Applied Mathematics for Modern Challenges
Vol. 2, No. 4, December 2024, pp. 365-381
doi:10.3934/ammc.2024019

DIFFUSION-BASED SPARSE-GRID GENERATIVE MODELS FOR DENSITY ESTIMATION

Yanfang Liu[1], Alisa Bryantseva[2], Miroslav Stoyanov[3], Feng Bao[4] and Guannan Zhang[3*]

[1]Department of Mathematical Sciences, Middle Tennessee State University, Murfreesboro, TN 37132, USA

[2]Farragut High School, Knoxville, TN 37934, USA

[3]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

[4]Department of Mathematics, Florida State University, Tallahassee, FL 32306, USA

(Communicated by Richard Archibald)

Abstract. In density estimation, generative models are usually categorized under unsupervised learning due to the lack of labeled data. These models apply various indirect loss functions to refine neural network training, yet face specific challenges. Issues like mode collapse and instability in generative adversarial networks are notable, while normalizing flows are constrained by the need to calculate the Jacobian matrix's determinant, limiting network design. While neural networks are well-suited for handling very high-dimensional data, they can be overly complex for moderately high dimensions. Here, traditional sparse polynomial approximation offers advantages by avoiding complex training requirements. This research employs a score-based diffusion model combined with sparse grid interpolation to estimate the unknown density function. The method involves generating labeled data pairs linking samples from the standard Gaussian distribution to the target distribution using the diffusion model. This model transports the Gaussian distribution to the target density through a backward stochastic differential equation, where a Monte Carlo method approximates the score function at any point, facilitating function interpolation for the transport model. A sparse grid interpolant can be built based on the labeled data. We leverage the Tasmanian library [32] for building this sparse-grid-based generative model. We demonstrate the performance of our method using a set of multi-dimensional benchmark distributions.

2020 *Mathematics Subject Classification.* 60G25, 65L99, 65Z05.

Key words and phrases. Diffusion models, sparse grids, score function, density estimation, generative models.

*Corresponding author: Guannan Zhang.

1. **Introduction.** The process of density estimation focuses on approximating a dataset's probability density function to uncover its fundamental structure. In this context, generative machine learning models are designed to replicate the dataset's probability distribution, facilitating the creation of new data points that mimic the original ones. Over recent years, several innovative approaches to generative modeling have been developed, such as Variational Auto-encoders (VAEs) [16], Generative Adversarial Networks (GANs) [7, 10], normalizing flows [17, 22], and diffusion models [35]. These techniques have proved valuable across various domains, including generating images, reducing noise in images [6, 28], detecting anomalies [23, 27], and processing natural language [1, 13]. The success of these modern generative models hinges on their ability to harness the deep neural networks' advanced features to analyze and model the intricate patterns present in data, which may be of high dimensionality.

In the realm of density estimation, generative models are primarily categorized under unsupervised learning techniques due to the non-requirement of labeled datasets. These models utilize various types of loss functions to optimize neural network training. Notably, GANs use adversarial loss [10], normalizing flows employ maximum likelihood loss [17], and diffusion models implement score matching losses [14, 29, 34]. While these models have shown efficacy, unsupervised training methods introduce specific challenges. Training issues in GANs, such as mode collapse and instability [26], are well-documented. Likewise, normalizing flows are limited by the need for calculating the determinant of the Jacobian matrix, affecting the design of the networks. Notably, Continuous Normalizing Flows (CNFs) address this challenge by utilizing neural Ordinary Differential Equations (ODEs) [8, 21]. However, CNFs increase computational demands and can exhibit numerical instability during the ODE integration. On another front, Monotone Triangular Transport Maps (MTTMs) [4, 24] provide a different approach by constructing inverse triangular transport maps through monotonic functions. Despite their computational efficiency, MTTMs often lack the flexibility and expressiveness required for managing complex, high-dimensional data, due to the sequential nature of their transformations.

Alternatively, generating labeled data from the observational data can enable supervised training of the model's generative parts, such as the decoders in VAEs or normalizing flows, helping to bypass many unsupervised training complications. In our recent research documented in [2, 3, 18, 19], we introduced a novel diffusion model that obviates the need for traditional training, allowing the target generator to benefit from supervised learning techniques. Supervised learning, despite its advantages in stability and manageability, still requires meticulous adjustment of various hyperparameters such as the learning rate, number of neurons and layers, regularization strategies, and criteria for early stopping. While neural networks are well-suited for handling very high-dimensional data, they can be overly complex for moderately high dimensions. In such cases, the traditional sparse polynomial approximation method offers substantial benefits, primarily because it bypasses the complex training demands associated with neural network models.

In this work, we address the problem of estimating the unknown density function using a score-based diffusion model combined with a sparse grid. The key idea is to use the diffusion model to generate labeled data pairs between samples from the standard Gaussian distribution and the unknown target distribution. Initially, we create a sparse grid and corresponding basis functions, where the sparse grid

serves as sparse samples of the standard Gaussian distribution. The score-based diffusion model is then employed to transport the Gaussian distribution to the target density function by solving the backward diffusion process in the form of a stochastic differential equation (SDE). The score function in the backward SDE stores all the information of the target data distribution. A training-free score function estimation is obtained by using a mini-batch-based Monte Carlo method, which directly approximates the score function at any spatial-temporal location in solving the reverse-time SDE. In this process, labeled data can be generated, which allows function interpolation to approximate the transport model. We utilize the ORNL-developed library Tasmanian (https://github.com/ORNL/TASMANIAN) [32] to build the sparse-grid-based generative model.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the density estimation problem under consideration. In Section 3, we provide a comprehensive discussion of the score-based generative diffusion model and sparse grid interpolation for density estimation. Finally in Section 4, we demonstrate the performance of our method by applying it to a set of 1D datasets as well as a 5D funnel example.

2. Problem setting.

We consider the problem of how to generate an unlimited number of samples from an unknown probability density function $p_Y(y)$ of a $d-$dimensional random variable, denoted by

$$Y \in \mathbb{R}^d,\ Y \sim p_Y(y), \tag{1}$$

given observational dataset

$$\mathcal{Y} = \{y_1, y_2, \cdots, y_K\} \subset \mathbb{R}^d, \tag{2}$$

where $y_k \sim p_Y(y)$ for $k = 1, 2, \cdots, K$ are independent and identically distributed samples from the target distribution. The main objective is to build a generator, denoted by

$$Y = F(X),\ X \in \mathbb{R}^d, \tag{3}$$

that can map the d-dimensional standard Gaussian random variable X to the target random variable Y. Once the generator F is determined, unlimited samples of Y can be drawn by evaluating F at the samples of the standard Gaussian random variable X.

The density estimation problem has been extensively studied in the machine learning community using methods such as generative adversarial networks, auto-encoders, normalization flow models, and diffusion generative models. However, due to the lack of labeled data, the target generator is usually obtained by unsupervised learning with indirect loss functions, which often leads to long-haul and unstable training processes. In our previous work [11, 18, 36], we developed a training-free diffusion model to generate labeled for the target generator F in Eq. (3), such that F can be trained with supervised learning. Even though supervised learning is more stable and easier to handle than unsupervised learning, the training process still needs to be fine-tuned by choosing appropriate hyper-parameters, e.g., learning rate, the number of neurons, the number of layers, regularization terms, and early stopping criteria. Neural network models may be a good choice for very high-dimensional problems. When the dimension d is moderately high, the traditional sparse polynomial approximation method has significant advantages over neural network models because no sophisticated training process is needed. Thus, we will

extend our previous work by replacing the neural network model with sparse grid interpolation.

3. Methodology.

This section includes the details of the proposed approach. We will recall the standard score-based diffusion model with the training-free score estimation scheme in Section 3.1, and the construction of sparse-grids interpolation in 3.2.

3.1. The training-free score-based diffusion model.

The key idea of the score-based diffusion model [30] is to perturb the data distribution to a tractable reference distribution by gradually adding random noise, and then iteratively denoise to convert the reference distribution back to the target data distribution. These steps are referred to as the forward process and the backward process, respectively. In this study, we define the forward process using a forward stochastic differential equation (SDE) for $t \in [0,1]$:

$$dZ_t = b(t)Z_t dt + \sigma(t)dW_t \quad \text{with } Z_0 = Y \text{ and } Z_1 = X, \tag{4}$$

where the initial state Z_0 is the target variable Y and the terminal state Z_1 is the reference variable X. Here, W_t is a standard d-dimensional Brownian motion, $b(t)$ and $\sigma(t)$ are the drift and diffusion coefficients, respectively. Specifically, we consider the tractable reference variable X to follow the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I}_d)$. According to [12, 30], by appropriately choosing the drift coefficient $b(t)$ and the diffusion coefficient $\sigma(t)$ in Eq. (4), we ensure that the terminal state distribution is a standard Gaussian distribution $p(Z_1) = \mathcal{N}(0, \mathbf{I}_d)$. In particular, we define $b(t)$ and $\sigma(t)$ as follows:

$$b(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t} \quad \text{and} \quad \sigma^2(t) = \frac{\mathrm{d}\beta_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\beta_t^2, \tag{5}$$

where the α_t and β_t are given by:

$$\alpha_t = 1 - t, \quad \beta_t^2 = t \quad \text{for} \quad t \in [0,1]. \tag{6}$$

The solution to the forward SDE in Eq. (4) is given by

$$Z_t = Z_0 \exp\left[\int_0^t b(s)ds\right] + \int_0^t \exp\left[\int_s^t b(r)dr\right]\sigma(s)dW_s. \tag{7}$$

Thus, for any fixed $Z_0 = z_0$, the conditional probability density function $p_{Z_t|Z_0}(z_t|z_0)$ follows the Gaussian distribution, i.e,

$$p_{Z_t|Z_0}(z_t|z_0) = \phi_{(\alpha_t z_0, \beta_t^2\mathbf{I}_d)}(z_t|z_0), \tag{8}$$

where $\phi_{(\alpha_t z_0, \beta_t^2\mathbf{I}_d)}$ is the probability density function of the Gaussian distribution with mean $\alpha_t z_0$ and covariance matrix $\beta_t^2\mathbf{I}_d$. More importantly, when t approaches 1, the conditional distribution $p_{Z_t|Z_0}(z_t|z_0)$ converges to the standard Gaussian distribution. Hence, by using the forward SDE in Eq. (4), the target distribution $p_Y(y)$ is transformed to the standard Gaussian distribution $\phi_{(0,\mathbf{I}_d)}(x)$.

The forward SDE provides the "normalizing" transport, i.e., mapping the target distribution to the standard normal distribution. To build the generator F in Eq. (3), we also need the associated backward SDE running from $t = 1$ to $t = 0$. Specifically, the backward SDE is defined by

$$dZ_t = \left[b(t)Z_t - \sigma^2(t)S(Z_t, t)\right]dt + \sigma(t)dB_t \quad \text{with } Z_0 = Y \text{ and } Z_1 = X, \tag{9}$$

where B_t is the backward Brownian motion, and the score function $S(z_t, t)$, defined as the gradient of the log of the probability density function:

$$S(z_t, t) := \nabla_z \log p_{Z_t}(z_t), \tag{10}$$

is unknown. Traditional neural network approaches, such as denoising score matching, and sliced score matching, are widely used to approximate the score function $S(z_t, t)$. Once the score function is learned, unlimited samples of the target distribution can be generated by simulating the backward SDE, starting with a final state sample from the Gaussian distribution. Although these methods are effective, they are computationally demanding due to several reasons. First, learning the score function in an unsupervised manner requires the storage of a substantial number of stochastic trajectories of the forward SDE as training data. Second, generating a single sample of the target distribution involves solving the discretized backward SDE over thousands of time steps.

To avoid learning the score function using neural networks, we use the direct Monte Carlo estimator to approximate the closed-form expression of the score function [18]. Specifically, the score function defined in Eq. (10) can be reformulated as

$$\begin{aligned} S(z_t, t) &= \nabla_z \log \left(\int_{\mathbb{R}^d} p_{Z_t|Z_0}(z_t|z_0) p_{Z_0}(z_0) dz_0 \right) \\ &= \frac{1}{\int_{\mathbb{R}^d} p_{Z_t|Z_0}(z_t|z_0') p_{Z_0}(z_0') dz_0'} \int_{\mathbb{R}^d} -\frac{z_t - \alpha_t z_0}{\beta_t^2} p_{Z_t|Z_0}(z_t|z_0) p_{Z_0}(z_0) dz_0 \\ &= \int_{\mathbb{R}^d} -\frac{z_t - \alpha_t z_0}{\beta_t^2} w(z_t, z_0) p_{Z_0}(z_0) dz_0, \end{aligned} \tag{11}$$

where the weight function $w(z_t, z_0)$ is

$$w(z_t, z_0) := \frac{p_{Z_t|Z_0}(z_t|z_0)}{\int_{\mathbb{R}^d} p_{Z_t|Z_0}(z_t|z_0') p_{Z_0}(z_0') dz_0'} \quad \text{and} \quad \int_{\mathbb{R}^d} w(z_t, z_0) p_{Z_0}(z_0) dz_0 = 1. \tag{12}$$

In the implementation, we utilize the Monte Carlo estimator to approximate the integrals over the initial state Z_0 in the reformulated score function in Eq. (11). Specifically, the score function can be approximated by

$$\tilde{S}(z_t, t) := \sum_{n=1}^{N} -\frac{z_t - \alpha_t y_{k_n}}{\beta_t^2} \tilde{w}(z_t, y_{k_n}), \tag{13}$$

where $\{y_{k_n}\}_{n=1}^N$ is a mini-batch of the dataset $\mathcal{Y}$ in Eq. (2) with size $N \leq K$. The weight function $w(z_t, y_{k_n})$ is approximated by

$$\tilde{w}(z_t, y_{k_n}) := \frac{p_{Z_t|Z_0}(z_t|y_{k_n})}{\sum_{m=1}^{N} p_{Z_t|Z_0}(z_t|y_{k_m})}, \tag{14}$$

where $p_{Z_t|Z_0}(z_t|y_{k_n})$ is the Gaussian distribution given in Eq. (8). The size N of the mini-batch can be adjusted to balance between computational efficiency and estimation accuracy. This approach does not require any training process, e.g., stochastic gradient descent, to approximate the score function. However, solving the backward SDEs is not useful for building the generator F using sparse grid interpolation, which will be addressed in the next subsection.

3.2. **The sparse-grid generative model.** In this section, we describe the sparse grid model we use for constructing the transport map F in Eq. (3). We are following the approach in [31] where we are using linear functions with local support and equidistant interpolation nodes. Higher order approximation can be constructed analogously. Starting with the one dimensional context, we take the domain $[-a, a]$ and n interpolation nodes $x_1, x_2, \cdots x_n \in [-a, a]$ with corresponding support $\Delta x_1, \Delta x_2, \cdots, \Delta x_n$. The linear basis functions are defined so that they are equal to 1 at the interpolation nodes and 0 outside of the support, i.e.,

$$\phi_{x_i,\Delta x_i}(x) = \begin{cases} 1 - \frac{|x|}{\Delta x_i}, & |x - x_i| \leq \Delta x_i \\ 0, & |x - x_i| > \Delta x_i. \end{cases} \tag{15}$$

We adopt a hierarchical approach where we group the nodes into levels, so that at level 1 we have $x_1 = 0$ and $\Delta x_1 = a$. On level 2 we add two mode nodes x_2 and x_3 so that each node is at an equal distance between x_1 and the edge of the domain, i.e., $x_{2/3} = \pm\frac{a}{2}$, and the support is half of the support on the previous level, i.e., $\Delta x_{2/3} = \frac{a}{2}$. Following the same pattern, at one level of 3 we add 4 more points interlaced between the existing set of points and the end points of the domain

$$x_4 = -\frac{3a}{4}, \qquad x_5 = -\frac{a}{4}, \qquad x_6 = \frac{a}{4}, \qquad x_7 = \frac{3a}{4}, \tag{16}$$

while the support is halved again $\Delta x_{4,5,6,7} = \frac{a}{4}$. The pattern continues so that for $l > 1$ the total number of points is $2^l - 1$ and the support at the finest level is $\Delta x = a2^{-l}$. Fig. 1 shows the first 7 nodes and functions for linear, quadratic, and cubic basis. For convenience, we label the basis with a single index so that

$$\phi_i(x) = \phi_{x_i,\Delta x_i}(x) \tag{17}$$

and we aim to construct an approximation to a target function.

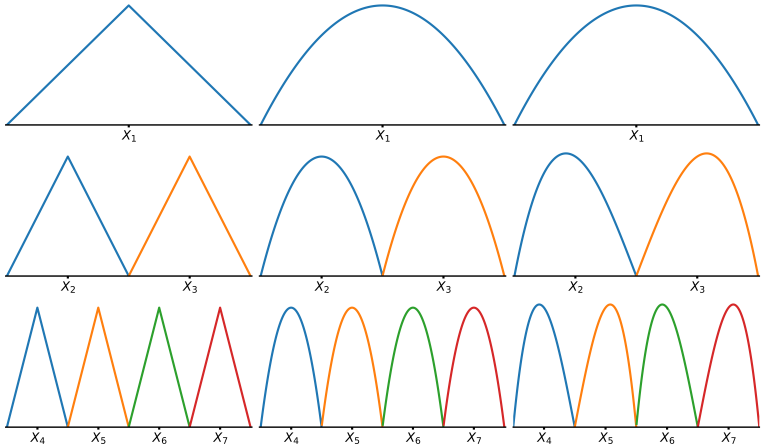

FIGURE 1. Semi-local polynomial points and basis functions, left to right: linear, quadratic, and cubic functions.

Given a level l and input-output pairs $\{(x_1, \bar{y}_1), (x_2, \bar{y}_2) \cdots (x_J, \bar{y}_J)\}$ for $J = 2^l - 1$ we can construct the approximation $F_l(x) \approx F(x)$

$$F_l(x) = \sum_{j=1}^{J} c_j \phi_j(x). \tag{18}$$

The interpolation coefficients are defined so that $F_l(x)$ is an interpolant, i.e., $F_l(x_i) = \bar{y}_i$. Observe that by definition of the reduced support, for any indexes i and j with $i < j$ we have that $\phi_j(x_i) = 0$, while $\phi_i(x_i) = 1$. Therefore, $\phi_1(x)$ is the only basis functions with support at x_1 and setting $F_l(x_1) = \bar{y}_1$ we have

$$F_l(x_1) = \sum_{j=1}^{J} c_j \phi_j(x_1) = c_1 \phi_1(x_1) = c_1 = \bar{y}_1. \tag{19}$$

For $i > 1$ we can find the coefficients recursively

$$F_l(x_i) = \sum_{j=1}^{i} c_j \phi_j(x_i) = \bar{y}_i, \qquad \Longrightarrow \qquad c_i = \bar{y}_i - \sum_{j=1}^{i-1} c_j \phi_j(x_i). \tag{20}$$

The resulting system of equations has additional sparsity and many algorithms have been studied in the literature [5,15] for efficiently computing $\{c_1, c_2, \cdots, c_n\}$. In this work, we use the ORNL developed Toolkit for Adaptive Stochastic Modeling and Non-Intrusive Approximation (Tasmanian) [32] which is a highly efficient library with support for multi-core CPUs and GPU accelerators from all major vendors (Nvidia, AMD, and Intel), as well as easy to use bindings for multiple programming languages including C++, Python, Matlab, and Fortran.

Extending the interpolation approach to a multidimensional context is challenging. The straightforward way to extend a one dimensional interpolation strategy to arbitrary d dimensions is through tensoring on one dimensional rules. Following common sparse grid notation [5, 9, 15] we define d-dimensional multi-indexes $(i_1, i_2, \cdots, i_d)$ and define the multi-dimensional interpolant:

$$F_{(l_1,l_2,\cdots,l_d)}(x_1, x_2, \cdots, x_d) = \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_d=1}^{J_d} c_{(j_1,j_2,\cdots,j_d)} \phi_{j_1}(x_1)\phi_{j_2}(x_2)\cdots\phi_{j_d}(x_d). \tag{21}$$

The multidimensional interpolation points have the form $(x_{j_1}, x_{j_2}, \cdots, x_{j_d})$ and tensors of $\phi_j(x)$ basis functions satisfy similar properties to the one dimensional ones. Thus, we can still apply a version to (20), where the ordering of the indexes has to obey a multi-dimensional hierarchy. The Tasmanian library handles all the necessary bookkeeping and details of the algorithms can be found in [31,33].

The choice of one dimensional levels $(l_1, l_2, \cdots, l_d)$ is a key component of every efficient sparse grid algorithm. In the single dimensional case (18) for any level l we have $J = 2^l - 1$. The approximation error of the linear interpolant is given by [5, 20, 25]

$$\max_x |F(x) - F_l(x)| \leq C \Delta x_l^2, \tag{22}$$

where $\Delta x_l = \min_j \Delta x_j = \frac{1}{J+1} = 2^{-l}$ and C is a constant that depends on the smoothness of $F(x)$. Saying this in another way, our error is bounded by the support of the excluded basis functions. In a one dimensional context, we select a specific level L such that Δx_L is sufficiently small.

In a multidimensional context, the error is related to the volume of the support, i.e., the product of the single dimensional support Δx_l

$$\max_{(x_1,x_2,\cdots,x_d)} \left| F - F_{(l_1,l_2,\cdots,l_d)} \right| \leq C \left(\Delta x_{l_1} \Delta x_{l_2} \cdots \Delta x_{l_d} \right)^2 \tag{23}$$

Therefore, if our goal is to obtain an error of order 2^{-L} (for some L), then we need to include in our approximation all basis functions that have a volume larger than

2^{-L}. Considering the combinations of $(l_1, l_2, \cdots, l_d)$ that yield such basis, we can substitute 2^{-l} for each Δx_l and obtain the inequality

$$2^{-l_1} 2^{-l_2} \cdots 2^{-l_d} \geq 2^{-L}, \tag{24}$$

i.e. $\sum_{k=1}^{d} l_k \leq L$. A naïve approach would be to assign the same value of L to all levels $(l_1, l_2, \cdots, l_d)$ and that would guarantee that we have included all the necessary basis function; however, this is wasteful as it will also include many superfluous bases with much smaller support. The sparse grid approach uses (24) to select only the minimum required set of basis functions to construct an interpolant with the desired accuracy. We do this by defining the sparse set of basis functions

$$\Lambda_L = \left\{ (j_1, j_2, \cdots, j_d) : \text{there are } (l_1, l_2, \cdots, l_d) \text{ with } j_k \leq 2^{l_k} - 1 \text{ and } \sum_k l_k \leq L \right\} \tag{25}$$

and the sparse grid interpolant is given by

$$F_L(x) = \sum_{(j_1, j_2, \cdots, j_d) \in \Lambda_L} c_{(j_1, j_2, \cdots, j_d)} \phi_{j_1}(x_1) \phi_{j_2}(x_2) \cdots \phi_{j_d}(x_d). \tag{26}$$

The Tasmanian library can find all the necessary basis functions for any user provided L and do all of the internal bookkeeping of the coefficients. However, for Tasmanian to compute the coefficients we must first find all the input-output pairs at all the sample points.

Stepping away from the multi-index notation (we leave that to the software), we need outputs for a set of sparse grid points $\{x_1, x_2, \ldots, x_J\}$ that are not the same as the input-output pairs $\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_K\}$ and $\{y_1, y_2, \ldots, y_K\}$, where $\tilde{x}$ follows standard Gaussian and y's are from the dataset $\mathcal{Y}$ in Eq. (2). Even though the backward SDE can be solved using the scheme in Section 3.1, the backward SDE cannot provide the desired input-output pairs. To address this issue, we instead solve the corresponding backward ODE, also known as the probability flow ODE, i.e.,

$$dZ_t = \left[b(t) Z_t - \frac{1}{2} \sigma^2(t) S(Z_t, t) \right] dt \quad \text{with} \quad Z_0 = Y \text{ and } Z_1 = X, \tag{27}$$

whose trajectories share the same marginal probability density functions as the backward SDE in Eq. (9). The backward ODE defines a much smoother function relationship between the initial state and the terminal state than that defined by the backward SDE, such that we can solve the ODE to help generate the input-output pairs $\{(x_1, \bar{y}_1), (x_2, \bar{y}_2) \cdots (x_J, \bar{y}_J)\}$ needed by the sparse grid interpolant in Eq. (18).

Specifically, we generate J sparse grid points $\{x_1, x_2, \ldots, x_J\}$ using the procedure above and observe that those are equidistant and do not follow the standard Gaussian distribution. Nevertheless, for $j = 1, \ldots, J$, we solve the backward ODE in Eq. (27) from $t = 1$ to $t = 0$ and collect the state $Z_0|x_j$, where the score function is computed using Eq. (13), Eq. (14), and the dataset $\mathcal{Y} = \{y_1, \ldots, y_K\}$ in Eq. (2). The generated input-output pairs are denoted by

$$\mathcal{D} := \{(x_j, \bar{y}_j) : \bar{y}_j = Z_0|x_j, \quad \text{for} \quad j = 1, \ldots, J\}, \tag{28}$$

where $\bar{y}_j$ is obtained by solving the ODE in Eq. (27). We note that the $\bar{y}_j$'s in $\mathcal{D}$ may not belong to $\mathcal{Y}$ and that J could be arbitrarily large. After obtaining $\mathcal{D}$ we can use it to interpolate the transport map $F(y)$ in Eq. (3) using spare grids.

Algorithm 1: Pseudo-algorithm for constructing the sparse-grid-based generative model

1: **Input**: the observation data $\mathcal{Y}$, the diffusion coefficient $\sigma(t)$, the drift coefficient $b(t)$;
2: **Output**: sparse-grid generative model $F(x)$;
3: Generate sparse grid points $\{x_1, \ldots, x_J\}$ using Tasmanian;
4: **for** $j = 1, \ldots, J$
5: Solve the ODE in Eq. (27) with the score function estimated by Eq. (13) and Eq. (14);
6: Define a pair of labeled data $(x_j, \bar{y}_j)$ where $x_j = Z_1$ and $\bar{y}_j = Z_0$ in Eq. (27);
7: **end**
8: Interpolate the transport map $y = F(x)$ using dataset $\mathcal{D}$.

4. Numerical experiments.

4.1. **1D example: Gaussian mixture model (GMM).** We consider the probability density function defined by

$$p_Y(y) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{N}(y; \mu_i, \Sigma_i), \tag{29}$$

where $m = 2$ is the number of Gaussian modes, $\mu = [-1, 1]$ denotes the mean values of the two modes, and Σ_i are the covariance matrices. We consider two scenarios. The first is connected modes with $\Sigma_i = 0.3\mathbf{I}$ and the second is well-separated modes with $\Sigma_i = 0.05\mathbf{I}$.

For the connected modes, the size of the training data in Eq. (2) is 600,000. We use sparse grids with levels of $\{3, 4, 5, 6\}$ to interpolate the labeled data generated by solving the backward ODE in Eq. (27), which corresponds to grid size $\{15, 31, 63, 127\}$, respectively. The density approximation is displayed in Fig. 2. The KL divergence between the exact PDF and approximated PDF can be seen in Fig. 3. We can observe that as the level of the sparse grid increases, the interpolation becomes more accurate. Furthermore, Fig. 4 presents the KL divergence and log KL divergence between true PDF and estimated PDF with sample sizes of $\{6000, 60000, 600000\}$. We can observe that larger training data improves the quality of density estimation.

For GMM with well-separated modes, we use 2,000,000 samples for the training set in Eq. 2. We use sparse grids with levels of $\{3, 4, \cdots, 9\}$ to interpolate the labeled data generated by solving the backward ODE in Eq. 27, which corresponds to grid size $\{15, 31, \cdots, 1023\}$, respectively. The probability density function approximation is displayed in Fig. 5. The KL divergence between the exact PDF and approximated PDF can be seen in Fig. 6. One can observe that as the level of the sparse grid increases, the interpolation becomes more accurate.

4.2. **2D example: Banana distribution.** We consider the following probability density function:

$$p_Y(y) \propto \exp\left\{-10\left(y_1^2 - y_2\right)^2 - \left(y_2 - \frac{1}{4}\right)^4\right\}. \tag{30}$$

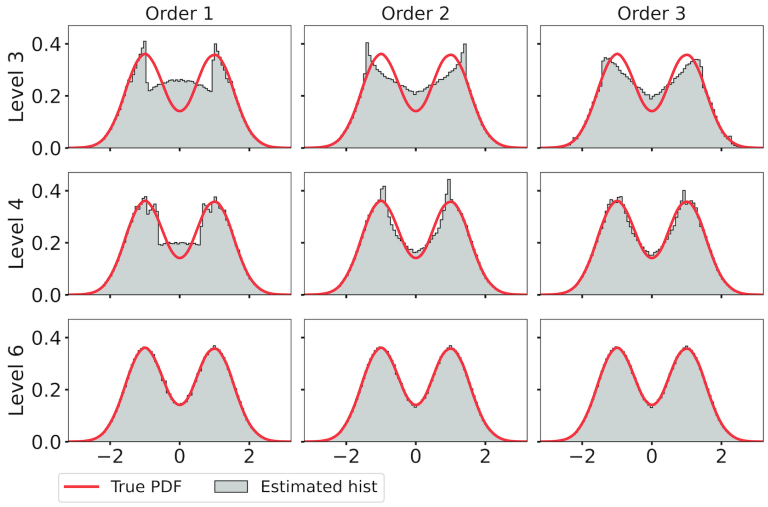

FIGURE 2. Comparison of the true PDF and estimated histogram for 1D GMM with two connected modes. From left to right: local polynomial grids with polynomial orders $\{1, 2, 3\}$. From top to bottom: local polynomial grids with levels $\{3, 4, 6\}$. The red line represents the exact PDF. The grey histogram is based on the samples generated by the sparse-grid generative model F.

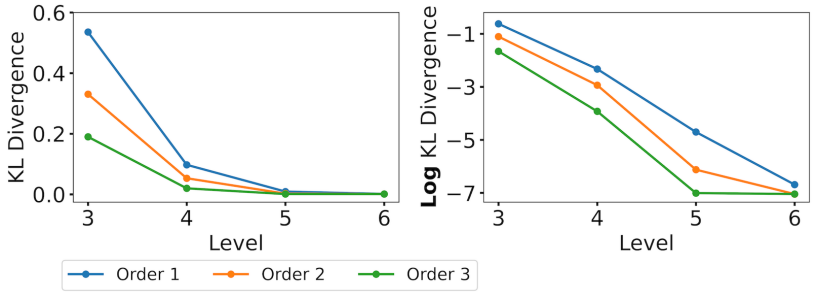

FIGURE 3. KL divergence and log KL divergence between true PDF and estimated PDF with orders $\{1, 2, 3\}$ and sparse grid levels $\{3, 4, 5, 6\}$ for the 1D GMM with connected modes.

In this example, we have 2,000,000 samples for the training dataset in Eq. (2). We use sparse grids with levels of $\{3, 4, \cdots, 9\}$ to interpolate the labeled data generated by solving the backward ODE, which corresponds to grid size $\{49, 129, \cdots, 9217\}$, respectively. Fig. 7 shows the sparse grid in 2D. After we obtain the interpolation using the sparse grid, we generate 2,000,000 new samples by pushing the Gaussian samples into the interpolation function. The probability density function approximation of the generated new samples is displayed in Fig. 9. The KL divergence between the exact PDF and approximated PDF can be seen in Fig. 10. One can observe that as the level of the sparse grid increases, the interpolation becomes more accurate.

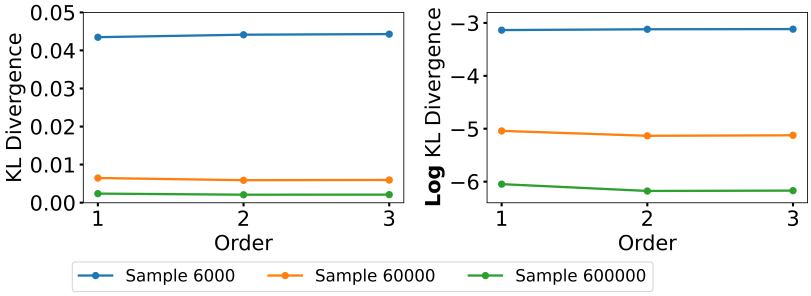

FIGURE 4. KL divergence and log KL divergence between true PDF and estimated PDF with orders $\{1, 2, 3\}$ and training data sample sizes $\{6000, 60000, 600000\}$ for the 1D GMM with connected modes.

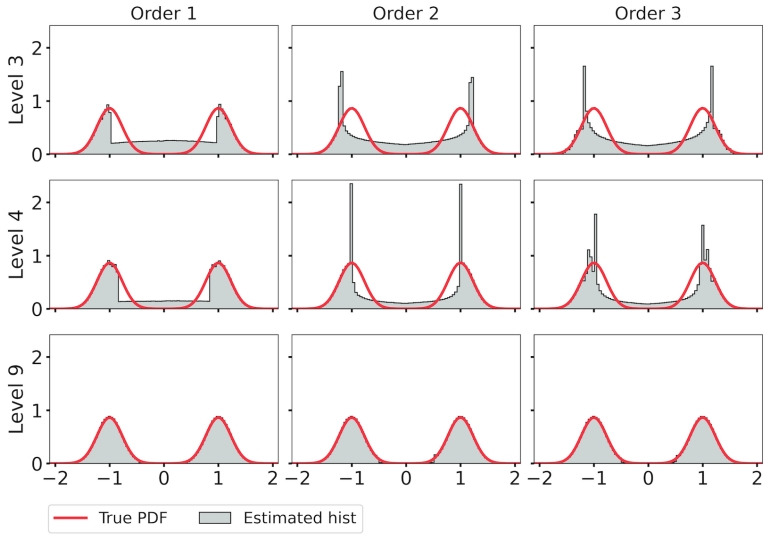

FIGURE 5. Comparison of the true PDF and estimated histogram for 1D well-separated GMM. From left to right: local polynomial grids with orders $\{1, 2, 3\}$. From top to bottom: local polynomial grids with levels $\{3, 4, 9\}$. The red line represents the exact PDF. The grey histogram is the approximated PDF by local polynomial grids.

4.3. **5D example: Funnel distribution.** We consider the following probability density function:

$$p_Y(y) = \mathcal{N}(y_1; 0, \eta^2) \prod_{i=2}^{5} \mathcal{N}(y_i; 0, e^{y_1}), \tag{31}$$

where $\eta = 1$. In this example, we have 50,000,000 samples in the training set in Eq. (2). Due to the computation complexity. We use sparse grids with levels of $\{2, 3, 4\}$ to interpolate the labeled data, which corresponds to grid size $\{71, 351, 1471\}$, respectively. For each level, 2,000,000 new samples are generated by the

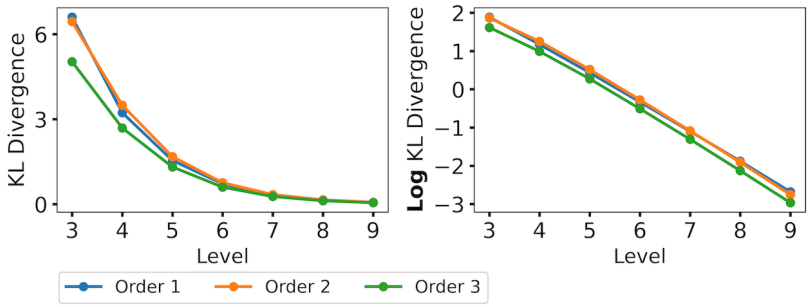

FIGURE 6. 1D well-separated GMM KL divergence and log KL divergence between true PDF and estimated PDF with orders $\{1, 2, 3\}$ and levels $\{3, 4, \cdots, 9\}$.

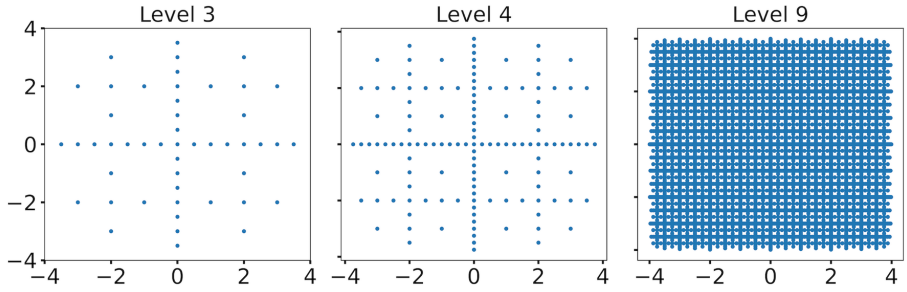

FIGURE 7. Grids of the 2D local polynomial interpolation with levels $\{3, 4, 9\}$.

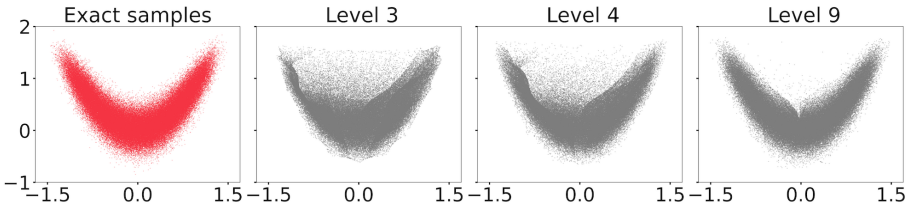

FIGURE 8. Comparison of the banana distribution using a scatter plot of the exact density function and samples obtained by the sparse grid generative model F of order 1 with levels $\{3, 4, 9\}$.

corresponding sparse grid generative model F. The approximated PDF based on the generated new samples is displayed in Fig. 11. The KL divergence between the exact PDF and approximated PDF can be seen in Fig. 12. One can observe that as the level of the sparse grid increases, the interpolation becomes more accurate.

5. **Conclusions.** This work has successfully demonstrated a novel approach to density estimation using a diffusion-based sparse-grid generative model. The integration of score-based diffusion modeling with sparse grid interpolation presents an advancement in generative modeling, particularly in scenarios where labeled data is scarce. Our method circumvents the complexities and limitations associated with

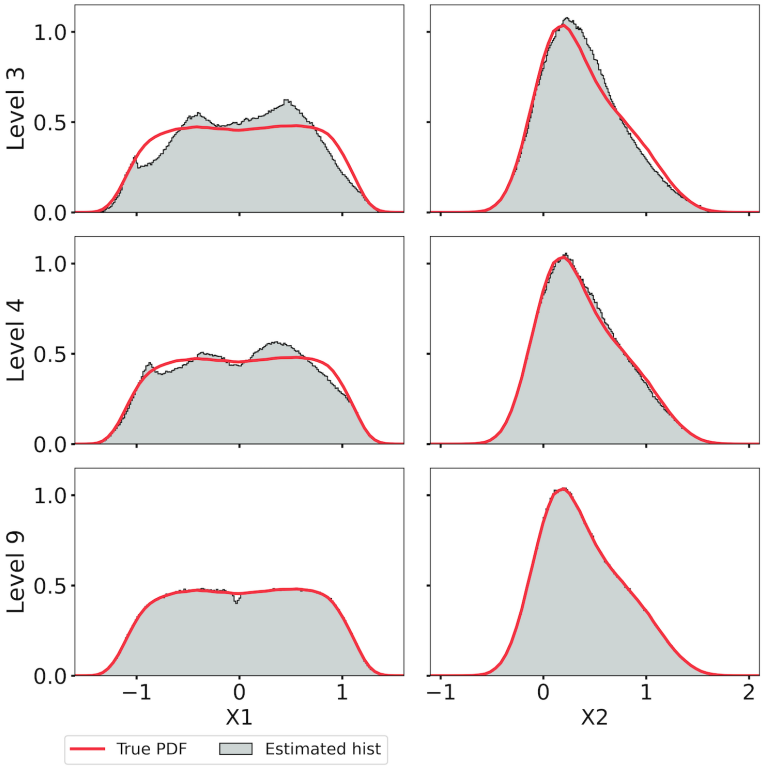

FIGURE 9. Comparison of the true PDF and estimated histogram for the banana distribution, using local polynomial grids of order 1 with levels $\{3, 4, 9\}$. From left to right: x_1 and x_2. From top to bottom: local polynomial grids with levels $\{3, 4, 9\}$. The red line represents the exact PDF. The grey histogram is the approximated PDF by local polynomial grids.

traditional neural network training by utilizing a training-free scheme that directly approximates the score function. The application of the Tasmanian library has enabled the construction of an efficient and accurate model capable of handling multi-dimensional data distributions. The numerical experiments conducted on various datasets, ranging from simple 1D Gaussian mixtures to complex 5D distributions, have showcased the model's ability to generate high-quality approximations of the target distributions, improving with the refinement of the sparse grid levels. This approach not only enhances computational efficiency but also provides a robust framework for further exploration and application in more diverse and challenging domains. The promising results affirm the potential of integrating machine learning techniques with traditional numerical methods, setting a precedent for future research in generative modeling and density estimation.

Author contributions. Yanfang Liu: Conceptualization, data curation, methodology, validation, software, writing-original draft, writing-review/editing. Alisa Bryantseva: Data curation, methodology, software, visualization, writing-original draft. Miroslav Stoyanov: Conceptualization writing-original draft, writing-review/

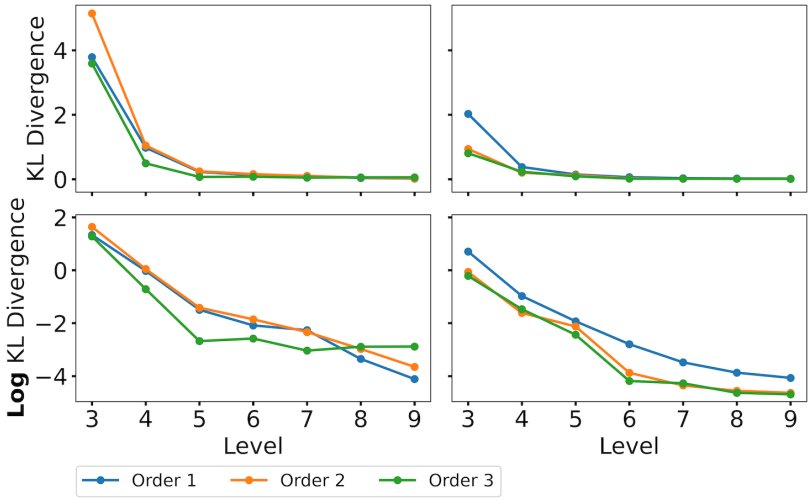

FIGURE 10. Banana distribution KL divergence and log KL divergence between true marginal PDF and estimated marginal PDF with sparse grid orders $\{1, 2, 3\}$ and levels $\{3, 4, \cdots, 9\}$. From left to right: x_1 and x_2. From top to bottom: KL divergence and log KL divergence.

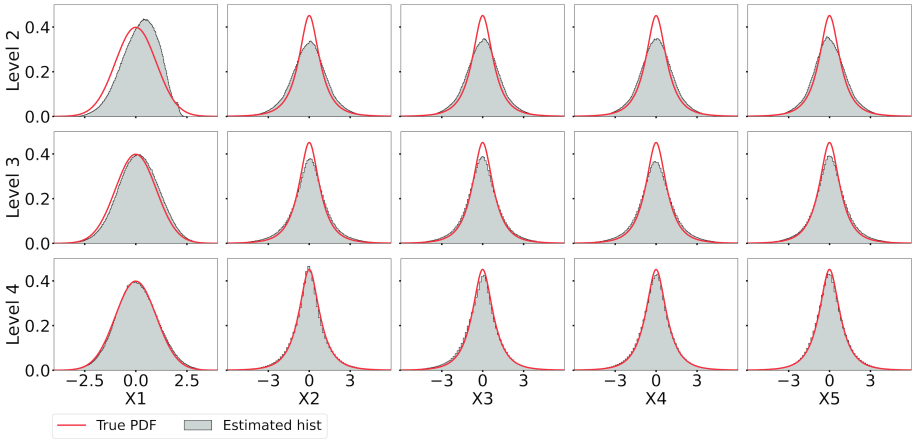

FIGURE 11. Comparison of the true PDF and estimated histogram for a funnel distribution, using local polynomial grids of order 1 with levels $\{2, 3, 4\}$. From left to right: x_1, x_2, $\cdots$, x_5. From top to bottom: local polynomial grids with levels $\{2, 3, 4\}$. The red line represents the exact PDF. The grey histogram is the approximated PDF by local polynomial grids.

editing. Feng Bao: Conceptualization, methodology. Guannan Zhang: Conceptualization, methodology, writing-original draft, writing-review/editing, funding acquisition.

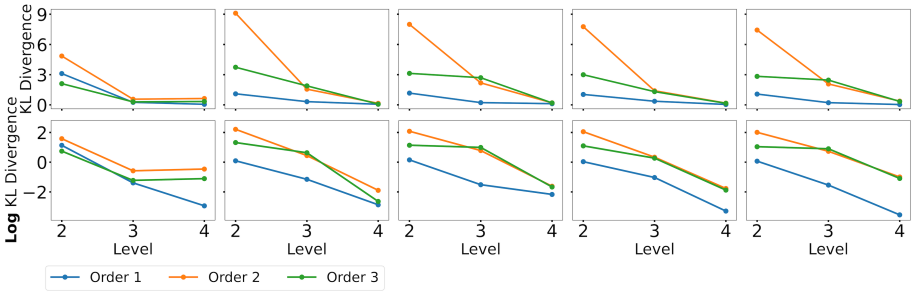

FIGURE 12. Funnel distribution KL divergence and log KL divergence between true marginal PDF and estimated marginal PDF with sparse grid orders $\{1, 2, 3\}$ and levels $\{2, 3, 4\}$. From left to right: x_1, x_2, $\cdots$, x_5. From top to bottom: KL divergence and log KL divergence.

Use of AI tools declaration. The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments. This work is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program, under the contract ERKJ388, ERKJ443, and accomplished at Oak Ridge National Laboratory (ORNL), and under Grant DE-SC0022254. ORNL is operated by UT-Battelle, LLC., for the U.S. Department of Energy under Contract DE-AC05-00OR22725. The first author (FB) would also like to acknowledge the support from U.S. National Science Foundation through project DMS-2142672 and the support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Grant DE-SC0022297.

Conflict of interest. There is no conflict of interest.

REFERENCES

[1] J. Austin, D. D. Johnson, J. Ho, D. Tarlow and R. van den Berg, Structured denoising diffusion models in discrete state-spaces, in *Advances in Neural Information Processing Systems*, NeurIPS 2021, virtual, **34** (2021), 17981-17993.

[2] F. Bao, Z. Zhang and G. Zhang, An ensemble score filter for tracking high-dimensional nonlinear dynamical systems, *Computer Methods in Applied Mechanics and Engineering*, **432** (2024), 117447.

[3] F. Bao, Z. Zhang and G. Zhang, A score-based nonlinear filter for data assimilation, *Journal of Computational Physics*, **514** (2024), Paper No. 113207, 16 pp.

[4] R. Baptista, Y. Marzouk and O. Zahm, On the representation and learning of monotone triangular transport maps, *Foundations of Computational Mathematics*, Springer, (2023), 1-46.

[5] H.-J. Bungartz and M. Griebel, Sparse grids, *Acta Numerica*, **13** (2004), 147-269.

[6] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. J. Belongie, N. Snavely and B. Hariharan, Learning gradient fields for shape generation, in *ECCV 2020, Part III*, Lecture Notes in Computer Science, Springer, **12348** (2020), 364-381.

[7] T. Chakraborty, U. K. Reddy, S. M. Naik, M. Panja and B. Manvitha, Ten years of generative adversarial nets (GANs): A survey of the state-of-the-art, *Machine Learning: Science and Technology*, IOP Publishing, **5** (2024), 011001.

[8] C. Chen, C. Li, L. Chen, W. Wang, Y. Pu and L. Carin, Continuous-time flows for efficient inference and density estimation, in *International Conference on Machine Learning*, PMLR, (2018), 824-833.
[9] A. Chkifa, A. Cohen and C. Schwab, High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs, *Foundations of Computational Mathematics*, **14** (2014), 601-633.
[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, *Generative Adversarial Nets*, in Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc., 2014.
[11] A. Haisam, J. Yin, Y. Geng, S. Liang, F. Bao, L. Ju and G. Zhang, *A Scalable Training-Free Diffusion Model for Uncertainty Quantification*, in Proceedings of the IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), IEEE, Denver, CO, USA, 2024.
[12] J. Ho, A. Jain and P. Abbeel, Denoising diffusion probabilistic models, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., **33** (2020), 6840-6851.
[13] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré and M. Welling, Argmax flows and multinomial diffusion: Learning categorical distributions, in *Advances in Neural Information Processing Systems*, NeurIPS 2021, virtual, **34** (2021), 12454-12465.
[14] A. Hyvärinen, Estimation of non-normalized statistical models by score matching, *Journal of Machine Learning Research*, **6** (2005), 695-709.
[15] J. D. Jakeman and S. G. Roberts, Local and dimension adaptive stochastic collocation for uncertainty quantification, in *Sparse Grids and Applications*, Springer, **88** (2012), 181-203.
[16] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, in 2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings, Banff, AB, Canada, 2014.
[17] I. Kobyzev, S. J. D. Prince and M. A. Brubaker, Normalizing flows: An introduction and review of current methods, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, **43** (2021), 3964-3979.
[18] Y. Liu, M. Yang, Z. Zhang, F. Bao, Y. Cao and G. Zhang, Diffusion-model-assisted supervised learning of generative models for density estimation, *Journal of Machine Learning for Modeling and Computing*, **5** (2024), 25-38.
[19] D. Lu, Y. Liu, Z. Zhang, F. Bao and G. Zhang, A diffusion-based uncertainty quantification method to advance E3SM land model calibration, *Journal of Geophysical Research: Machine Learning and Computation*, **1** (2024), e2024JH000234.
[20] X. Ma and N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, *Journal of Computational Physics*, Elsevier, **228** (2009), 3084-3113.
[21] D. Onken, S. W. Fung, X. Liang and L. Ruthotto, OT-flow: Fast and accurate continuous normalizing flows via optimal transport, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **35** (2021), 9223-9232.
[22] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, *The Journal of Machine Learning Research*, **22** (2021), Paper No. 57, 64 pp.
[23] G. Papamakarios, T. Pavlakou and I. Murray, *Masked Autoregressive Flow for Density Estimation*, in Advances in Neural Information Processing Systems, vol. 30, 2017.
[24] M. Parno, P.-B. Rubio, D. Sharp, M. Brennan, R. Baptista, H. Bonart and Y. Marzouk, Mpart: Monotone parameterization toolkit, *Journal of Open Source Software*, **7** (2022), 4843.
[25] D. Pflüger, *Spatially Adaptive Sparse Grids for High-Dimensional Problems*, Verlag Dr. Hut, Munich, 2010.
[26] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, Improved techniques for training GANs, in *Advances in Neural Information Processing Systems*, NeurIPS, **29** (2016), 2226-2234.
[27] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth and G. Langs, Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, in *Information Processing in Medical Imaging, IPMI 2017*, Lecture Notes in Computer Science, Springer, **10265** (2017), 146-157.
[28] Y. Song and S. Ermon, *Generative Modeling by Estimating Gradients of the Data Distribution*, in Advances in Neural Information Processing Systems, vol. 32, 2019.

[29] Y. Song, S. Garg, J. Shi and S. Ermon, Sliced score matching: A scalable approach to density and score estimation, in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, PMLR*, **115** (2020), 574-584.

[30] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, *Score-Based Generative Modeling Through Stochastic Differential Equations*, in International Conference on Learning Representations, 2021.

[31] M. Stoyanov, Adaptive sparse grid construction in a context of local anisotropy and multiple hierarchical parents, in *Sparse Grids and Applications–Miami 2016*, Springer, **123** (2018), 175-199.

[32] M. Stoyanov, D. Lebrun-Grandie, J. Burkardt and D. Munster, *Tasmanian*, **9**, 2013. Available from: `https://github.com/ORNL/Tasmanian`.

[33] M. K. Stoyanov and C. G. Webster, A dynamically adaptive sparse grids method for quasi-optimal interpolation of multidimensional functions, *Computers & Mathematics with Applications*, Elsevier, **71** (2016), 2449-2465.

[34] P. Vincent, A connection between score matching and denoising autoencoders, *Neural Comput.*, MIT Press, **23** (2011), 1661-1674.

[35] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui and M.-H. Yang, Diffusion models: A comprehensive survey of methods and applications, *ACM Comput. Surv.*, **56** (2023), Article No.: 105, 1-39.

[36] J. Yin, S. Liang, S. Liu, F. Bao, H. G. Chipilski, D. Lu and G. Zhang, *A Scalable Real-Time Data Assimilation Framework for Predicting Turbulent Atmosphere Dynamics*, in Proceedings of the IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), IEEE, Atlanta, GA, USA, 2024.

Received July 2024; revised November 2024; early access December 2024.