised learning methods for high-dimensional uncertainty quantification in black-box-type problems. Journal of Computational Physics **464**, 111313 (Sep 2022)
7. Koratikere, P., Leifsson, L.: Efficient airfoil geometric uncertainty quantification using neural network models and sequential sampling. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers (2023)
8. Lai, C.H., Song, Y., Kim, D., Mitsufuji, Y., Ermon, S.: The principles of diffusion models. arXiv preprint arXiv:2510.21890 (2025)
9. Lee, S.H., Chen, W.: A comparative study of uncertainty propagation methods for black-box-type problems. Structural and multidisciplinary optimization **37**(3), 239–253 (2009)
10. Prince, S.J.: Understanding Deep Learning. The MIT Press (2023)
11. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: Proceedings of the 32nd International Conference on Machine Learning. vol. 37, pp. 2256–2265. PMLR (07–09 Jul 2015)
12. Tripathy, R., Bilionis, I., Gonzalez, M.: Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. Journal of Computational Physics **321**, 191–223 (2016)
13. Wang, C., Qiang, X., Xu, M., Wu, T.: Recent advances in surrogate modeling methods for uncertainty quantification and propagation. Symmetry **14**(6), 1219 (2022)
14. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19. Munich, Germany (2018)
15. Xiu, D., Karniadakis, G.E.: Modeling uncertainty in flow simulations via generalized polynomial chaos. Journal of computational physics **187**(1), 137–167 (2003)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_10