Journal of Computational Physics 408 (2020) 109281

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Coupling the reduced-order model and the generative model for an importance sampling estimator

Xiaoliang Wan [a,*], Shuangqing Wei [b]

[a] Department of Mathematics and Center for Computation and Technology, Louisiana State University, Baton Rouge 70803, United States of America

[b] Division of Electrical & Computer Engineering, Louisiana State University, Baton Rouge 70803, United States of America

ARTICLE INFO

Article history:
Received 12 April 2019
Received in revised form 13 November 2019
Accepted 23 January 2020
Available online 30 January 2020

Keywords:
Uncertainty quantification
Importance sampling
Reduced-order modeling
Generative model
Machine learning

ABSTRACT

In this work, we develop an importance sampling estimator by coupling the reduced-order model and the generative model in a problem setting of uncertainty quantification. The target is to estimate the probability that the quantity of interest (QoI) in a complex system is beyond a given threshold. To avoid the prohibitive cost of sampling a large scale system, the reduced-order model is usually considered for a trade-off between efficiency and accuracy. However, the Monte Carlo estimator given by the reduced-order model is biased due to the error from dimension reduction. To correct the bias, we still need to sample the fine model. An effective technique to reduce the variance reduction is importance sampling, where we employ the generative model to estimate the distribution of the data from the reduced-order model and use it for the change of measure in the importance sampling estimator. To compensate the approximation errors of the reduced-order model, more data that induce a slightly smaller QoI than the threshold need to be included into the training set. Although the amount of these data can be controlled by a posterior error estimate, redundant data, which may outnumber the effective data, will be kept due to the epistemic uncertainty. To deal with this issue, we introduce a weighted empirical distribution to process the data from the reduced-order model. The generative model is then trained by minimizing the cross entropy between it and the weighted empirical distribution. We also introduce a penalty term into the objective function to deal with the overfitting for more robustness. Numerical results are presented to demonstrate the effectiveness of the proposed methodology.

1. Introduction

Generative modeling has become a central object in modern machine learning. The goal of generative modeling is to model all dependencies within high-dimensional data using a full joint probability density function (PDF), and to generate new samples from the learned distribution. The ability to manipulate the joint PDF enables the probabilistic unsupervised learning of realistic world models. Generative modeling has found a wide range of applications such as image processing, speech synthesis, text analysis, etc. Significant advances have been achieved in the recent development of generative

* Corresponding author.
E-mail addresses: xlwan@lsu.edu (X. Wan), swei@lsu.edu (S. Wei).

https://doi.org/10.1016/j.jcp.2020.109281
0021-9991/

modeling. Typical approaches include variational autoencoders [11], autoregressive models [5,12–14], flow-based generative models [1,2,9], and generative adversarial networks (GANs) [4].

Meanwhile we note that multivariate density estimation is a classical topic in statistics [15], where it shows the equivalent sample size with respect to a dimensionless measure of accuracy will increase at least exponentially with respect to the dimensionality. In contrast to the thousands of dimensions considered in generative modeling, practical applications of nonparametric density estimators in more than three dimensions often suffer a great deal from the curse of dimensionality. Although it is not quite fair to consider a direct comparison between nonparametric density estimators and generative models, where nonparametric density estimators focus on the asymptotic behavior of mean integrated square error while the generative models focus on the learning ability and flexibility, generative models, which can be regarded as parametric models, seems able to provide a very general representation of data like the nonparametric estimator, thanks to the capability of deep neural networks for high-dimensional nonlinear approximation. In this work, we are trying to understand if we are able to adapt the generative modeling into a problem setting of uncertainty quantification as a flexible means to establish communications between two mathematical models through data. In particular, we consider an importance sampling estimator

$$\mathbb{E}_\rho[I_B] = \int \frac{I_B(\boldsymbol{y})\rho(\boldsymbol{y})}{\eta(\boldsymbol{y})}\eta(\boldsymbol{y})d\boldsymbol{y} = \mathbb{E}_\eta[I_B\frac{\rho}{\eta}],$$

where ρ and η are two PDFs, and I_B is an indicator function in terms of the set B. Each sample $\boldsymbol{y}$ may be related to the solution $u(t,\boldsymbol{x},\boldsymbol{Y})$ of a PDE subject to random inputs $\boldsymbol{Y}\in\mathbb{R}^n$, where t and $\boldsymbol{x}$ indicate the time and space variable respectively. The random event B is defined by a functional of $u(t,\boldsymbol{x},\boldsymbol{Y})$, e.g., the L_2 norm on a space-time domain is larger than a prescribed threshold. $\rho(\boldsymbol{y})$ is the PDF of $\boldsymbol{Y}$ and $\eta(\boldsymbol{y})$ is the candidate for the change of measure. We assume that our best a prior knowledge of $\eta(\boldsymbol{y})$ is given by a set of data such that we need to estimate the data distribution first before implementing the importance sampling estimator. We then use a generative model to represent $\eta(\boldsymbol{y})$. The study of generative modeling usually focuses on the minimization of a certain measure on the distance between the model and the data distribution while our main concern is the effectiveness of the importance sampling estimator which can be measured quantitatively by the degree of variance reduction. Due to the overfitting, $\eta(\boldsymbol{y})$ that is closer to the data distribution might not introduce variance reduction. Thus the robustness is an important issue in addition to the dimensionality of $\boldsymbol{y}$. Our problem setting requires an explicit evaluation of the density function, which makes the adaption of some generative models such as GAN and variational autoencode not straightforward. In this work, we will employ the flow-based generative models [2,9], which provide tractable likelihood and exact inference due to the invertible transport map.

We will construct an importance sampling estimator using multi-fidelity models: one fine model and its reduced-order model. The goal is to obtain the probability $\Pr(B)$ or $\mathbb{E}[I_B]$ with respect to the fine model. However, since each sample corresponds to solving a large scale problem, which is time consuming, we want to collect some data from a reduced-order model, and use them to construct $\eta(\boldsymbol{y})$ for the importance sampling on the fine model. To make the strategy practical, we have considered the following two issues: First, with respect to the fine model, there exists noise in the data from the reduced-order model, which means we cannot simply keep the data satisfying B for the reduced-order model. We need to enlarge the data set to tolerate the errors from model reduction. Unfortunately because of the epistemic uncertainty in the errors of the reduced-order model, redundant data, which do not satisfy B for the fine model, might be kept. To alleviate this issue, we have proposed a weighted empirical distribution such that the important data have a larger weight while the less important data have a smaller weight. We then approximate the weighted empirical distribution using a flow-based generative model. Second, the importance sampling estimator may not perform well when the original flow-based generative model is employed. There are two reasons for this issue: one is the overfitting and the other one is the bad conditioning in the original generative model. When the overfitting occurs, less or no variance reduction may be obtained. This implies that extra regularization is needed other than that provided by the stochastic optimization. We will show that incorporating the properties of the problem can provide a much more robust regularization than the general regularization techniques such as early stopping. More specifically, we add a penalty term to balance the minimization of the cross entropy and the fact that the ratio $\frac{I_B\rho}{\eta}$ should be close to a constant for the maximization of variance reduction. Because the flow-based generative model has an explicit density function, such a penalty term can be easily implemented. The bad conditioning of the original generative model may produce outliers with a very small density $\eta(\boldsymbol{y})$ such that $\frac{\rho}{\eta}$ will be vary large. This problem can also be alleviated by our proposed regularization term. A more direct way of resolving this problem is to improve the original flow-based generative model to make it more suitable for scientific computing. We will not focus on this possibility in this work.

This paper is organized as follows. In the next section we specify the problem setting and develop a guiding principle for our methodology. In section 3 we build up the flow-based generative model used in this work. The main numerical strategy is developed in section 4. Some details related to implementation are given in section 5. We present numerical experiments in section 6 followed by a summary section.

2. Problem description

We are interested in simulating the random events given by a partial differential equation (PDE) subject to uncertainty. We present our methodology using the following general mathematical model:

$$\mathcal{L}(u(t,\boldsymbol{x});\boldsymbol{Y})=0, \tag{1}$$

where $\mathcal{L}$ is a space-time differentiation operator, t the time, $\boldsymbol{x}\in\mathbb{R}^d$ the space variable, and $\boldsymbol{Y}\in\mathbb{R}^n$ a n-dimensional random vector. Let $B=\{\boldsymbol{y}|g(u)>0\}$, where $g(\cdot)$ is a functional indicating the Quantity of Interest (QoI). We intend to estimate the following probability

$$\ell=\Pr(\boldsymbol{Y}\in B)=\mathbb{E}[I_B],$$

where $I_B(\cdot)$ is an indicator function such that $I_B(\boldsymbol{y})=1$ if $\boldsymbol{y}\in B$, and 0 otherwise. To make our target problem more specific, we introduce the following two assumptions:

1. The random variable $\boldsymbol{Y}$ can be effectively sampled.
2. The probability $\mathbb{E}[I_B]$ is not too small.

These two assumptions simply mean that we are able to obtain a moderate number of effective samples satisfying $g(u)\geq 0$ by directly sampling $\boldsymbol{Y}$. We can then consider the Monte Carlo estimator:

$$P_{\mathrm{MC}}=\frac{1}{N}\sum_{i=1}^{N}I_B(\boldsymbol{y}^{(i)}). \tag{2}$$

To sample u, equation (1) is usually solved numerically. Let $\mathcal{L}_{h,f}$ and $\mathcal{L}_{h,c}$ indicate a fine and a coarse discretization of $\mathcal{L}$, where h indicates a discretization parameter such as the element size in the finite element method. Let $u_{h,f}(t,\boldsymbol{x},\boldsymbol{Y})$ and $u_{h,c}(t,\boldsymbol{x},\boldsymbol{Y})$ be the two approximate solutions induced by $\mathcal{L}_{h,f}$ and $\mathcal{L}_{h,c}$ respectively. Since each sample of $\boldsymbol{Y}$ corresponds to solving a PDE, it can be very expensive if only $\mathcal{L}_{h,f}$ is employed for sampling. Then $\mathcal{L}_{h,c}$ is often used for variance reduction such that less samples from the fine model are needed to reach a certain accuracy, e.g., the multi-level Monte Carlo method [3]. In this work, we consider a predictor-corrector strategy, which is widely used in scientific computing:

1. **Predictor**: We sample the reduced-order model to obtain the distribution of data satisfying $I_{B_{h,c}}=1$, where $B_{h,c}$ indicates the approximation of B by $\mathcal{L}_{h,c}$.
2. **Corrector**: Note that

$$\{\boldsymbol{y}|g(u_{h,f}\geq 0)\}\cap\{\boldsymbol{y}|g(u_{h,c})<0\}\neq\emptyset.$$

We need to correct the prediction given by the reduced-order model by sampling the fine model.

The reasoning of the predictor-corrector strategy is as follows. The predictor given by the reduced-order model is relatively cheap to sample. Although the Monte Carlo estimator based on the reduced-order model is biased due to the error from dimension reduction, it provides useful information for variance reduction, meaning that the corrector based on the fine model does not require a large number of samples. In the next section, we will give a detailed presentation of the predictor-corrector strategy in the framework of importance sampling.

Remark 1. In this work, we will not take into account the error of $\mathcal{L}_{h,f}$. When we say the reduced-order model induces a biased estimator, the bias is up to the accuracy of the fine model.

Remark 2. Although we refer to $\mathcal{L}_{h,c}$ as a reduced-order model, it can be understood in a more general sense. $\mathcal{L}_{h,c}$ can be any approximation model of $\mathcal{L}$ with a relatively low resolution. It can be regarded as a means to generate data efficiently subject to the approximation error in $u_{h,c}$.

2.1. Importance sampling

Let $\rho(\boldsymbol{y})$ be the probability density function of $\boldsymbol{Y}$. The basic idea of importance sampling is to compute the expectation with respect to another density function $\eta(\boldsymbol{y})$ such as

$$\ell=\int I_B(\boldsymbol{y})\frac{\rho(\boldsymbol{y})}{\eta(\boldsymbol{y})}\eta(\boldsymbol{y})d\boldsymbol{y}=\mathbb{E}_\eta\left[I_B(\boldsymbol{y})\frac{\rho(\boldsymbol{y})}{\eta(\boldsymbol{y})}.\right] \tag{3}$$

The corresponding estimator is

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} I_B(\boldsymbol{y}^{(i)})W(\boldsymbol{y}^{(i)}), \tag{4}$$

where $W(\boldsymbol{y}) = \rho(\boldsymbol{y})/\eta(\boldsymbol{y})$ is the likelihood ratio, and the superscript $*^{(i)}$ is the index for samples. It is well known that the best candidate for the change of measure is

$$\eta^*(\boldsymbol{y}) = \frac{I_B(\boldsymbol{y})\rho(\boldsymbol{y})}{\ell}, \tag{5}$$

i.e., the conditional PDF of $\boldsymbol{y}$ satisfying $I_B(\boldsymbol{y}) = 1$. For this case, we have

$$\frac{I_B(\boldsymbol{y}^{(i)})\rho(\boldsymbol{y}^{(i)})}{\eta^*(\boldsymbol{y}^{(i)})} = \ell,$$

meaning that the variance of this estimator is zero. Since ℓ is unknown, $\eta^*(\boldsymbol{y})$ is only of theoretical importance.

In reality, we usually replace $\eta^*(\boldsymbol{y})$ with an approximate one, which is, among a family of parameterized PDFs, the closest one to the data set $\{\boldsymbol{y}^{(i)}|I_B(\boldsymbol{y}^{(i)}) = 1\}$. In our problem setting, extra difficulties come from the fact that each sample $\boldsymbol{y}^{(i)}$ corresponds to solving a PDE, which can be time consuming. One commonly used strategy to alleviate this difficulty is to take advantage of the reduced-order model to achieve a trade-off between efficiency and accuracy. To estimate η^*, two cases can be considered depending on the source of the data: (1) The data are just from the reduced-order model, or (2) The data are from both the reduced-order model and the fine model. For simplicity, we will only consider the first case in this work, where we need to resolve the following two general issues:

1. How to use the data from the reduced-order model $\mathcal{L}_{h,c}$ to estimate η^*? A straightforward way is to approximate data distribution given by $B_{h,c} = \{\boldsymbol{y}^{(i)}|g(u_{h,c}) \geq 0\}$. The problem of doing this is that the data satisfying $g(u_{h,c}) \geq 0$ may not satisfy $g(u_{h,f}) \geq 0$ due to the approximation errors of model reduction. In other words, η^* is not absolutely continuous to its approximation.
2. How to choose a model $\eta(\boldsymbol{y};\boldsymbol{\theta})$ for the density estimation, where $\boldsymbol{\theta}$ indicates the model parameter. A widely used model is the Gaussian mixture, which can be viewed as a kind of kernel method. It is well known that learning high-dimensional Gaussian mixtures is difficult due to the curse of dimensionality, where the sample size needs to increase exponentially.

2.2. Our general methodology

Corresponding to the aforementioned two general issues, our methodology consists of two parts: 1) data preparation, where we include some extra data that satisfy $g(u_{h,c}) < 0$ and define a weighted empirical distribution, and 2) density estimation, where we resort to deep learning to construct an explicit model $\eta(\boldsymbol{y};\boldsymbol{\theta})$.

Before a detailed presentation of our methodology, we generalize the understanding of $\eta^*(\boldsymbol{y})$ for the change of measure in the importance sampling. The effectiveness of the importance sampling estimator is determined by the variance of the function

$$w(\boldsymbol{Y}) = \frac{I_B(\boldsymbol{Y})\rho(\boldsymbol{Y})}{\eta(\boldsymbol{Y})}. \tag{6}$$

When $w(\boldsymbol{Y})$ provides an unbiased estimator, i.e., $\mathbb{E}_\eta[w] = \mathbb{E}_\rho[I_B]$, the effectiveness of the estimator is determined by the second-order moment of $w(\boldsymbol{Y})$:

$$\mathbb{E}_\eta[w^2] = \int_B \frac{\rho^2}{\eta} d\boldsymbol{y}. \tag{7}$$

Due to the introduction of reduced-order model, we cannot guarantee that all data from the reduced-order model satisfy $g(u_{h,f}) \geq 0$. Instead we can assume that the density estimation will be implemented on a set $\hat{B}$ that is larger than B, i.e., $B \subset \hat{B}$. This means that

$$\int_B \eta(\boldsymbol{y}) d\boldsymbol{y} = \alpha < 1. \tag{8}$$

We now look for the best η which satisfies equation (8), and minimizes the second-order moment of w. In other words, we consider the optimization problem

$$\min_\eta \left[J(\eta) = \int_B \frac{\rho^2}{\eta} d\boldsymbol{y} + \lambda \left(\int_B \eta d\boldsymbol{y} - \alpha \right) \right],$$

where λ is a Lagrange multiplier. Considering the first-order variation, we have

$$\delta J = -\int_B \frac{\rho^2}{\eta^2}\delta\eta d\boldsymbol{y} + \lambda \int_B \delta\eta d\boldsymbol{y},$$

where $\delta\eta$ is a perturbation function. This means that the optimal η satisfies

$$\frac{\rho^2}{\eta^2} = \lambda, \quad \forall \boldsymbol{y} \in B, \tag{9}$$

from which we obtain the minimizer

$$\eta_\alpha^*(\boldsymbol{y}) = \frac{\alpha}{\mathbb{E}[I_B]}\rho(\boldsymbol{y}), \quad \forall \boldsymbol{y} \in B. \tag{10}$$

The value of η_α^* on $\hat{B}\backslash B$ does not affect the performance of η_α^*. The variance of I_B is

$$\mathrm{Var}(I_B) = \mathbb{E}[I_B] - \mathbb{E}[I_B]^2. \tag{11}$$

The variance of $w(\boldsymbol{Y})$ is

$$\mathrm{Var}(w) = \frac{1}{\alpha}\mathbb{E}[I_B]^2 - \mathbb{E}[I_B]^2. \tag{12}$$

Thus, the closer α is to 1, the smaller the variance of w is. When $\alpha = 1$, i.e., $\hat{B} = B$, we have the best scenario with zero variance. Note that $\mathrm{Var}(w) > \mathrm{Var}(I_B)$ if $\alpha < \mathbb{E}[I_B]$.

To this end, we obtain the following two general principles to guide the development of our methodology: 1) η must provide a substantial probability on B, i.e., α should be close to 1; and 2) On B, the ratio between η_α^* and ρ is always a constant even if η_α^* has a larger support than B.

3. Change of measure via generative models

3.1. Flow-based generative models

Density estimation is a difficult problem, especially for high-dimensional data. Many techniques have recently been developed in the framework of machine learning under the term generative modeling. Generative models are usually with likelihood-based methods, such as the autoregressive models [5,12,13], variational autoencoders [11], and flow-based generative models [1,2,9]. A particular case is the generative adversarial networks (GANs) [4], which requires finding a Nash equilibrium of a game. All generative models rely on the ability of deep nets for the nonlinear approximation of high-dimensional mapping. To incorporate the generative modeling into our problem setting, we here pay particular attention to the flow-based generative model. Simply speaking, the flow-based generative model implements a change of variable though an invertible mapping, which can be regarded as a transport map. It has two distinct features: 1) it provides an explicit form of the probability density function (PDF), and 2) it is easy to sample the estimated distribution. Other generative models usually do not have these two features at the same time. For example, GANs do not require an explicit form of the PDF, which makes it very flexible, but not straightforward for our purpose.

Let $\boldsymbol{Y} \in \mathbb{R}^n$ be a random variable associated with the given data. Our target is to estimate the PDF of $\boldsymbol{Y}$ using the available data. Consider another random variable $\boldsymbol{Z} = f(\boldsymbol{Y}) \in \mathbb{R}^n$, where $f(\cdot)$ is a bijection: $f : \boldsymbol{Y} \mapsto \boldsymbol{Z}$. Let $p_{\boldsymbol{Y}}$ and $p_{\boldsymbol{Z}}$ be the PDFs of $\boldsymbol{Y}$ and $\boldsymbol{Z}$, respectively. We have

$$p_{\boldsymbol{Y}}(\boldsymbol{y}) = p_{\boldsymbol{Z}}(f(\boldsymbol{y}))\left|\det \nabla_{\boldsymbol{y}} f\right|. \tag{13}$$

Once a prior distribution $p_{\boldsymbol{Z}}(\boldsymbol{z})$ is specified for $\boldsymbol{Z}$, equation (13) provides a model for the density estimation of $\boldsymbol{Y}$. The key component of this model is the nonlinear mapping $f(\cdot)$. In flow-based generative models, an invertible mapping $f(\cdot)$ is constructed by deep nets. After the density estimation, the samples of $\boldsymbol{Y}$ can be easily generated as $\boldsymbol{Y} = f^{-1}(\boldsymbol{Z})$, thanks to the invertible mapping.

To construct $f(\cdot)$, the main difficulties are twofold: (1) $f(\cdot)$ is highly nonlinear since the prior distribution for $\boldsymbol{Z}$ must be simple enough, and (2) the mapping $f(\cdot)$ is a bijection. Flow-based generative models deal with these difficulties by stacking together a sequence of simple bijections, each of which is a shallow neural network, and the overall mapping is a deep net. Mathematically, the mapping $f(\cdot)$ can be written in a composite form:

$$\boldsymbol{z} = f(\boldsymbol{y}) = f_{[L]} \circ \ldots \circ f_{[1]}(\boldsymbol{y}), \tag{14}$$

where $f_{[i]}$ indicates a coupling layer at stage i. The mapping $f_{[i]}(\cdot)$ is expected to be simple enough such that its inverse and Jacobi matrix can be easily computed. Then given any $\boldsymbol{z}$, we can efficiently compute the inverse

$$\boldsymbol{y} = f^{-1}(\boldsymbol{z}) = f_{[1]}^{-1} \circ \ldots \circ f_{[L]}^{-1}(\boldsymbol{z}). \tag{15}$$

Using the chain rule of differentiation, the determinant of the Jacobian matrix is obtained as

$$|\det \nabla_{\boldsymbol{y}} f| = \prod_{i=1}^{L} |\det \nabla_{\boldsymbol{y}_{[i-1]}} f_{[i]}|, \tag{16}$$

where $\boldsymbol{y}_{[i-1]}$ indicate the intermediate variables with $\boldsymbol{y}_{[0]} = \boldsymbol{y}$ and $\boldsymbol{y}_{[L]} = \boldsymbol{z}$.

One way to define $f_{[i]}$ is given by the real NVP [2]. Consider a partition $\boldsymbol{Y} = (\boldsymbol{Y}_1, \boldsymbol{Y}_2)$ with $\boldsymbol{Y}_1 \in \mathbb{R}^m$ and $\boldsymbol{Y}_2 \in \mathbb{R}^{n-m}$. A simple bijection $f_{[i]}$ is defined as

$$\boldsymbol{z}_1 = \boldsymbol{y}_1, \tag{17}$$

$$\boldsymbol{z}_2 = \boldsymbol{y}_2 \odot \exp(\boldsymbol{s}(\boldsymbol{y}_1)) + \boldsymbol{t}(\boldsymbol{y}_1), \tag{18}$$

where $\boldsymbol{s}$ and $\boldsymbol{t}$ correspond to scaling and translation depending only on $\boldsymbol{y}_1$, and $\odot$ indicates the Hadamard product or component-wise product. Note that only part of the input vector is updated using the information that depends on the rest of the input vector. The inverse of this mapping is also simple:

$$\boldsymbol{y}_1 = \boldsymbol{z}_1, \tag{19}$$

$$\boldsymbol{y}_2 = (\boldsymbol{z}_2 - \boldsymbol{t}(\boldsymbol{y}_1))/\exp(\boldsymbol{s}(\boldsymbol{y}_1)), \tag{20}$$

where the division is component-wise. Note that the mappings $\boldsymbol{s}(\boldsymbol{y}_1)$ and $\boldsymbol{t}(\boldsymbol{y}_1)$ can be arbitrarily complicated, which will be modeled as a neural network (NN), i.e.,

$$(\boldsymbol{s}, \boldsymbol{t}) = \mathrm{NN}(\boldsymbol{y}_1). \tag{21}$$

The simple bijection given by equations (17) and (18) is also referred to as an affine coupling layer [2]. Since only part of the input vector is updated, at least two affine couple layers need to be stacked together to update the whole input vector. The Jacobian matrix induced by one affine coupling layer is lower triangular:

$$\nabla_{\boldsymbol{y}} \boldsymbol{z} = \begin{bmatrix} \mathbf{I} & 0 \\ \nabla_{\boldsymbol{y}_1} \boldsymbol{z}_2 & \mathrm{diag}(\exp(\boldsymbol{s}(\boldsymbol{y}_1))) \end{bmatrix}, \tag{22}$$

whose determinant can be easily computed as

$$\log|\det \nabla_{\boldsymbol{y}} \boldsymbol{z}| = \sum_{i=1}^{n-m} s_i(\boldsymbol{y}_1). \tag{23}$$

Remark 3. Note that $\exp(s_i(\boldsymbol{y}_1)) \in (0, +\infty)$ and $t_i(\boldsymbol{y}_1) \in (-\infty, +\infty)$ in equation (18). It is possible that s_i can be really large or small at a certain sample, which makes the problem ill-posed. This implies that the robustness of the flow-based generative model deserves more attention for scientific computing problems.

3.2. Improve the multi-layer invertible mapping $f(\cdot)$

It is seen that the multi-layer invertible mapping $f(\boldsymbol{y})$ relies on the stacking of some simple coupling layers $f_{[i]}$. For the effectiveness of this strategy, we need to pay attention to several issues.

3.2.1. The depth L

If $\boldsymbol{y}$ is partitioned to two parts, at least two affine coupling layers are needed for a complete modification of $\boldsymbol{y}$. Note that the update in equation (18) is linear in terms of $\boldsymbol{y}_2$, meaning that a large depth L may be needed to obtain a good transport map between $\boldsymbol{Y}$ and $\boldsymbol{Z}$. It usually is enough to define a shallow neural network NN (see equation (21)) for each affine coupling layer since the update given by $f_{[i]}$ is limited by its definition. In this work, we use two fully coupled hidden layers for NN. The capability of $f(\boldsymbol{y})$ mainly relies on the depth L.

3.2.2. The partition of $\boldsymbol{Y}$

The key to maintain the invertibility of the mapping is to partition $\boldsymbol{Y}$, such that each part can be updated in an interweaving way. We have several available options for the partition of $\boldsymbol{Y}$:

1. Fixed partition. This is the choice we are using so far for the presentation, which is also the simplest one. In every affine coupling layer, the first m components are modified or remain unchanged, where we usually let $m = \lfloor n/2 \rfloor$. The drawback of this choice is twofold: (1) we treat the two halves of $\boldsymbol{Y}$ equally although they may not be of the same importance; and (2) The degree of mixing of all components of $\boldsymbol{Y}$ is limited. For example, if $\boldsymbol{y}_1$ is nearly independent of $\boldsymbol{y}_2$, we expect to mix the components of $\boldsymbol{y}_1$ instead of modifying $\boldsymbol{y}_1$ linearly using a function of $\boldsymbol{y}_2$.

2. Random partition. If we do not have a prior knowledge of the importance or independence of each dimension of $\boldsymbol{Y}$, a random partition provides a simple way to increase the correlation between the components of $\boldsymbol{Y}$. The random partition shuffles all the components of $\boldsymbol{y}$ before implementing a fixed partition such that each coupling layer $f_{[i]}$ may have a different partition pattern.
3. Linear transformation of $\boldsymbol{Y}$. We can define a new random variable $\hat{\boldsymbol{Y}} = \mathbf{W}\boldsymbol{Y}$, where $\mathbf{W}$ is a non-singular matrix and can be regarded as a rotation between two coordinate systems. We then consider a fixed partition of $\hat{\boldsymbol{Y}}$ instead of $\boldsymbol{Y}$. Furthermore, $\mathbf{W}$ can be included into the trainable parameters. In other words, although we do not know the importance of each dimension of $\boldsymbol{Y}$ for the desired nonlinear mapping, we can let the algorithm learn from the data a better coordinate system for the fixed partition. In [9] a similar strategy was used to improve the performance of real NVP for image processing.

In this work, we mainly stick to the fixed partition of $\boldsymbol{Y}$ to test the effectiveness of our methodology for importance sampling. Once the effectiveness is verified, the second and third options can be considered for further improvement.

3.2.3. Scale and bias layer

It is well known that batch normalization can improve the propagation of training signal in a deep net. Let $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}^2$ be the mean and variance estimated from the mini batch [7]. The batch normalization algorithm includes two steps: the first step defines for each layer of the neural network the following normalization

$$y_i \leftarrow \frac{y_i - \tilde{\mu}_i}{\sqrt{\tilde{\sigma}_i^2 + \epsilon}}, \quad i = 1, \ldots, n, \tag{24}$$

and the second step refines the previous step by a trainable scale-shift operation:

$$\hat{\boldsymbol{y}} = \gamma\, \boldsymbol{y} + \boldsymbol{\beta}. \tag{25}$$

When the size of minibatch is small, batch normalization (24) becomes less effective due to the noise in the computation of $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$. A compromise of the two steps in the batch normalization algorithm is proposed in [9], i.e.,

$$\hat{\boldsymbol{y}} = \boldsymbol{a} \odot \boldsymbol{y} + \boldsymbol{b}, \tag{26}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are trainable, and initialized by $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$ associated with the initial data. After the initialization, $\boldsymbol{a}$ and $\boldsymbol{b}$ will be treated as regular trainable parameters that are independent of the data. In this work, we simplify the procedure (26) further by only applying the scale and bias layer given by equation (26) to the input of $f_{[i]}$. In other words, we do not apply any normalization techniques to the shallow neural network for $\boldsymbol{s}(\cdot)$ and $\boldsymbol{t}(\cdot)$. The only motivation of this simplification is to study the robustness of the generative model in our problem setting.

Combining the above discussions, we can refine the coupling layer $f_{[i]}$ as shown in Fig. 1, where the input of an affine coupling layer is partitioned in a certain way after a scale and shift layer is implemented.

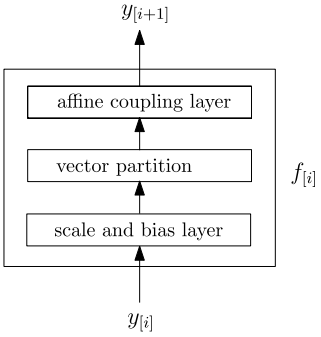

Fig. 1. The diagram of a general coupling layer $f_{[i]}$.

4. Cross entropy by the weighted empirical distribution

4.1. Likelihood and cross entropy

The generative model will be trained by maximizing the likelihood. In terms of data, the minimum cross entropy is the same as the maximum likelihood. Consider the set of data $\{\boldsymbol{y}^{(i)}\}_{i=1}^N$ from the distribution of $\boldsymbol{Y}$. Specifying a distribution for $\boldsymbol{Z}$ in equation (13), we obtain a model for the PDF of $\boldsymbol{Y}$. Let θ be the parameter from the definition of the mapping $f(\boldsymbol{y})$. The maximum likelihood estimator of θ is

$$\theta_{\mathrm{ML}} = \mathrm{argmax}_\theta \prod_{i=1}^{N} p_{\boldsymbol{Y}}(\boldsymbol{y}^{(i)};\theta) = \mathrm{argmax}_\theta \sum_{i=1}^{N} \log p_{\boldsymbol{Y}}(\boldsymbol{y}^{(i)};\theta). \tag{27}$$

Let $\mu_{\mathrm{data}}(\boldsymbol{y})$ be the empirical distribution of the data. Multiplying $1/N$ to the right-hand side of the above equation, the maximum likelihood estimator can also be regarded as

$$\theta_{\mathrm{ML}} = \mathrm{argmax}_\theta \mathbb{E}_{\mu_{\mathrm{data}}}[\log p_{\boldsymbol{Y}}(\boldsymbol{y};\theta)], \tag{28}$$

where the expectation is with respect to μ_{data}. To estimate θ, we can also minimize the distance between μ_{data} and $\mu_{\boldsymbol{Y}}$ using the Kullback-Leibler (KL) divergence:

$$D_{\mathrm{KL}}(\mu_{\mathrm{data}}||\mu_{\boldsymbol{Y}}) = \mathbb{E}_{\mu_{\mathrm{data}}}\left[\log \frac{d\mu_{\mathrm{data}}}{d\mu_{\boldsymbol{Y}}}\right] = H(\mu_{\mathrm{data}}, \mu_{\boldsymbol{Y}}) - H(\mu_{\mathrm{data}}) \tag{29}$$

where $\mu_{\boldsymbol{Y}}(d\boldsymbol{y}) = p_{\boldsymbol{Y}} d\boldsymbol{y}$, $H(\mu_{\mathrm{data}}, \mu_{\boldsymbol{Y}})$ is the cross entropy of μ_{data} and $\mu_{\boldsymbol{Y}}$, and $H(\mu_{\mathrm{data}})$ is the entropy of the empirical distribution solely determined by the data. It is seen that to minimize the KL divergence, we only need to minimize the cross entropy

$$H(\mu_{\mathrm{data}}, \mu_{\boldsymbol{Y}}) = -\mathbb{E}_{\mu_{\mathrm{data}}}[\log p_{\boldsymbol{Y}}] = -\frac{1}{N}\sum_{i=1}^{N} \log p_{\boldsymbol{Y}}(\boldsymbol{y}^{(i)};\theta), \tag{30}$$

because the entropy $H(\mu_{\mathrm{data}})$ only depends on data. Comparing equations (28) and (30), we know that maximizing the maximum likelihood is equivalent to minimizing the cross entropy. Let us assume that the components of $\boldsymbol{Z}$ are i.i.d. normal random variables. We then have

$$\begin{aligned}\log p_{\boldsymbol{Y}}(\boldsymbol{y}) &= \log|\det \nabla_{\boldsymbol{y}} f| - \frac{1}{2}\sum_{i=1}^{n} z_i^2(\boldsymbol{y}) - d\log\sqrt{2\pi} \\ &= \sum_{i=1}^{L} \log|\det \nabla_{\boldsymbol{y}_{[i-1]}} f_{[i]}| - \frac{1}{2}\sum_{i=1}^{n} z_i^2(\boldsymbol{y}) - d\log\sqrt{2\pi}.\end{aligned}$$

4.2. Weighted empirical distribution

We still consider the set of data $\{\boldsymbol{y}^{(i)}\}_{i=1}^N$ from the distribution $\mu_{\boldsymbol{Y}}$ of $\boldsymbol{Y}$. The empirical measure μ_N associated with the data set is defined as

$$\mu_N(A) = \frac{1}{N}\sum_{i=1}^{N} I_A(\boldsymbol{y}^{(i)}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{y}^{(i)}}(A), \tag{31}$$

where $\delta_{\boldsymbol{y}}$ is the Dirac measure. We also define a weighted version of μ_N as follows:

$$\hat{\mu}_N(A) = \sum_{i=1}^{N} w_i \delta_{\boldsymbol{y}^{(i)}}(A), \tag{32}$$

with $\sum_{i=1}^N w_i = 1$. It recovers the empirical measure when $w_i = \frac{1}{N}$. For the empirical measure, each sample in the data set is equally important in the sense of the law of large numbers, since all samples have the same weight $1/N$ and are obtained independently. However, in reality we are often more interested in the information of $\boldsymbol{Y}$ that satisfies a certain constraint. A simple and flexible way to incorporate constraints into the data is to associate the data with varying weights.

Let us consider a simple scenario to illustrate the weighted empirical measure. We partition data set $\{\boldsymbol{y}^{(i)}\}_{i=1}^N = \{\boldsymbol{y}^{(i)}\}_{i\in\mathcal{I}_1} \cup \{\boldsymbol{y}^{(i)}\}_{i\in\mathcal{I}_2}$ with $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$, where $\mathcal{I}_i$ indicates an index set with $i = 1, 2$. We let

$$w_i = \pi_1,\ \forall i \in \mathcal{I}_1 \quad \text{and} \quad w_i = \pi_2,\ \forall i \in \mathcal{I}_2,$$

where $\pi_1, \pi_2 \geq 0$ are two constants, satisfying $N_1\pi_1 + N_2\pi_2 = 1$ with N_i being the cardinality of $\mathcal{I}_i$, $i = 1, 2$. We expect to emphasize the information given by the data set $\{\boldsymbol{y}^{(i)}\}_{i\in\mathcal{I}_1}$ by increasing the value of π_1. For simplicity, we assume that $\{\boldsymbol{y}^{(i)}\}_{i\in\mathcal{I}_1} \subset A \subset \mathbb{R}^n$, and $\{\boldsymbol{y}^{(i)}\}_{i\in\mathcal{I}_2} \subset A^c$ with A^c being the complement of A. Let

$$\rho_1(\boldsymbol{y}) = \frac{\rho_{\boldsymbol{Y}}(\boldsymbol{y})}{\mathbb{E}_{\mu_{\boldsymbol{Y}}}[I_A]}, \quad \rho_2(\boldsymbol{y}) = \frac{\rho_{\boldsymbol{Y}}(\boldsymbol{y})}{\mathbb{E}_{\mu_{\boldsymbol{Y}}}[I_{A^c}]}$$

be the two conditional PDFs. We then seek a PDF of the form

$$\rho_w(\boldsymbol{y};\gamma) = \gamma\rho_1(\boldsymbol{y}) + (1-\gamma)\rho_2(\boldsymbol{y}), \tag{33}$$

that is closest to the weighted measure $\hat{\mu}_N$ in terms of the KL divergence, where $0 < \gamma < 1$. For the given data set, minimizing the KL divergence is equivalent to minimizing the cross entropy

$$\begin{aligned} H(\hat{\mu}_N, \eta) &= \mathbb{E}_{\hat{\mu}_N}[\log \rho_w(\boldsymbol{y})] \\ &= \sum_{i\in\mathcal{I}_1} \pi_1 \log(\rho_w(\boldsymbol{y}^{(i)};\gamma)) + \sum_{i\in\mathcal{I}_2} \pi_2 \log(\rho_w(\boldsymbol{y}^{(i)};\gamma)), \end{aligned} \tag{34}$$

where $\eta(d\boldsymbol{y}) = \rho_w d\boldsymbol{y}$. Then $\partial_\gamma H = 0$ yields that

$$\frac{N_1\pi_1}{\gamma} - \frac{N_2\pi_2}{1-\gamma} = 0 \quad \Rightarrow \quad \gamma = N_1\pi_1. \tag{35}$$

When $\pi_1 = 1/N$ as in the empirical distribution, $\gamma = \frac{N_1}{N} \approx \mathbb{E}_{\mu_{\boldsymbol{Y}}}[I_A]$. It is seen that if we increase the weights for the data in $\{\boldsymbol{y}^{(i)}\}_{i\in\mathcal{I}_1}$, the corresponding PDF $\rho_w(\boldsymbol{y})$ will increase the probability of taking values in A, compared to the PDF $\rho_{\boldsymbol{Y}}(\boldsymbol{y})$.

When considering the weighted empirical distribution, we only need a slight modification of the objective function for the minimization of the cross entropy, where equation (30) becomes

$$H(\hat{\mu}_N, \mu_{\boldsymbol{Y}}) = -\mathbb{E}_{\hat{\mu}_N}[\log p_{\boldsymbol{Y}}] = -\sum_{i=1}^{N} w_i \log p_{\boldsymbol{Y}}(\boldsymbol{y}^{(i)};\theta), \tag{36}$$

which corresponds to the maximization of a weighted likelihood:

$$\prod_{i=1}^{N} p_{\boldsymbol{Y}}^{\gamma_i}(\boldsymbol{y}^{(i)};\theta) \tag{37}$$

with $\gamma_i = Nw_i$.

4.3. Weight the data given by the reduced-order model

Recall that the optimal choice for the change of measure in importance sampling is

$$\eta^*(\boldsymbol{y}) = \frac{I_B(\boldsymbol{y})\rho(\boldsymbol{y})}{\ell}.$$

Sampling the reduced-order model, a straightforward approximation of $\eta^*(\boldsymbol{y})$ is

$$\eta^*_{h,c}(\boldsymbol{y}) = \frac{I_{B_{h,c}}(\boldsymbol{y})\rho(\boldsymbol{y})}{\ell_{h,c}}, \tag{38}$$

where $\ell_{h,c} = \mathbb{E}[I_{B_{h,c}}]$. Due to the errors induced by the model reduction, $\eta^*(\boldsymbol{y})$ is not absolutely continuous with respect to $\eta^*_{h,c}(\boldsymbol{y})$. More specifically, when $I_{B_{h,c}} = 0$ or $\eta^*_{h,c}(\boldsymbol{y}) = 0$, it is possible that $I_B = 1$, i.e., $\eta^*(\boldsymbol{y}) > 0$. If $\eta^*_{h,c}(\boldsymbol{y})$ is used for importance sampling, the estimation will be obviously biased, although the convergence can still be reached as the numerical discretization of u is refined. An easy way to fix this problem is to enlarge the support of $\eta^*_{h,c}$ by incorporating the error estimate of $g(u_{h,c})$. Note that for any $\boldsymbol{y}$, we have

$$g(u_{h,c}) = g(u) + \left\langle \frac{\delta g}{\delta u}, u_{h,c} - u \right\rangle + O(\|u_{h,c} - u\|^2),$$

where $\frac{\delta g}{\delta u}$ indicates the functional derivative and $\langle\cdot,\cdot\rangle$ the inner product in the physical space. The first-order variation of $g(u)$ in terms of $u - u_{h,c}$ yields the leading term in the error of $g(u_{h,c})$. Instead of $g(u_{h,c}) = 0$, we can obtain a better guess of $g(u) = 0$ using

$$g(u_{h,c}) \approx 0 + \left\langle \frac{\delta g}{\delta u}, u_{h,c} - u \right\rangle,$$

which is possibly smaller than 0. When sampling the reduced-order model, we need to keep the data satisfying

$$g(u_{h,c}) \geq -\left| \left\langle \frac{\delta g}{\delta u}, u_{h,c} - u \right\rangle \right|, \tag{39}$$

such that we will not reject the data satisfying $g(u_{h,c}) < 0$ while $g(u) \geq 0$. In reality, the error of $g(u_{h,c})$ can be estimated by a posterior error estimate techniques, which has a general form

$$|g(u_{h,c}) - g(u)| \le C_{\boldsymbol{y}} h^m, \tag{40}$$

where $C_{\boldsymbol{y}}$ is a positive constant depending on $\boldsymbol{y}$, and m is an index indicating the accuracy of the reduced-order model. Instead of using equation (39), we can use

$$g(u_{h,c}) \ge -Ch^m \tag{41}$$

as the acceptance criterion of data, where C is a positive constant chosen according to $C_{\boldsymbol{y}^{(i)}}$. Unfortunately, by doing this, we often accept a lot of redundant data. We rewrite

$$g(u_{h,c}) = g(u) + \epsilon(\boldsymbol{y})$$

and look at the discrepancy between $I_{\{g(u_{h,c})\ge 0\}}$ and $I_{\{g(u)\ge 0\}}$. Note that $\{g(u_{h,c}) \ge 0\} = \{g(u) \ge -\epsilon(\boldsymbol{y})\}$. If $\epsilon \ge 0$, $\{g(u) \ge 0\} \subseteq \{g(u_{h,c}) \ge 0\}$; if $\epsilon \le 0$, $\{g(u_{h,c}) \ge 0\} \subseteq \{g(u) \ge 0\}$. Thus the information missed by $I_{\{g(u_{h,c})\ge 0\}}$ is that $\{0 \le g(u) < -\epsilon(\boldsymbol{y})\}$ subject to the condition that $\epsilon(\boldsymbol{y}) \le 0$. In terms of u_h, what is missing is that $\{\epsilon(\boldsymbol{y}) \le g(u_{h,c}) < 0\}$ when $\epsilon(\boldsymbol{y}) \le 0$. Then the following data included by equation (41), are unnecessary:

$$\begin{cases} \{g(u_{h,c}) < 0\}, & \text{if } \epsilon(\boldsymbol{y}) \ge 0, \\ \{-Ch^m \le g(u_{h,c}) < \epsilon(\boldsymbol{y})\}, & \text{if } \epsilon(\boldsymbol{y}) \le 0. \end{cases}$$

The portion of the redundant data in $\{-Ch^m \le g(u_{h,c}) < 0\}$ is

$$\begin{aligned} &\frac{\Pr(\{-Ch^m \le g(u_{h,c}) < 0\})\Pr(\epsilon \ge 0) + \Pr(\{-Ch^m \le g(u_h) < \epsilon\})\Pr(\epsilon \le 0)}{\Pr(\{-Ch^m \le g(u_{h,c}) < 0\})} \\ =&\Pr(\epsilon \ge 0) + \frac{\Pr(\{-Ch^m \le g(u_{h,c}) < \epsilon\})\Pr(\epsilon \le 0)}{\Pr(\{-Ch^m \le g(u_{h,c}) < 0\})} \\ \approx&\frac{1}{2} + \frac{1}{2}\frac{\Pr(\{-Ch^m \le g(u_{h,c}) < \epsilon\})}{\Pr(\{-Ch^m \le g(u_{h,c}) < 0\})}, \end{aligned}$$

where we assume that $\Pr(\epsilon \le 0) \approx \Pr(\epsilon \ge 0) \approx \frac{1}{2}$. In other words, at least 50% of the data in $\{-Ch^m \le g(u_{h,c}) < 0\}$ are not necessary, and if the a posterior error estimate is not tight, most of the data are redundant. If $\mathbb{E}[I_{\{g(u_{h,c})>0\}}]$ is relatively small, the scenario is worse since the probability induced by the unnecessary data might be larger than $\mathbb{E}[I_{\{g(u_{h,c})>0\}}]$. For this case, most of the data may be nothing but pollution (see the example in section 6.3) in terms of the approximation of $\eta^*(\boldsymbol{y})$. According to equation (12), a large amount redundant data implies a small α which makes it difficult to achieve variance reduction.

To deal with this issue, we will adjust the weights of the data such that the undesired data do not contribute too much in the empirical distribution. Following is our plan to weight the data from the reduced-order model:

- All data points satisfying $g(u_{h,c}) \ge 0$ share the same weight, which mimics equation (9).
- For the data points satisfying $g(u_{h,c}) \in [-Ch^m, 0]$, the weight decreases exponentially as $g(u_{h,c})$ decreases away from 0, as illustrated in Fig. 2. We will use a half-normal distribution in terms of $g(u_{h,c})$ to weight the data.

More details about the implementation will be given in section 5.

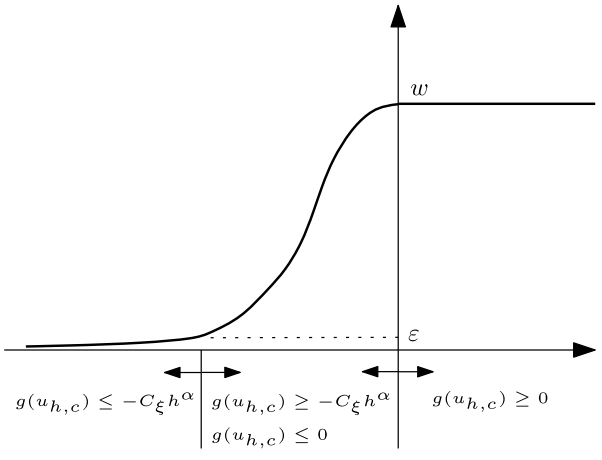

Fig. 2. The diagram of the weight distribution for the data points from sampling the coarse model.

4.4. A penalty term

Let $\eta^*_{h,c}(\boldsymbol{y})$ be the estimated PDF using the weighted data from the reduced-order model. If the generative model is overly complex or the size of data set for training is not large enough, we need to pay particular attention to the overfitting. Many general techniques such as early stopping have been developed in machine learning [18]. We here focus on regularization related to our problem setting. If the conditional PDF $\eta^*(\boldsymbol{y})$ can be well approximated, we should expect that

$$w_{h,c}(\boldsymbol{y}) = \frac{I_B(\boldsymbol{y})\rho(\boldsymbol{y})}{\eta^*_{h,c}(\boldsymbol{y})} \approx C, \tag{42}$$

or

$$\nabla_{\boldsymbol{y}} \log w_{h,c}(\boldsymbol{y}) \approx 0, \tag{43}$$

where C is a positive constant. If the overfitting is not a concern, equation (42) is a natural result given by the density estimation. However, when the overfitting occurs, the minimization of the cross entropy $H(\mu_{\text{data}}, \mu_{\boldsymbol{Y}})$ may yield an approximate distribution such that $w_{h,c}$ has a large standard deviation, which means that the important sampling estimator based on $\eta^*_{h,c}$ may fail to induce variance reduction.

To increase the robustness of the algorithm, we want to balance the minimization of the cross entropy and the condition (42). A convenient way to do this is to add a penalty term

$$\begin{aligned} &\beta \mathbb{E}_{p_{\boldsymbol{Y}}}\left[\left|\nabla_{\boldsymbol{y}} \log w_{h,c}(\boldsymbol{y})\right|^2\right]^{1/2} \\ =&\beta\left(\int \left|\rho^{-1}\nabla\rho - p_{\boldsymbol{Y}}^{-1}\nabla p_{\boldsymbol{Y}}\right|^2 p_{\boldsymbol{Y}} d\boldsymbol{y}\right)^{1/2}, \end{aligned} \tag{44}$$

into the objective function, where β is a penalty parameter. The term $\nabla p_{\boldsymbol{Y}}$ in the integrand provides an H_1 regularization of the objective function. Note that the condition $\boldsymbol{y} \in B$ is determined by the fine model, which is unknown. In reality, we compute the penalty term with respect to the weighted empirical distribution, i.e.,

$$\beta \mathbb{E}_{\hat{\mu}_N}\left[|\nabla_{\boldsymbol{y}} \log w_{h,c}(\boldsymbol{y})|^2\right]^{1/2}. \tag{45}$$

To this end, we have the final objective function for training the generative model as

$$H(\hat{\mu}_N, \mu_{\boldsymbol{Y}}) + \beta \mathbb{E}_{\hat{\mu}_N}\left[|\nabla_{\boldsymbol{y}} \log w_{h,c}(\boldsymbol{y})|^2\right]^{1/2}, \tag{46}$$

where $\mu_{\boldsymbol{Y}}(d\boldsymbol{y}) = p_{\boldsymbol{Y}} d\boldsymbol{y}$.

5. Implementation

We sample $\boldsymbol{Y}$ to obtain $\{\boldsymbol{y}^{(i)}\}_{i=1}^M$. For each $\boldsymbol{y}^{(i)}$, we solve a PDE to obtain $u_{h,c}(\boldsymbol{y}^{(i)})$, and compute an error estimate $\epsilon_{h,c}(\boldsymbol{y}^{(i)})$ of $g(u_{h,c}(\boldsymbol{y}^{(i)}))$. Let $g_{h,c}(\boldsymbol{y}) = g(u_{h,c}(\boldsymbol{y}))$. We organize the data as $\{(\boldsymbol{y}^{(i)}, \epsilon_{h,c}(\boldsymbol{y}^{(i)}), g_{h,c}(\boldsymbol{y}^{(i)}))\}_{i=1}^M$. Let

$$\epsilon^-_{\max} = \max_i \left|\epsilon_{h,c}(\boldsymbol{y}^{(i)}) I_{\{g_{h,c}(\boldsymbol{y}^{(i)})<0\}}\right|. \tag{47}$$

We will keep the data $\{(\boldsymbol{y}^{(i)}, \epsilon_{h,c}(\boldsymbol{y}^{(i)}), g_{h,c}(\boldsymbol{y}^{(i)}))\}_{i=1}^N$, where $g_{h,c}(\boldsymbol{y}^{(i)}) \geq -\epsilon^-_{\max}$. We then use the half-normal distribution

$$f_\tau(\tau;\sigma) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{\tau^2}{2\sigma^2}\right), \quad z \geq 0$$

to fit the data $\tau^{(i)} = g_{h,c}(\boldsymbol{y}^{(i)})$ satisfying $-\epsilon^-_{\max} \leq g_{h,c}(\boldsymbol{y}^{(i)}) < 0$. For the sample $\boldsymbol{y}^{(i)}$, we associate a weight

$$w_i = \begin{cases} c_1 f_\tau(\tau^{(i)}), & \text{if } \tau^{(i)} < 0, \\ c_2, & \text{if } \tau^{(i)} \geq 0, \end{cases} \tag{48}$$

where c_1 and c_2 are two positive constants. Let N_+ be the number of $\boldsymbol{y}^{(i)}$ satisfying $g_{h,c}(\boldsymbol{y}^{(i)}) \geq 0$. We determine c_1, c_2 and σ using the following relations:

$$\begin{cases} N_+ c_2 &= \hat{\alpha}, \\ c_1 \sum_{i=1}^{N-N_+} f_\tau(\tau^{(i)}) &= 1-\hat{\alpha}, \\ c_1 \frac{\sqrt{2}}{\sigma\sqrt{\pi}} &= c_2, \end{cases} \tag{49}$$

where $0 < \hat{\alpha} < 1$. We assign uniform weights to the data $\{g_{h,c}(\boldsymbol{y}^{(i)}) \geq 0\}$, whose probability from the weighted empirical distribution is $\hat{\alpha}$. The parameter $\hat{\alpha}$ has a similar meaning as α in equation (8), which will be prescribed such that the set $\{g_{h,c}(\boldsymbol{y}^{(i)}) \geq 0\}$ has a substantial probability. For example, $\hat{\alpha} = 0.85$ is used in our numerical experiments. The data $\{g_{h,c}(\boldsymbol{y}^{(i)}) < 0\}$ has a probability $1 - \hat{\alpha}$, where the weight decays exponentially as the value $|g_{h,c}(\boldsymbol{y})|$ increases. The third equation can be regarded as a continuity condition, meaning that weight should be continuous when crossing the interface $g_{h,c}(\boldsymbol{y}) = 0$. It is seen that c_2 can be easily obtained from the first equation. From the third equation, we have $c_1 = \frac{c_2\sigma\sqrt{\pi}}{\sqrt{2}}$, which simplifies the second equation as

$$\sum_{i=1}^{N-N_+} c_2 \exp\left(-\frac{(\tau^{(i)})^2}{2\sigma^2}\right) = 1 - \theta.$$

Note that the left-hand side is an increasing function with respect to $\sigma \in (0, +\infty)$, meaning there exists a unique $\sigma \in (0, +\infty)$ satisfying the above equation. Considering $\sigma = c_3 \max_i |g_{h,c}(\boldsymbol{y}^{(i)})|$ with $c_3 > 0$, we have

$$\sum_{i=1}^{N-N_+} c_2 \exp\left(-\frac{(\tau^{(i)})^2}{2\sigma^2}\right) > \theta\left(\frac{N}{N_+} - 1\right)\exp\left(-\frac{1}{2c_3^2}\right).$$

Letting

$$\theta\left(\frac{N}{N_+} - 1\right)\exp\left(-\frac{1}{2c_3^2}\right) = (1 - \theta),$$

i.e.,

$$c_3 = \left(-0.5\left(\log\frac{1-\theta}{Nc_2 - \theta}\right)^{-1}\right)^{1/2},$$

we have the root located in $[0, c_3 \max_i |g_{h,c}(\boldsymbol{y}^{(i)})|]$, which can be computed numerically by a root-finding algorithm.

Another implement issue is related to the stochastic optimization. For unweighted data, a commonly used strategy in stochastic optimization is to split the uniformly shuffled training samples into mini-batches. For the weighted data, a uniform shuffle is obviously not optimal. We then generate mini-batches in a way that is more consistent with the distribution of the weights. We partition the interval $[-\epsilon_{\max}^-, 0] = \cup_{k=1}^K e_k$ uniformly into K disjoint sub-intervals e_k. Let $e_{K+1} = [0, \infty)$. We then group all the training samples as

$$S_k = \{\boldsymbol{y}^{(i)} | g_{h,c}(\boldsymbol{y}^{(i)}) \in e_k\}, \quad k = 1, \ldots, K+1. \tag{50}$$

We will shuffle the training samples in S_k uniformly before we split each S_k to a certain number of batches. We pick one batch in each S_k to assemble the training mini-batch for each iteration step of the stochastic optimization. This way, the data in the mini-batch are distributed similarly to the weighted empirical distribution.

Once the generative model $p_{\boldsymbol{Y}}(\boldsymbol{y})$ is trained, we use it to construct an importance sampling estimator for the fine model

$$\begin{aligned}\ell = \int I_{\{g(u_{h,f})\geq 0\}}\rho(\boldsymbol{y})d\boldsymbol{y} &= \mathbb{E}_{p_{\boldsymbol{Y}}}\left[I_{\{g(u_{h,f})\geq 0\}}\frac{\rho(\boldsymbol{y})}{p_{\boldsymbol{Y}}(\boldsymbol{y})}\right] \\ &= \mathbb{E}_{p_{\boldsymbol{Z}}}\left[I_{\{g(u_{h,f})\geq 0\}}\frac{\rho(f^{-1}(\boldsymbol{z}))}{p_{\boldsymbol{Y}}(f^{-1}(\boldsymbol{z}))}\right],\end{aligned} \tag{51}$$

where $p_{\boldsymbol{Z}}$ is the prior distribution, e.g., the Gaussian $\mathcal{N}(0, I)$ with I being a n-dimensional identity matrix.

We summarize our algorithm as follows:

Algorithm 1 An importance sampling estimator based on generative model.

- Sample the reduced-order model to compute the solution $u_{h,c}(\boldsymbol{y}^{(i)})$ and the error estimate of $g(u_{h,c}(\boldsymbol{y}^{(i)}))$, where we keep the data $\{\boldsymbol{y}^{(i)}\}_{i=1}^N$ with $g(u_{h,c}(\boldsymbol{y}^{(i)})) \geq \epsilon_{\max}^-$ (see equation (47)).
- Compute the weights associated with each sample using equation (49).
- Generate training dataset $\mathcal{D} = \cup_{i=1}^{K+1} S_k$ (see equation (50)).
- Train the generative model $p_{\boldsymbol{Y}}(\boldsymbol{y})$ by minimizing the objective function (46) with a stochastic gradient method such as ADAM.
- Use the importance sampling estimator (51) to sample the fine model.

6. Numerical experiments

In this section, we do some experiments to study the numerical strategies we have proposed. The ADAM optimization solver with a fixed learning rate is used for all examples [10]. The numerical algorithm has been implemented with Tensorflow.

6.1. Rotate Gaussian random variables

We start with a simple case. Assume that we have data for the random variable $\boldsymbol{Y} = (Y_1, Y_2)$ with Y_i being i.i.d. normal random variables. The entropy of $\mu_{\boldsymbol{Y}}$ is $H(\mu_{\boldsymbol{Y}}) = \ln(2\pi e)$. We know that $\hat{\boldsymbol{Y}} = A\boldsymbol{Y}$ are still Gaussian random variables, where $A \in \mathbb{R}^{2\times 2}$. Furthermore,

$$\mathrm{Cov}(\hat{\boldsymbol{Y}}) = A\mathrm{Cov}(\boldsymbol{Y})A^{\mathsf{T}} = AA^{T}.$$

If A is a unitary matrix, $\hat{Y}_1$ and $\hat{Y}_2$ are i.i.d. normal random variables. We use the flow-based generative model to describe the mapping, i.e., rotation, from $\hat{\boldsymbol{Y}}$ to $\boldsymbol{Y}$.

Let us see if the multi-layer mapping $f(\boldsymbol{x})$ defined in (14) is able to provide a rotation of (Y_1, Y_2) using two affine coupling layers. According to equations (17) and (18), $f_{[1]}$ yields

$$y_1^{[1]} = y_1, \quad y_2^{[1]} = ay_2 + by_1,$$

where we choose $s(y_1) = a$ and $t(y_1) = by_1$ with a, b being constant. Similarly, we have the output of $f_{[2]}$ as

$$\hat{y}_1 = cy_1 + d(ay_2 + by_1) = (c + bd)y_1 + ady_2, \quad \hat{y}_2 = ay_2 + by_1,$$

where two more constants c and d are introduced. We then obtain the following condition such that A is unitary:

$$\begin{pmatrix} c+bd & ad \\ a & b \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} c+bd & ad \\ a & b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The above equation admits many possible solutions, e.g., $a = b = \sqrt{2}/2$, $c = -\sqrt{2}$ and $d = 1$. Any possible solution is a good enough for our purpose. In equation (30), μ_{data} is given by N samples of $\boldsymbol{Y}$. Then the cross entropy $H(\mu_{\text{data}}, \mu_{\boldsymbol{Y}})$ should converge to the entropy $H(\mu_{\boldsymbol{Y}})$, i.e., $\ln(2\pi e) \approx 2.8379$, as $N \to \infty$. If the flow-based generative model $p_{\boldsymbol{Y}}d\boldsymbol{y} = \tilde{\mu}_{\boldsymbol{Y}}$ provides a good approximation of $\mu_{\boldsymbol{Y}}$, the minimum of the cross entropy $H(\mu_{\text{data}}, \tilde{\mu}_{\boldsymbol{Y}})$ should yield a minimizer that converges to $\mu_{\boldsymbol{Y}}$ and a minimum value that converges to $H(\mu_{\boldsymbol{Y}})$. Such a convergence behavior is shown in Fig. 3, meaning that a rotation of Gaussian variables is well captured. The initial cross entropy is large because we choose a large standard deviation on purpose when we initialize the weights of each neuron. It is seen that the ADAM method stabilizes quickly.

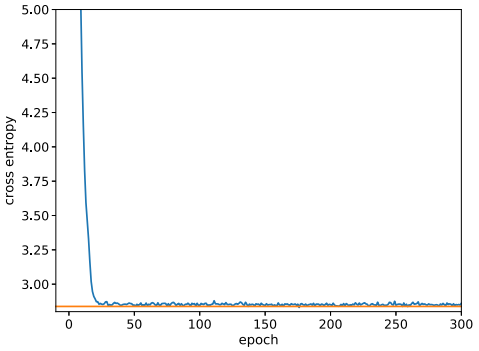

Fig. 3. The convergence behavior for the rotation of Gaussian variables, where the horizontal line indicates the entropy $H(\mu_{\boldsymbol{Y}}) = \ln(2\pi e)$. Four general coupling layers are used, i.e., $L = 2$. In equation (18), we let $\boldsymbol{s}(\cdot) = 1$ and only model $\boldsymbol{t}(\cdot)$ as a neural network $\mathrm{NN}(\cdot)$. The sample size is $N = 10^4$.

6.2. Two-dimensional conditional PDFs

We now consider the approximation of the following conditional PDF

$$p_{\boldsymbol{Y}|B}(\boldsymbol{y}) = \frac{I_B(\boldsymbol{y})\rho(\boldsymbol{y})}{\mathbb{E}[I_B]},$$

where we choose $\rho(\boldsymbol{y})$ as the joint PDF given by two i.i.d. normal random variables Y_1 and Y_2. The condition $B = \{\boldsymbol{y} | g(\boldsymbol{y}) \geq 0\}$ will introduce correlations between Y_1 and Y_2. Let $\hat{\boldsymbol{y}} = \Lambda R\boldsymbol{y}$, where $\Lambda = \mathrm{diag}(a, 1)$ is a scaling matrix with a being a constant, and R is a unitary matrix for rotation, i.e.,

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

We define the set $B = \{\boldsymbol{y} | \hat{\boldsymbol{y}}^{\mathsf{T}}\hat{\boldsymbol{y}} \geq C^2\}$. The distribution of $p_{\boldsymbol{Y}|B}(\boldsymbol{y})$ is demonstrated in Fig. 4 for $a = 2$, $\theta = \pi/4$ and $C = 3.0$ by $N = 5000$ samples. These are the data we will use to train the generative model.

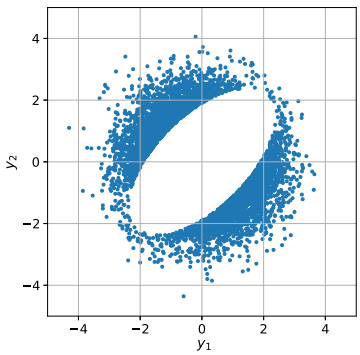

Fig. 4. The set $B = \{\boldsymbol{y} | \hat{\boldsymbol{y}}^{\mathsf{T}} \hat{\boldsymbol{y}} \geq C^2\}$ with $a = 2$, $\theta = \pi/4$ and $C = 3.0$. We assume that Y_1 and Y_2 are two i.i.d. normal random variables.

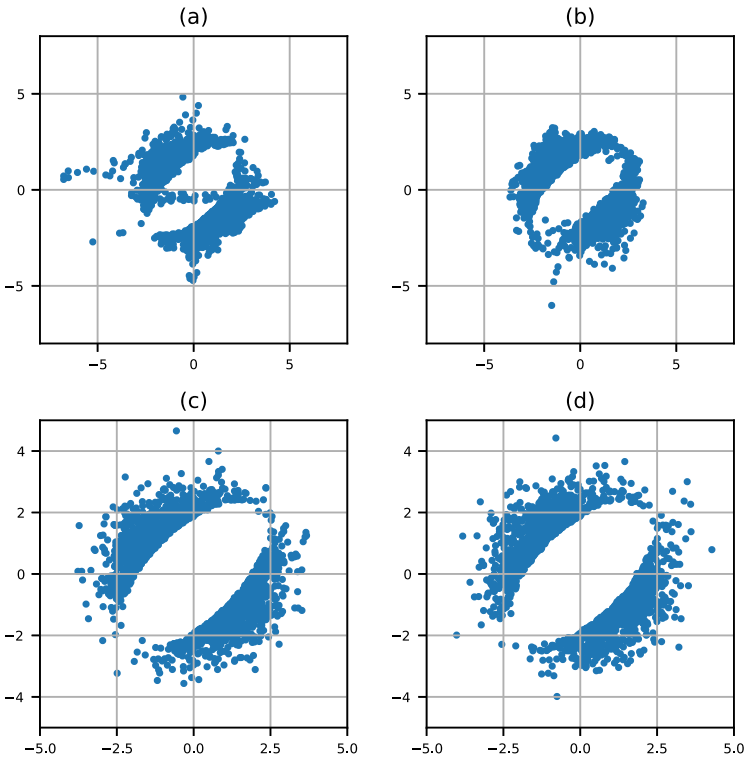

Fig. 5. Data sampled from the flow-based generative models with a Gaussian prior distribution. The sample size is $N = 10^4$. (a): $L = 2$; (b): $L = 4$; (c): $L = 8$; (d): $L = 16$.

We let the prior distribution be the two-dimensional normal distribution $\mathcal{N}(0, I)$ with I being a two-dimensional identity matrix. The transportation from the normal distribution to the desired conditional distribution is highly nonlinear due to the fact that the region of the highest density in the prior distribution has been removed. In Fig. 5, we plot the data sampled from the generative models trained with different depths. The neural network $\mathsf{NN}(\cdot)$ in equation (21) has two dense hidden layers, where the first hidden layer has 512 neurons and the second hidden layer has 256 neurons. It is seen that the approximated distribution improves as the depth L increases. When $L = 8$, the approximated distribution already agrees very well with the original distribution showed in Fig. 4. In Fig. 6, we demonstrate the mapping from $\boldsymbol{Z}$ to $\boldsymbol{Y}$, where $\boldsymbol{Z}$ is sampled from the Gaussian prior. For clarity, we split the data $\boldsymbol{z}$ to three groups, indicated by blue, red and green. The one-to-one correspondence between $\boldsymbol{z}$ and $\boldsymbol{y}$ yields the corresponding splitting of the data $\boldsymbol{y}$. It appears that the nonlinear mapping $f(\cdot)$ overall maps the high-density region in the prior distribution to the high-density region in the data distribution. Note that the blue region has been separated into two parts, meaning that the deep net is able to handle such a "discontinuity" using a continuous mapping.

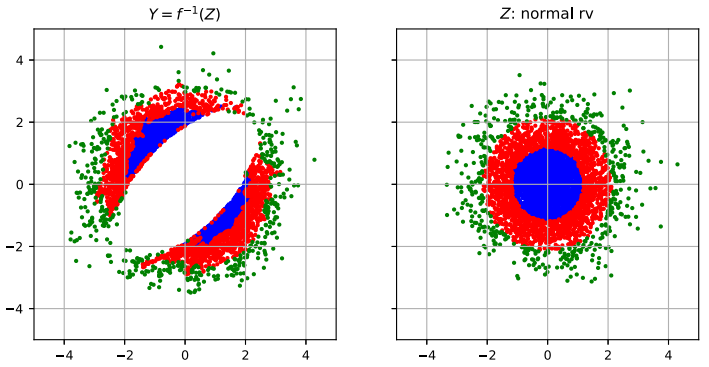

Fig. 6. The mapping from $\boldsymbol{Z}$ to $\boldsymbol{Y}$ given by the generative model with $L = 16$, where $\boldsymbol{Z}$ is subject to the prior normal distribution $\mathcal{N}(0, I)$. The sample size is $N = 10^4$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

6.3. One-dimensional elliptic problems with log-normal coefficients

We now consider a one-dimensional elliptic problem [16]

$$-\frac{d}{dx}\left(e^{a(x;\omega)}\frac{du}{dx}\right) = 1, \quad x \in [0,1], \tag{52}$$

where $a(x;\omega)$ is a zero-mean Gaussian random field subject to a normalized covariance kernel $K(x_1, x_2)$. For this one-dimensional problem, we can write down the exact solution

$$u(x;\omega) = -\int_0^x s e^{-a(s;\omega)} ds + \gamma \int_0^x e^{-a(s;\omega)} ds, \tag{53}$$

where γ is a random variable

$$\gamma = \left(\int_0^1 e^{-a(s;\omega)} ds\right)^{-1} \int_0^1 s e^{-a(s;\omega)} ds.$$

The random coefficient $a(x;\omega)$ can be approximated by the Karhunen-Loéve expansion:

$$a(x;\omega) \approx a_M(x;\xi) = \sigma \sum_{i=1}^{M} \sqrt{\lambda_i}\theta_i(x)\xi_i, \tag{54}$$

where σ indicates the standard deviation, $\xi_i \sim \mathcal{N}(0,1)$ are i.i.d. normal random variables, and $(\lambda_i, \theta_i(x))$ are the eigen-pairs of the covariance kernel $K(x_1, x_2)$. σ will be fixed to 1 from now on. Replacing $a(x)$ with $a_M(x)$ in u, we obtain $u_M(x) \approx u(x)$, which will be our exact solution. Define the set $B = \{u_M \| u_M \|_{H^1} \geq C\}$ with C being a positive number. We will estimate $\mathbb{E}[I_B]$ by sampling.

Consider a one-dimensional exponential covariance kernel on $x \in [0,1]$

$$K(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{l_c}}.$$

Its eigenvalues satisfy

$$v^2 = \frac{2\epsilon - \epsilon^2\lambda_i}{\lambda_i}, \quad (v^2 - \epsilon^2)\tan(v) - 2\epsilon v = 0, \tag{55}$$

where $\epsilon = 1/l_c$. Its eigenfunctions have the following form [8]

$$\theta_i(x) = \frac{v\cos(vx) + \epsilon\sin(vx)}{\sqrt{\frac{1}{2}(\epsilon^2 + v^2) + (w^2 - \epsilon^2)\frac{\sin(2v)}{4v} + \frac{\epsilon}{2}(1 - \cos(2v))}}. \tag{56}$$

Let $\Pi_{h,c}$ be an interpolation operator defined on the coarse mesh. We let

$$a_{M,h,c}(x;\boldsymbol{y}) = \sum_{i=1}^{M} \sqrt{\lambda_i}\Pi_{h,c}\theta_i(x)\xi_i,$$

which yields the approximate solution $u_{M,h,c}$. For the reduced-order model, all the integrals will be approximated by the rectangle rule which has a first-order accuracy. The fine model will be based on spectral/*hp* element method. More specifically, we consider the interpolation and integration using 64 equidistant elements with 8 Gauss-Lobatto-Legendre points in each element. For simplicity, the error of the reduced-order model will be computed directly using the fine model as the reference solution.

6.3.1. Distribution of data missed by the reduced-order model

We first look at the necessity of considering the weighted empirical distribution. In Table 1, we summarize the information about 10^4 samples from both the reduced-order and the fine models, where the mesh for the reduced-order model $u_{h,c}$ consists of 10 equidistant linear finite elements. The probability $\Pr(B)$ is chosen around 0.1. It is seen that to reduce the bias from the reduced-order model, we need to keep another 2,546 samples that do not satisfy $\|u_{M,h,c}\|_{H^1} \geq C$. However, among these samples, only 107 are effective, which is around $\frac{107}{2546} \approx 4\%$. If we do density estimation using $1{,}263 + 2{,}546 = 3{,}809$ samples, $2{,}546 - 107 = 2{,}539$ samples do not contribute at all to our desired random event, which are $\frac{2{,}539}{3{,}809} \approx 67\%$ of the total samples. Such a situation can be worse if we use a posterior error estimate because the effective index of the estimator may be several times larger than 1, i.e., the estimated error may be several times larger than the real error. To alleviate this issue, we need to put more weights into the 1,263 samples that satisfy $\|u_{M,h,c}\|_{H^1} \geq C$ and less weights to the redundant 2,546 samples that are induced by the discretization error of the reduced-order model. We note that the 107 useful samples will also be weighed by doing so. A compromise is to assign the weights to the data $\{C - \epsilon^-_{\max} \leq \|u_{M,h,c}\| < C\}$ in a consistent way with the distribution of the data $\{\|u_{M,h,c}\| < C \text{ and } \|u_{M,h,f}\| \geq C\}$. In Fig. 7, we plot the normalized histograms of some conditioned distribution of $g(u_{M,h,c}) = \|u_{M,h,c}\| - C$. In the left plot of Fig. 7, we show the distribution of $g(u_{M,h,c})$ given by the data where the reduced-order model fails to capture B, i.e., $g(u_{M,h,c}) < 0$ while $g(u_{M,h,f}) \geq 0$. It is seen that as the value of $g(u_{M,h,c})$ decreases, the probability that the reduced-order model fails also decreases. In the right plot of Fig. 7, we show the distribution of $g(u_{M,h,c})$ given by the data that satisfy $-\epsilon^-_{\max} \leq g(u_{M,h,c}) < 0$. It is seen that the density increases as the value of $g(u_{M,h,c})$ decreases, which is the opposite of the histogram in the left plot. This is because we have kept redundant data to compensate the discretization error of the reduced-order model. First, the probability that $C - \epsilon^-_{\max} \leq \|u_{M,h,f}\|_{H^1} < C$ is much larger than the probability that $\|u_{M,h,c}\|_{H^1} < C$ and $\|u_{M,h,f}\|_{H^1} \geq C$. Second, $\epsilon^-_{\max}$ is not the optimal choice, which may be much larger than necessary. At this moment, we do not have a better understanding about the choice of the lower bound for $-\epsilon^-_{\max} \leq g(u_{M,h,c}) < 0$.

Table 1
Samples from the coarse model, where $C = 0.8$, $l_c = 1$, and $M = 50$.

# of samples	10^4
$\|u_{M,h,c}\|_{H^1} \geq C$	1,263
$\|u_{M,h,f}\|_{H^1} \geq C$	1,300
$C - \epsilon^-_{\max} \leq \|u_{M,h,c}\|_{H^1} < C$	2,546
$\|u_{M,h,c}\|_{H^1} < C$ and $\|u_{M,h,f}\|_{H^1} \geq C$	107

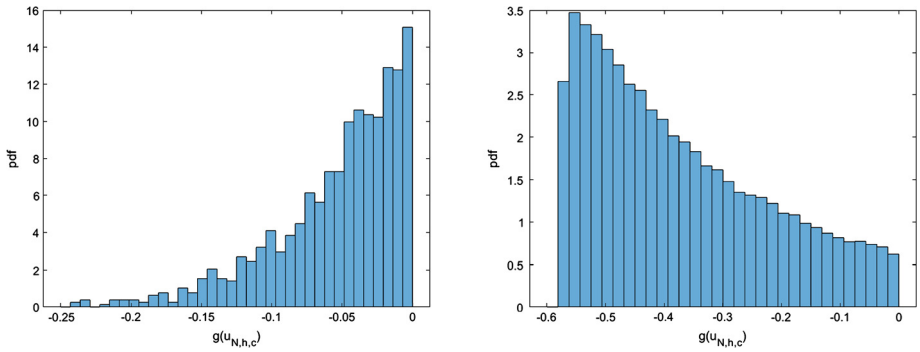

Fig. 7. The conditioned distribution of $g(u_{N,h,c})$. Left: The data missed by the coarse model, i.e., $g(u_{N,h,c}) < 0$ while $g(u_{N,h,f}) \geq 0$. Right: The data that satisfy $C - \epsilon^-_{\max} \leq \|u_{N,h,c}\|_{H^1} < C$.

6.3.2. Importance sampling via the trained generative model

We now look at the performance of the generative model for the importance sampling estimator. Let σ_{I_B} and σ_w be the standard deviation of I_B and

$$w(\boldsymbol{Y}) = \frac{I_B(\boldsymbol{Y})\rho(\boldsymbol{Y})}{p_{\boldsymbol{Y}}(\boldsymbol{Y})},$$

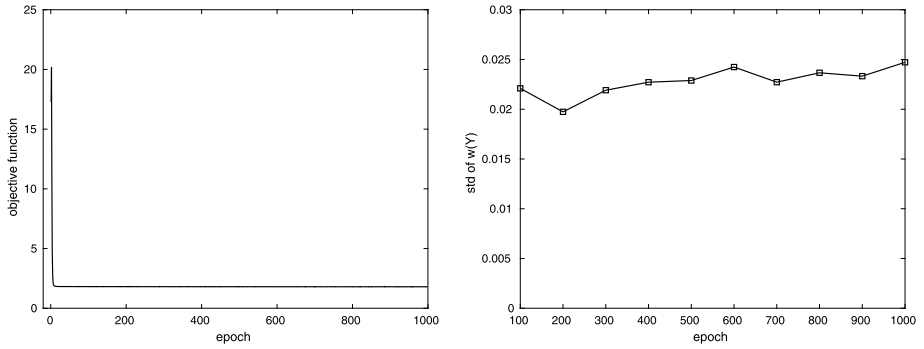

Fig. 8. $M = 2$. Left: The evolution behavior of stochastic optimization, where the penalty term is not included in the objective function. Right: The standard deviation of $w(\boldsymbol{Y})$.

where $p_{\boldsymbol{Y}}$ indicates the trained generative model. Let N_{MC} and N_{IS} be the sample size for the Monte Carlo estimator and the importance sampling estimator to achieve the same degree of confidence interval for the mean subject to a certain error. We know that

$$\frac{N_{\mathsf{IS}}}{N_{\mathsf{MC}}} \approx \left(\frac{\sigma_w}{\sigma_{I_B}}\right)^2.$$

So we only need to focus on the variance reduction of σ_w in terms of σ_{I_B}.

Following is the setup of our numerical experiments. We compute σ_{I_B} using the fine physical model by the Monte Carlo method with 10^5 samples. The depth L of the generative model is set to 16. Each affine coupling layer has two fully coupled hidden layers, where the first one has 512 neurons and the second one has 256 neurons. Since we focus on the robustness and effectiveness of the generative model, we simple use a large number of neurons to encourage overfitting for the unregularized generative model. In each coupling layer $f_{[i]}$, we consider a fixed partition of the vector. Considering that the eigenvalue decays, we split $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_{2m})$ into $\boldsymbol{\xi}_1 = (\xi_1, \xi_3, \ldots, \xi_{2m-1})$ and $\boldsymbol{\xi}_2 = (\xi_2, \xi_4, \ldots, \xi_{2m})$, where all components with odd indices are separated from those with even indices. We then train the generative model using the data given by the reduced-order model and compute σ_w by sampling the generative model 10^5 times. For all cases, the generative model will be trained by the ADAM method with a learning rate 2e-4, where the data have been split to 23 mini-batches. We sample the reduced-order model 10^5 times, and keep a portion of the data as the training set. Since we choose that $\Pr(B) \approx 0.1$, about 10^4 samples satisfy $g(u_{h,c}) \geq 0$, although the real number may vary a little. We set $\hat{\alpha} = 0.85$ when computing the weights of the data (see equation (49)).

We start with a relatively large correlation length $l_c = 1$, such that the eigenvalue decays fast. The coarse mesh consists of 10 equidistant linear finite elements. We first look at a two-dimensional case, i.e., $M = 2$, where $\mathbb{E}[I_B] \approx 0.109$ and $\sigma_{I_B} \approx 0.312$. The training set from the reduced-order model includes 10,683 samples satisfying $g(u_{M,h,c}) \geq 0$, and 3,051 samples satisfying $g(u_{M,h,c}) < 0$, among which only 242 samples are really missed by the reduced-order model, i.e., $g(u_{M,h,c}) < 0$ while $g(u_{M,h,f}) \geq 0$. In Fig. 8, we plot the results for $M = 2$. On the left, we plot the evolution behavior of the stochastic optimization, where no penalty term is included in the objective function, i.e., $\beta = 0$; On the right, we plot the standard deviation of σ_w versus the epoch, where σ_w is computed in terms of the generative model trained up to a certain epoch. It is seen that the stochastic optimization stabilizes quickly while σ_w varies a little around 0.025. For this case,

$$\frac{N_{\mathsf{IS}}}{N_{\mathsf{MC}}} \approx \left(\frac{0.025}{0.312}\right)^2 \approx 0.64\%.$$

In other words, for the same level of accuracy, the number of samples needed by the importance sampling estimator is about 0.64% of that for a direct Monte Carlo estimator. The speed up can be significant even after taking into account the cost from sampling the reduced-order model and training the generative model, since the complexity of the generative model does not increase with the complexity of the physical model. The comparison between the data distribution and the estimated distribution is given in Fig. 9.

We then consider a four-dimensional case, i.e., $M = 4$, where $\mathbb{E}[I_B] \approx 0.121$ and $\sigma_{I_B} \approx 0.326$. The training set from the reduced-order model includes 12,032 samples satisfying $g(u_{M,h,c}) \geq 0$, and 7,519 samples satisfying $g(u_{M,h,c}) < 0$, among which only 593 samples are really missed by the reduced-order model, i.e., $g(u_{M,h,c}) < 0$ while $g(u_{M,h,f}) \geq 0$. The simulation results are given in Fig. 10. There are several interesting observations: First, if no penalty term is included in the objective function, the evolution of stochastic optimization has two types of behavior. The function value plummets at the beginning and then decays very slowly. This is because the size of the data set is relatively small in terms of the dimension M such that the overfitting occurs. Note that for this case, the standard deviation of σ_w increases with respect to the epoch, meaning that the efficiency of the importance sampling estimator decreases if the training of the generative model

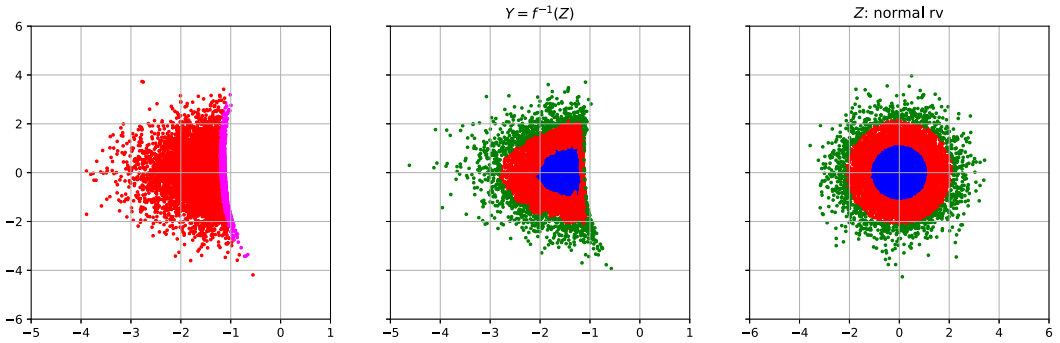

Fig. 9. Left: The data from the reduced-order models, where the red color indicates $g(u_{h,c}) \geq 0$ while the magenta color indicates $g(u_{h,c}) < 0$. Middle: The estimated distribution given by the generative model with $L = 16$; Right: The priori distribution given by two iid normal random variables.

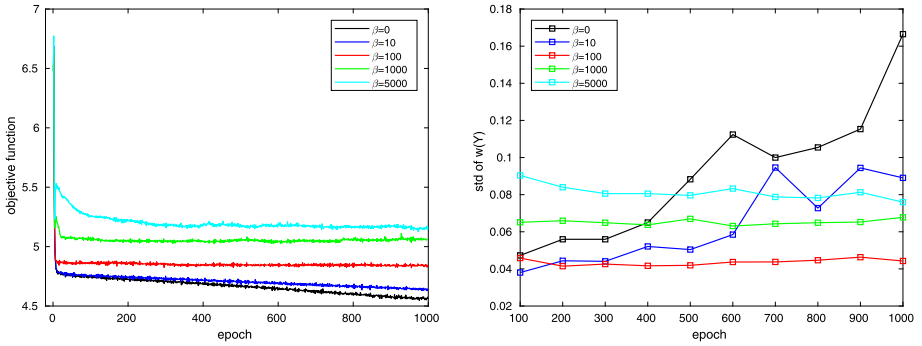

Fig. 10. $M = 4$. Left: The evolution behavior of stochastic optimization, where the penalty term varies in terms of β. Only the cross entropy has been plotted. Right: The standard deviation of $w(\boldsymbol{Y})$.

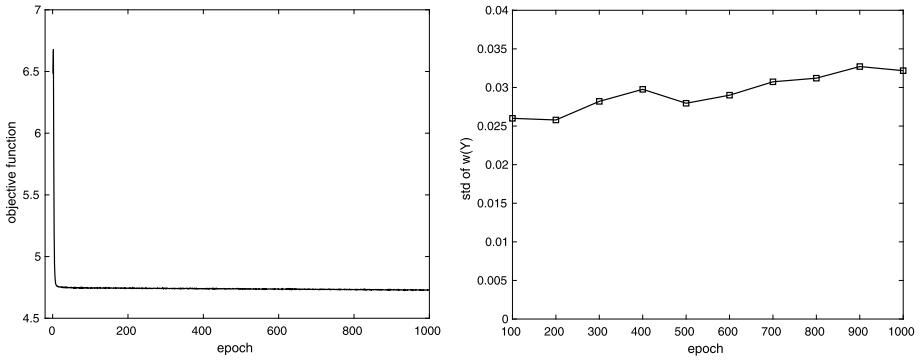

Fig. 11. $M = 4$. Left: The evolution behavior of stochastic optimization, where the penalty term is not included in the objective function. Right: The standard deviation of $w(\boldsymbol{Y})$.

is stopped at a larger epoch. Second, when more and more penalty is included, the slow decay in the optimization iteration disappears, implying that the regularization works. Furthermore, σ_w stops increasing after the regularization is introduced. It appears that σ_w increases with respect to β, meaning too much regularization will deteriorate the efficiency of importance sampling estimator. Third, note that when the epoch is 100, the generative models subject to $\beta = 0$ and 100 give a comparable σ_w. This implies that early stopping may be used. However, it seems that the penalty term yields much more robustness. For $\beta = 100$, $\sigma \approx 0.042$, which yields that

$$\frac{N_{\mathrm{IS}}}{N_{\mathrm{MC}}} \approx \left(\frac{0.042}{0.326}\right)^2 \approx 1.66\%.$$

The other way to alleviate the overfitting is to enlarge the training set. In Fig. 11, we plot the results subject to a larger training set, which has 120,137 samples satisfying $g(u_{M,h,c}) \geq 0$ and 81,839 samples satisfying $g(u_{M,h,c})<0$. For this case, a smaller σ_w is achieved without using any penalty term in the objective function.

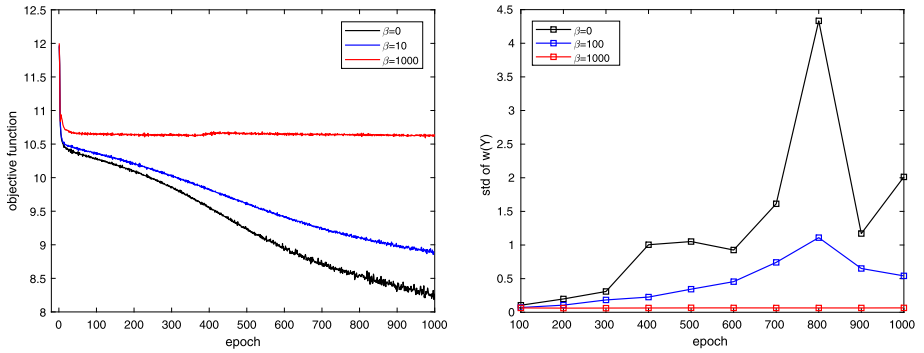

Fig. 12. $M = 8$. Left: The evolution behavior of stochastic optimization, where the penalty term varies in terms of β. Only the cross entropy has been plotted. Right: The standard deviation of $w(\boldsymbol{Y})$.

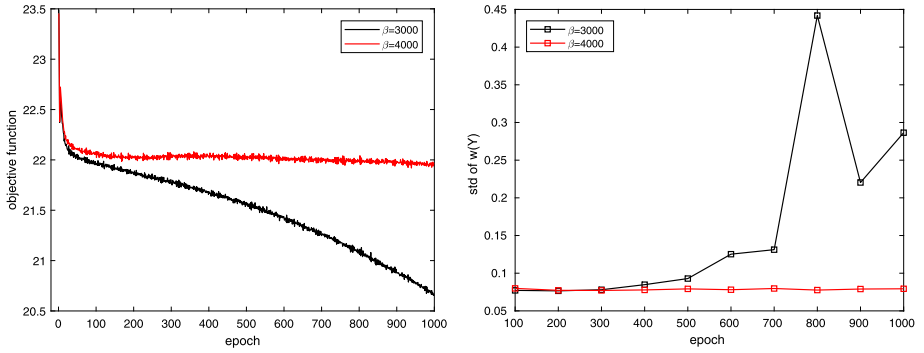

Fig. 13. $M = 16$. Left: The evolution behavior of stochastic optimization, where the penalty term varies in terms of β. Only the cross entropy has been plotted. Right: The standard deviation of $w(\boldsymbol{Y})$.

We now look at an eight-dimensional case, i.e., $M = 8$, where $\mathbb{E}[I_B] \approx 0.130$ and $\sigma_{I_B} \approx 0.336$. The training set from the reduced-order model includes 12,504 samples satisfying $g(u_{M,h,c}) \geq 0$, and 22,049 samples satisfying $g(u_{M,h,c}) < 0$, among which only 828 samples are really missed by the reduced-order model, i.e., $g(u_{M,h,c}) < 0$ while $g(u_{M,h,f}) \geq 0$. The results are given in Fig. 12. Compared to the previous case, similar results have been observed. Since the dimension is doubled but the size of training set remains the same, it is seen that the performance of the generative model deteriorates quickly as the epoch increases if no penalty term is used. Actually, after epoch 300 σ_w is larger than 0.33 when $\beta = 0$, meaning that the importance sampling estimator is less efficient than the Monte Carlo estimator. Again, the penalty term can stabilize σ_w, which is about 0.063 for $\beta = 1000$. For this case,

$$\frac{N_{\mathrm{IS}}}{N_{\mathrm{MC}}} \approx \left(\frac{0.063}{0.336}\right)^2 \approx 3.52\%.$$

We double the dimension to consider $M = 16$, where $\mathbb{E}[I_B] \approx 0.134$ and $\sigma_w \approx 0.340$. The training data set includes 12,975 samples that $g(u_{M,h,c}) \geq 0$ and 34,847 samples that $g(u_{M,h,c}) < 0$, among which only 913 samples are really missed by the reduced-order model. The results are plotted in Fig. 13. For $\beta = 4000$, we obtain $\sigma_w \approx 0.078$, which yields that

$$\frac{N_{\mathrm{IS}}}{N_{\mathrm{MC}}} \approx \left(\frac{0.078}{0.340}\right)^2 \approx 5.26\%.$$

We finally consider a case that $M = 32$, where $\mathbb{E}[I_B] \approx 0.134$ and $\sigma_w \approx 0.341$. The training data set includes 13,267 samples that $g(u_{M,h,c}) \geq 0$ and 30,402 samples that $g(u_{M,h,c}) < 0$, among which only 1,067 samples are really missed by the reduced-order model. Although the dimension is high and the data set is relatively small, we obtain $\sigma_w \approx 0.089$ with $\beta = 8000$ (see Fig. 14), which yields that

$$\frac{N_{\mathrm{IS}}}{N_{\mathrm{MC}}} \approx \left(\frac{0.089}{0.341}\right)^2 \approx 6.81\%.$$

To this end, we have studied the performance of the generative-model-based importance sampling estimator for different random dimension M, where the configuration of the generative model is fixed, the ADAM method is subject to a fixed

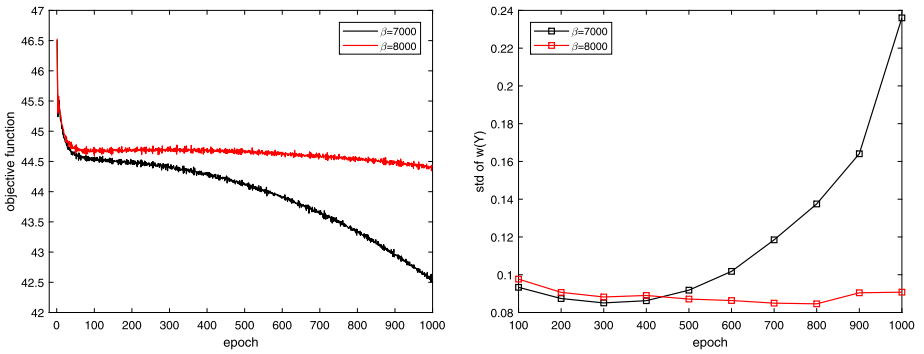

Fig. 14. $M = 32$. Left: The evolution behavior of stochastic optimization, where the penalty term varies in terms of β. Only the cross entropy has been plotted. Right: The standard deviation of $w(\boldsymbol{Y})$.

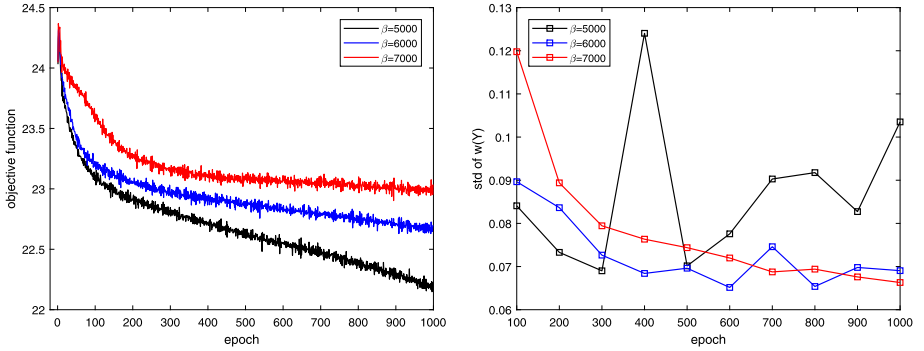

Fig. 15. $M = 16$. Left: The evolution behavior of stochastic optimization, where the penalty term varies in terms of β. Only the cross entropy has been plotted. Right: The standard deviation of $w(\boldsymbol{Y})$.

learning rate, and the size of training set remains comparable for all M. Very encouraging results have been obtained in terms of the ratio $N_{\mathrm{IS}}/N_{\mathrm{MC}}$ for M varying from 2 to 32. The penalty term in the objective function appears important for the robustness of the algorithm. Note that all cases we have studied so far are subject to a relative large correlation length $l_c = 1$, e.g., $\frac{\lambda_{16}}{\lambda_1} = 1.22\text{e-}3$. The fast decay of the eigenvalues may reduce the difficulty of density estimation in terms of the dimensionality. To clarify this concern, we study a relatively small correlation length $l_c = 0.1$ and let $M = 16$, where $\frac{\lambda_{16}}{\lambda_1} = 4.53\%$. Due to the slower decay of eigenvalues, the high-order modes in the Karhunen-Loéve expansion will play a much more important role for the value of $\|u\|_{H^1(D)}$. We then refine the coarse mesh from 10 equidistant linear finite elements to 30 to let the coarse model capture more information about the eigenfunctions for small eigenvalues. We have $\mathbb{E}[I_B] \approx 0.093$ and $\sigma_w \approx 0.290$. The training data set includes 9,010 samples that $g(u_{M,h,c}) \geq 0$ and 40,568 samples that $g(u_{M,h,c}) < 0$, among which only 609 samples are really missed by the reduced-order model. The results have been plotted in Fig. 15. Compared to the previous cases with a large correlation length, the relaxation time of stochastic optimization increases in the sense that the optimal generative model will be achieved at a larger epoch. Other than that, the results are qualitatively similar to previous observations. In particular, the penalty term is critical for robustness. For $\beta = 7000$, we are able to obtain

$$\frac{N_{\mathrm{IS}}}{N_{\mathrm{MC}}} \approx \left(\frac{0.071}{0.290}\right)^2 \approx 6.00\%.$$

We now let $M = 32$, where $\frac{\lambda_{32}}{\lambda_1} = 1.11\%$, $\mathbb{E}[I_B] \approx 0.115$ and $\sigma_w \approx 0.319$. The data from the reduced-order model include 11,260 samples that $g(u_{M,h,c}) \geq 0$ and 88,342 samples that $g(u_{M,h,c}) < 0$, among which only 905 samples are really missed by the reduced-order model. It is seen that the redundant data is about eight times as many as the data that $g(u_{M,h,c}) \geq 0$. This is because the high-order eigenfunctions θ_i are highly oscillating, and become more important in the evaluation of $\|u\|_{H^1(D)}$ when eigenvalues decay slowly. The coarse mesh cannot capture the high oscillation well, and introduce a large error when the random variables associated with the high-order eigenfunctions take a large value. We here simply truncate the data set with respect to the value of $|g_{h,c}(\boldsymbol{y}^{(i)})|$. We only keep half of the data that $g(u_{M,h,c}) < 0$, which have a smaller $|g_{h,c}(\boldsymbol{y}^{(i)})|$. Since the dependence on the high-order eigenfunctions is stronger, we increase the depth L from 16 to 24 for the generative model. Other than that, all other set-up remains the same. The results are plotted in Fig. 16. When the epoch is 1000, we obtain

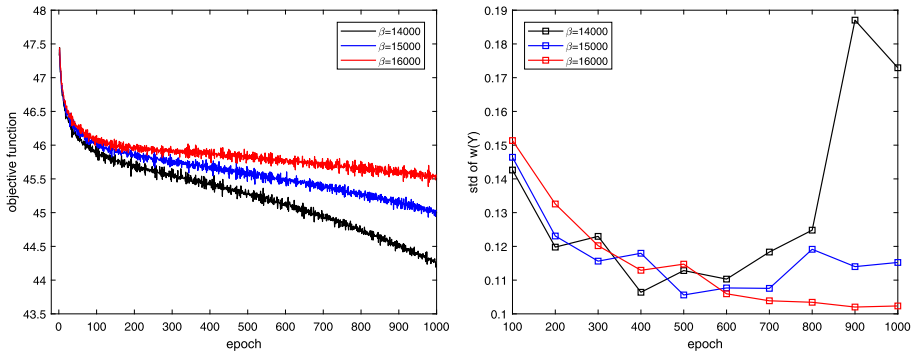

Fig. 16. $M = 32$. Left: The evolution behavior of stochastic optimization, where the penalty term varies in terms of β. Only the cross entropy has been plotted. Right: The standard deviation of $w(\boldsymbol{Y})$.

$$\frac{N_{\mathrm{IS}}}{N_{\mathrm{MC}}} \approx \left(\frac{0.102}{0.319}\right)^2 \approx 10.22\%,$$

with $\beta = 16000$.

In summary, we are able to obtain a speedup of $O(10)$ for a random dimension up to 32 for the elliptic problem studied, where the speed up is measured by the ratio $N_{\mathrm{MC}}/N_{\mathrm{IS}}$. It is observed that the speedup will decrease with respect to the dimension. However, we should note that for all the cases we use a comparable amount of effective samples, in other words, the number of samples satisfying $g(u_{M,h,c}) \geq 0$ is always around 10^4. In contrast to other density techniques such as mixture of Gaussians, the generative model seems less sensitive to dimensionality in the sense that it is able to achieve a good performance for a relatively large dimension using a relatively small training dataset.

7. Summary and discussions

In this work we have proposed a methodology to couple the reduced-order model and the generative model to construct an importance sampling estimator. Our numerical experiments show that this idea is actually feasible although the approximation of high-dimensional PDF is difficult due to the curse of dimensionality. From the application point of view, the generative models have been trained to approximate the data distribution given by high-resolution images, where the criterion for effectiveness is quite ad hoc although the dimensionality is really high. We adapt the generative model to deal with a physical problem and measure its effectiveness rigorously through the variance reduction it is able to achieve. It appears that the generative model does have the ability to encode the information in the high-dimensional data from a physical model. However, we note that the properties of the problem should be incorporated into the training process to enhance the robustness. For our problem, the regularization induced by the penalty term is much more robust than a general regularization technique in machine learning such as early stopping. We have demonstrated that the generative-model-based important sampling estimator can achieve a significant variance reduction for at least random dimensions of $O(10)$ with respect to a UQ problem. To test the robustness, we have fixed the configuration of the generative model and the parameters of the optimization algorithm. For the problems studied, at least about 90% reduction in variance is achieved for the dimension M up to 32 with about 10^4 samples. At this moment, it is unclear how many random dimensions the generative model can effectively deal with for UQ problems in terms of the variance reduction of importance sampling. However, the scalability of deep nets makes it very promising to apply our methodology to a larger random dimension by using a larger depth L.

There are many possibilities to improve the current work. For example, in all our numerical experiments, a fixed partition of the random vector is used. A more effective partition strategy can be employed especially when the number of effective random dimensions is much smaller than the total number of random dimensions. Other generative models can also be employed. The invertible mapping has been recently introduced into a general adversarial network such that GAN is able to perform exact likelihood evaluation [6]. In [17], a new flow-based generative model is proposed by incorporating the optimal transport theory. How these flow-based models help importance sampling in our problem setting is an interesting question. Another possibility is to take into account the dimension reduction in the probability space such that we can mainly focus on the effective random dimensions. The research on these issues will be reported elsewhere.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The first author's work was supported by AFOSR grant FA9550-15-1-0051, and NSF grants DMS-1620026 and DMS-1913163, and the second author's work was supported by NSF grants 1320351 and 1642991.

References

[1] L. Dinh, D. Krueger, S. Bengio, Nice: non-linear independent components estimation, arXiv:1410.8516, 2014.
[2] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real NVP, arXiv:1605.08803v3, 2017.
[3] M.B. Giles, Multilevel Monte Carlo methods, Acta Numer. (2015) 259–328.
[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. (2014) 2672–2680.
[5] A. Graves, Generating sequences with recurrent neural networks, arXiv:1308.0850, 2013.
[6] A. Grover, M. Dhar, S. Ermon, Flow-GAN: combining maximum likelihood and adversarial learning in generative models, arXiv:1705.08868v2, 2018.
[7] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariance shift, arXiv:1502.03167v3, 2015.
[8] M. Jardak, C.-H. Su, G. Karniadakis, Spectral polynomial chaos solutions of the stochastic advection equation, J. Sci. Comput. 17 (2002) 319–338.
[9] D.P. Kingma, P. Dhariwal, Glow: generative flow with invertible 1x1 convolutions, arXiv:1807.03039v2, 2018.
[10] D.P. Kingma, J.L. Ba, ADAM: a method for stochastic optimization, arXiv:1412.6980v9, 2017.
[11] D.P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, M. Welling, Improving variational inference with inverse autoregressive flow, Adv. Neural Inf. Process. Syst. (2016) 4743–4751.
[12] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, arXiv:1601.06759, 2016.
[13] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, K. Kavukcuoglu, Conditional image generation with PixcelCNN decoders, arXiv:1606.05328, 2016.
[14] G. Papamakarios, T. Pavlakou, I. Murray, Masked autoregressive flow for density estimation, arXiv:1705.07057v4, 2018.
[15] D. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, 2nd edition, John Wiley & Sons, Inc., 2015.
[16] X. Wan, B.L. Rozovskii, The Wick-Malliavin approximation of elliptic problems with log-normal random coefficients, SIAM J. Sci. Comput. 35 (5) (2013) A2370–A2392.
[17] L. Zhang, W. E, L. Wang, Monge-Ampére flow for generative modeling, arXiv:1809.10188v1, 2018.
[18] T. Zhang, B. Yu, Boosting with early stopping: convergence and consistency, Ann. Stat. 33 (4) (2005) 1538–1579.