Transport-Based Generative Modeling for Stochastic Dynamics, Uncertainty Quantification, and High-Dimensional Sampling

by

Pengjun Wang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 8, 2026

Keywords: generative model, normalizing flow, diffusion model, stochastic differential equation, uncertainty quantification, numerical analysis

Approved by

Yanzhao Cao, Chair, Professor of Mathematics
Guannan Zhang, Co-Chair, Distinguished Staff Scientist at Oak Ridge National Laboratory
Junshan Lin, Professor of Mathematics
Hans Werner Van Wyk, Associate Professor of Mathematics
Thi Thao Phuong Hoang, Associate Professor of Mathematics

Abstract

This dissertation develops and analyzes generative modeling methods for scientific computing, where repeated sampling from complex probability distributions is needed for uncertainty quantification, estimation of quantities of interest, and statistical inference. The main contributions are pseudo-reversible normalizing flows and training-free diffusion models, which provide two complementary transport-based approaches for generating samples from target distributions by transforming samples from simple reference distributions.

In the first part of the dissertation, we present a pseudo-reversible normalizing flow method for efficiently generating samples of the state of a stochastic differential equation (SDE) with various initial distributions. The main novelty of our normalizing flow model is that it can learn the conditional distribution of the state, i.e., the distribution of the final state conditional on any initial state, such that the model only needs to be trained once and the trained model can be used to handle various initial distributions. We provide a rigorous convergence analysis of the learned distribution to the target distribution in the Kullback–Leibler divergence metric. Numerical experiments are provided to demonstrate the effectiveness of the proposed normalizing flow model.

Next, we extend this framework to uncertainty quantification for noisy physical models. A conditional pseudo-reversible normalizing flow is constructed directly from input-output data to generate samples from conditional distributions relevant to forward and inverse uncertainty propagation. The method does not require prior knowledge of the deterministic model component or the distribution of additive noise. Theoretical results establish convergence for the forward conditional distribution, and numerical experiments include benchmark examples, high-dimensional uncertainty propagation, and a geologic carbon storage application.

In this dissertation, we also study a training-free diffusion model for high-dimensional sampling. A Gaussian mixture model constructed from available samples yields an analytically tractable score function and avoids propagating score function approximation errors through the reverse process. The resulting error bounds exhibit favorable dimension dependence, scaling as $\mathcal{O}(d)$ in the ℓ_2 norm and $\mathcal{O}(\log d)$ in the ℓ_∞ norm. Importantly, the proposed error estimates are fully numerically verifiable with respect to both time-step size and dimensionality, thereby bridging the gap between theoretical analysis and observed numerical behavior.

Artificial Intelligence (AI) Use Disclosure Statement

ChatGPT was used to assist with language editing, including grammar and clarity checks, and with coding support for numerical experiments. All mathematical proofs and theoretical arguments in this dissertation were completed by the author. The author designed the numerical experiments, reviewed the code and computational results, and takes full responsibility for the methods, analysis, and conclusions presented in this dissertation. All AI-generated content was reviewed and validated for relevance, appropriateness, and accuracy before incorporation into the final document to maintain the scholarly integrity of this research.

Digital Accessibility Disclosure Statement

In the preparation of this dissertation, the following digital accessibility tool was used to ensure that this document complies with federal requirements: PDF Accessibility Checker (PAC). The author acknowledges full responsibility for the intellectual content of this work and has made a good faith effort to comply with digital accessibility requirements in publishing, provided that doing so does not significantly change the nature of the content. Furthermore, all content has been reviewed and revised to meet these requirements prior to final publication.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Yanzhao Cao, for his unwavering support throughout my doctoral studies. His academic guidance, encouragement, and confidence in me have shaped both my research and personal growth. Beyond academia, I am especially grateful for his kindness and care in my daily life. Without Dr. Cao's encouragement and belief in me, I might never have had the opportunity to come to the United States, nor would I have been fortunate enough to meet so many wonderful people during this journey.

My sincere appreciation also goes to my co-advisor, Dr. Guannan Zhang. Dr. Zhang helped guide me toward my current research direction and introduced me to a broader and richer mathematical world. Through his mentorship, I learned not only how to solve problems, but also how to think more deeply, develop mathematical intuition, and refine my mathematical taste. I am also deeply grateful to my master's advisor, Dr. Lijin Wang, for introducing me to academic research and setting me on the path that eventually led to my doctoral studies. My thanks extend to all of my collaborators for their discussions, support, and shared efforts. In particular, I would like to thank Dr. Minglei Yang for his continued help and generous guidance throughout the development and writing of my research papers.

The Department of Mathematics and Statistics at Auburn University has provided invaluable academic and financial support during my doctoral studies. I am truly thankful for the opportunities, resources, and training that Auburn University has offered me. I have also been fortunate to encounter many wonderful teachers and mentors during my time at Auburn. Special thanks are due to Dr. Junshan Lin for his continued care and support in both my academic and personal life. I am grateful to Dr. Hans Werner Van Wyk for his kindness, encouragement, and generosity. I would also like to thank Dr. Thi Thao

Phuong Hoang for serving on my dissertation committee and for her valuable guidance on my dissertation research.

My most special thanks go to Yi Liu. Looking back, it feels as though every decision I have made in life has come together to allow me to meet you at the right time and in the right place. You have brought color, joy, and meaning to my world. Your understanding, encouragement, and love have continually given me the strength to move forward, especially during the most challenging moments of this journey.

I am deeply thankful to my friends Menglei Wang, Cao Kha Doan, and Li Zhou for their friendship, support, and companionship. May each of you continue to shine brightly along your own paths and find happiness and fulfillment in the future.

Finally, my heartfelt gratitude goes to all of my family members for their constant love, encouragement, and support. In particular, I am forever indebted to my mother for her unconditional love and for everything she has done for me throughout my life. Without my family, I would not be where I am today.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Motivation and Overview

Generative modeling aims to construct efficient samplers for complex probability distributions. In applications such as image synthesis, the realism and quality of individual generated samples are often of primary interest. In scientific computing, however, it is often necessary to generate many samples in order to characterize uncertainty, estimate quantities of interest, or perform statistical inference. Such sampling problems arise in stochastic dynamical systems [51], where random perturbations affect the evolution of physical states, in uncertainty quantification [65], where uncertain inputs and model errors propagate through physical systems, and in Bayesian inverse problems [70], where posterior samples are used to quantify uncertainty in unknown parameters given observational data. Traditional approaches, such as repeated numerical simulation, Monte Carlo methods, and Markov chain Monte Carlo methods, may require substantial computational resources to achieve a desired level of accuracy. For example, the statistical error of standard Monte Carlo estimators typically decreases at the rate $O(N^{-1/2})$ with respect to the number of samples N [11], repeated numerical simulations must solve the underlying model for each new sample and may also introduce discretization errors, and Markov chain Monte Carlo methods may suffer from sample correlation and slow mixing [59]. These difficulties become more pronounced when the physical model is costly to evaluate, when samples are needed for many different input settings, or when the problem is high-dimensional. Generative surrogate models seek to shift much of the computational cost to an offline stage, after which new samples can be generated at a substantially lower online cost.

A common strategy in generative modeling is to transform samples from a simple reference distribution, such as a standard Gaussian distribution, into samples from a target distribution. Deep generative models, including generative adversarial networks

[30], variational autoencoders [44], normalizing flows [57], and diffusion models [33], have been developed for this purpose. Generative adversarial networks and variational autoencoders have achieved significant success in data generation and representation learning. This dissertation focuses on normalizing flows and diffusion models. Both methods are based on probability transport and are well suited for scientific computing problems involving stochastic dynamics, uncertainty propagation, and conditional sampling. They also provide a natural setting for studying model approximation, sampling accuracy, and numerical error. In this dissertation, they are considered as two complementary approaches for constructing generative surrogate models.

1.2 Normalizing Flows and Scientific Applications

Normalizing flows construct an invertible transformation between a target distribution and a tractable reference distribution, such as a standard Gaussian distribution. The change-of-variables formula makes it possible to evaluate the target probability density through the reference density and the Jacobian determinant of the transformation. Once the transformation has been learned, new samples can be generated by drawing samples from the reference distribution and applying the inverse map. Normalizing flows therefore provide sample generation and density evaluation within a unified framework [45, 52]. A variety of architectures have been developed to make the inverse transformation and the Jacobian determinant computationally tractable. Representative examples include coupling-based models such as Real NVP [19], autoregressive models such as masked autoregressive flows [53], and continuous normalizing flows such as FFJORD [31]. These methods generally achieve exact invertibility or efficient Jacobian computation through specially designed network structures. Although such designs make likelihood-based training computationally feasible, they may restrict the flexibility of the transport maps and the choice of neural network architectures.

Normalizing flows have increasingly been applied to scientific computing problems involving time-dependent probability distributions and stochastic dynamics. The probability density associated with a stochastic differential equation evolves according to the corresponding Fokker–Planck equation, making probability transport a natural approach for approximating its evolution. Temporal normalizing flows have been developed to solve time dependent Fokker–Planck equations [26], and related methods have been proposed to learn the temporal evolution of multivariate probability densities directly from data [50]. Physics informed normalizing flow models have also been used to solve forward and inverse stochastic differential equations [32]. These developments complement data driven approaches for learning dynamical systems from observations [40, 16]. However, many existing flow-based models learn the state distribution associated with a prescribed initial distribution. Because the resulting distribution changes when the initial distribution changes, the model generally needs to be retrained for each new initial distribution.

The first part of this dissertation is motivated by the desire to avoid imposing restrictive structural constraints on neural networks, since such constraints may reduce their expressive and approximation capabilities. To this end, the forward and inverse transport maps are represented by two independent fully connected neural networks, rather than by a single exactly reversible architecture. Their approximate inverse relationship is enforced through an additional loss term, leading to a pseudo-reversible normalizing flow [41, 71, 58]. This construction allows standard feed-forward neural networks to be used while retaining the likelihood-based formulation of normalizing flows. The initial state of the stochastic system is incorporated as a conditioning variable, enabling a single trained model to represent the evolution associated with various initial distributions without retraining [77]. A rigorous convergence analysis is provided to establish convergence of the learned probability density to the target density in the Kullback–Leibler divergence.

Numerical experiments involving Brownian motion, stochastic dynamics arising in magnetically confined fusion plasmas, and advection–diffusion transport demonstrate the accuracy and computational efficiency of the proposed method.

1.3 Generative Surrogate Modeling for Uncertainty Quantification

Uncertainty quantification concerns the identification, representation, and propagation of uncertainties arising from physical models, numerical simulations, and experimental data. When repeated evaluations of a physical model are computationally expensive, surrogate modeling provides an effective way to reduce the cost of uncertainty analysis. Classical surrogate methods include polynomial chaos expansions, Gaussian process regression, kernel methods, and deterministic neural networks [42, 75, 80]. These approaches are effective for approximating the deterministic relationship between model inputs and outputs. However, a deterministic surrogate does not directly represent the uncertainty introduced by random model errors or observational noise. To quantify forward uncertainty propagation, one must still know the noise distribution and be able to generate samples from it. For inverse uncertainty propagation, an additional sampling procedure, such as Markov chain Monte Carlo, is generally required to characterize the distribution of the model inputs conditioned on observed outputs. These limitations motivate surrogate models that directly represent conditional probability distributions rather than only deterministic model responses.

Deep learning has introduced several approaches for representing uncertainty in complex models and data of high dimension [1, 56]. Variational inference and Bayesian neural networks provide probabilistic frameworks for approximating posterior distributions and model uncertainty [5, 37, 74]. Methods such as Monte Carlo dropout and stochastic variational inference can reduce some of the associated computational costs [27, 34]. Amortized variational inference further seeks to learn conditional approximations that can be evaluated for new observations without solving a separate inference problem each

time [39]. Generative models provide another approach by learning distributions directly from data and have been applied to uncertainty estimation [7, 61]. In particular, normalizing flows can represent complex conditional and posterior distributions. Normalizing field flows have been developed for forward and inverse stochastic differential equations [32], while conditional normalizing flows have also been used to quantify uncertainty in applications such as medical image segmentation [60]. Nevertheless, these approaches commonly rely on exactly invertible transformations, which may limit the flexibility and scalability of the resulting models.

The second part of this dissertation develops a conditional pseudo-reversible normalizing flow as a generative surrogate model for forward and inverse uncertainty propagation [78]. The model is trained directly from paired samples of physical inputs and outputs and does not require prior knowledge of either the deterministic model or the distribution of the additive noise. The same class of neural network architectures and the same dataset can be used to learn conditional distributions in both the forward and inverse directions. Once trained, the model can efficiently generate samples for new conditioning values covered by the training data, thereby avoiding repeated evaluations of the physical model and auxiliary sampling procedures. The theoretical analysis establishes the convergence of the training loss, quantifies the error introduced by its Monte Carlo approximation, and proves convergence of the learned conditional density in the Kullback–Leibler divergence. Numerical experiments consider several forms of additive noise, problems with inputs and outputs of increasing dimension, and a geologic carbon storage application involving the prediction of three dimensional pressure fields from sparse well observations. The method is intended for conditioning values whose regions of high probability are adequately represented in the training data and is not designed for unrestricted extrapolation beyond the training domain.

1.4 Diffusion Models for High Dimensional Sampling

Diffusion models provide a continuous-time approach to probability transport. The basic idea is to define a forward process that gradually adds noise to samples from the target distribution until the resulting distribution becomes close to a tractable reference distribution. Samples are then generated by reversing this process. This idea was first developed through a connection with nonequilibrium thermodynamics [66] and was subsequently advanced through denoising diffusion probabilistic models [33], denoising diffusion implicit models [67], and score-based generative modeling [68]. In the continuous time formulation, the reverse dynamics can be described by a reverse-time stochastic differential equation or an associated probability flow ordinary differential equation. Both formulations depend on the score function of the evolving probability density, which describes the gradient of its log density. In standard diffusion models, this score function is typically approximated by a neural network through score matching or related training procedures [36, 68].

Although diffusion models have demonstrated strong performance in generating samples from complex and high-dimensional distributions, their practical implementation presents several challenges. Learning an accurate score function can be difficult, especially near the low noise end of the forward process and in high dimensions. Errors in the learned score may accumulate and be amplified as the reverse dynamics are solved. Training-free diffusion methods provide an alternative by constructing the score directly from available samples, without training a separate score network. In particular, when the target distribution is represented by an empirical distribution or a Gaussian mixture model, the corresponding score function can be evaluated analytically. Such a diffusion process can be used to generate labeled pairs between a reference distribution and the target distribution. These labeled data enable a final generative map to be trained through supervised learning [48]. Once this map has been trained, new samples can be generated directly without repeatedly solving the reverse diffusion process. The supervised learning

formulation also removes the need to impose an exactly reversible architecture on the final generative model.

A growing body of research has studied the convergence and error propagation of score-based diffusion models. A common approach assumes that the learned score is accurate in an integrated ℓ_2 sense and then estimates the discrepancy between the generated and target distributions. Early results under this assumption may exhibit an unfavorable dependence on the dimension [6, 8]. Later studies impose regularity conditions on the score or the reverse drift, such as spatial Lipschitz continuity, to obtain improved stability and convergence estimates [13, 14, 15]. Other analyses control both the score approximation error and the error in its Jacobian, which permits a more refined description of the stability of the reverse dynamics [46]. Early stopping has also been used to limit the accumulation of score estimation errors near the terminal region of the reverse process [4, 35]. Another line of research exploits additional structure in the target distribution, particularly Gaussian and Gaussian mixture models, for which the score has a simpler form and sharper convergence estimates can be obtained [62, 28]. Some recent work further combines score training error and sampling error to provide a complete analysis of the diffusion generation process [73]. These studies provide substantial theoretical insight, but many available estimates depend on quantities that are difficult to evaluate numerically, or they do not fully describe the error behavior observed in computational experiments.

The third part of this dissertation develops error estimates for a class of training-free diffusion models used to construct labeled data for the supervised learning of generative samplers [72]. The total approximation error is separated into three components: the discrepancy between the target distribution and its Gaussian mixture representation, the numerical discretization error of the reverse probability flow ordinary differential equation, and the approximation error introduced when the final generative map is trained from the labeled data. Since the first and third components can be studied using established results in density approximation and supervised learning, the analysis focuses on the numerical

discretization of the reverse dynamics. The Gaussian mixture representation provides an analytically available score function, which avoids the propagation of neural network score approximation errors. It also shows that the apparent singularity in the coefficients of the reverse dynamics is removable after the exact score is incorporated. Consequently, the analysis recovers classical convergence rates for ordinary differential equation solvers. The resulting bounds have an explicit and favorable dependence on the dimension, with growth of order $O(d)$ in the ℓ_2 norm and $O(\log d)$ in the ℓ_∞ norm. Numerical experiments verify the predicted convergence behavior with respect to both the time-step size and the dimensionality, providing a direct connection between the theoretical estimates and the observed numerical errors.

Chapter 2 Generative Modeling Problems and Mathematical Preliminaries

2.1 Mathematical Preliminaries

This section introduces the basic concepts and notation from probability theory and stochastic analysis that will be used throughout this dissertation. Standard references for these topics include [51, 38]. Throughout this dissertation, $\|\cdot\|$ denotes the Euclidean norm for vectors. For a matrix $A \in \mathbb{R}^{d\times m}$, $\|A\|_{\mathsf{F}}$ denotes its Frobenius norm.

Definition 2.1. *Let Ω be a nonempty set. A collection $\mathcal{F}$ of subsets of Ω is called a σ-**algebra** if the following properties hold:*

1. *$\Omega \in \mathcal{F}$;*
2. *if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$;*
3. *if $\{A_n\}_{n=1}^{\infty} \subseteq \mathcal{F}$, then*

$$\bigcup_{n=1}^{\infty} A_n \in \mathcal{F}.$$

*The elements of $\mathcal{F}$ are called **measurable sets** or **events**.*

Definition 2.2. *Let $\mathcal{F}$ be a σ-algebra on Ω. A mapping*

$$\mathbb{P} : \mathcal{F} \to [0,1]$$

*is called a **probability measure** if*

$$\mathbb{P}(\Omega) = 1$$

and, for every sequence of pairwise disjoint sets $\{A_n\}_{n=1}^{\infty} \subseteq \mathcal{F}$,

$$\mathbb{P}\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mathbb{P}(A_n).$$

The triple

$$(\Omega, \mathcal{F}, \mathbb{P})$$

is called a ***probability space****.*

The set Ω represents the collection of possible outcomes, $\mathcal{F}$ specifies the events to which probabilities can be assigned, and $\mathbb{P}$ assigns probabilities to these events.

Let $\mathcal{B}(\mathbb{R}^d)$ denote the Borel σ-algebra on $\mathbb{R}^d$, namely, the smallest σ-algebra that contains all open subsets of $\mathbb{R}^d$.

Definition 2.3. *A mapping*

$$X : (\Omega, \mathcal{F}) \to (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$$

is called ***measurable*** *if*

$$X^{-1}(B) \in \mathcal{F}, \qquad \textit{for every } B \in \mathcal{B}(\mathbb{R}^d).$$

A measurable mapping $X : \Omega \to \mathbb{R}^d$ is called an $\mathbb{R}^d$-valued ***random variable****, or a* ***random vector****. When $d = 1$, X is simply called a random variable.*

The probability distribution, or law, of X is the probability measure μ_X on $\mathbb{R}^d$ defined by

$$\mu_X(B) = \mathbb{P}(X \in B), \qquad B \in \mathcal{B}(\mathbb{R}^d).$$

If μ_X is absolutely continuous with respect to the Lebesgue measure, then there exists a probability density function $p_X : \mathbb{R}^d \to [0, \infty)$ such that

$$\mathbb{P}(X \in B) = \int_B p_X(x)\, dx, \qquad B \in \mathcal{B}(\mathbb{R}^d).$$

In this case, we write

$$X \sim p_X.$$

For an integrable random variable X, its expectation is defined by

$$\mathbb{E}[X] = \int_{\Omega} X(\omega)\, d\mathbb{P}(\omega).$$

More generally, if $\varphi : \mathbb{R}^d \to \mathbb{R}$ is measurable and $\varphi(X)$ is integrable, then

$$\mathbb{E}[\varphi(X)] = \int_{\mathbb{R}^d} \varphi(x) p_X(x)\, dx.$$

The standard Gaussian distribution on $\mathbb{R}^d$ is denoted by

$$\mathcal{N}(0, I_d),$$

where I_d is the $d \times d$ identity matrix.

Let $X \in \mathbb{R}^{d_1}$ and $Y \in \mathbb{R}^{d_2}$ be random vectors with joint density $p_{X,Y}(x, y)$. If $p_X(x) > 0$, the conditional density of Y given $X = x$ is defined by

$$p_{Y|X}(y \mid x) = \frac{p_{X,Y}(x, y)}{p_X(x)}. \tag{2.1}$$

Conditional probability distributions will be used in later chapters to describe transition distributions of stochastic dynamical systems and uncertainty propagation in noisy physical models.

For two probability densities p and q on $\mathbb{R}^d$, the **Kullback–Leibler divergence** from p to q is defined by

$$D_{\mathsf{KL}}(p\|q) = \int_{\mathbb{R}^d} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx, \tag{2.2}$$

provided that p is absolutely continuous with respect to q. This quantity will be used to measure the discrepancy between target distributions and their generative approximations.

We next introduce stochastic processes and Brownian motion, which are needed to formulate stochastic differential equations and diffusion models.

Definition 2.4. *Let $T > 0$. A family of random variables*

$$\{X_t\}_{t\in[0,T]}$$

*defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is called a **stochastic process**. A family of sub-σ-algebras*

$$\mathbb{F} = \{\mathcal{F}_t\}_{t\in[0,T]}$$

*is called a **filtration** if*

$$\mathcal{F}_s \subseteq \mathcal{F}_t \subseteq \mathcal{F}, \qquad 0 \le s \le t \le T.$$

*A stochastic process $\{X_t\}_{t\in[0,T]}$ is called **adapted** to $\mathbb{F}$ if X_t is $\mathcal{F}_t$-measurable for every $t \in [0, T]$.*

The σ-algebra $\mathcal{F}_t$ represents the information available up to time t. Unless otherwise stated, all filtrations are assumed to satisfy the usual conditions of completeness and right continuity.

Definition 2.5. *An $\mathbb{R}^d$-valued stochastic process*

$$W_t = (W_t^{(1)}, \dots, W_t^{(d)})^\top, \qquad t \in [0, T],$$

*is called a d-dimensional **standard Brownian motion** with respect to the filtration $\mathbb{F}$ if the following conditions hold:*

1. *$W_0 = 0$ almost surely;*
2. *W_t has continuous sample paths almost surely;*
3. *for every $0 \le s < t \le T$, the increment $W_t - W_s$ is independent of $\mathcal{F}_s$;*

4. *for every* $0 \le s < t \le T$,

$$W_t - W_s \sim \mathcal{N}(0, (t-s)I_d).$$

Let W_t be an m-dimensional Brownian motion. A d-dimensional **stochastic differential equation** takes the form

$$dX_t = b(t, X_t)\,dt + \sigma(t, X_t)\,dW_t, \qquad X_0 = \xi, \tag{2.3}$$

where

$$b : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$$

is the drift coefficient,

$$\sigma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times m}$$

is the diffusion coefficient, and ξ is an $\mathcal{F}_0$-measurable initial random variable. Equation (2.3) is understood in the integral form

$$X_t = \xi + \int_0^t b(s, X_s)\,ds + \int_0^t \sigma(s, X_s)\,dW_s, \qquad t \in [0,T], \tag{2.4}$$

where the second integral is an Itô stochastic integral.

An adapted process $\{X_t\}_{t\in[0,T]}$ with continuous sample paths is called a **strong solution** of (2.3) if it satisfies (2.4) almost surely for every $t \in [0,T]$. A standard sufficient condition for the existence and uniqueness of a strong solution is that the drift and diffusion coefficients satisfy global Lipschitz and linear growth conditions. More precisely, suppose that there exist constants $L, C > 0$ such that

$$\|b(t,x) - b(t,y)\| + \|\sigma(t,x) - \sigma(t,y)\|_{\mathsf{F}} \le L\|x-y\|, \tag{2.5}$$

and

$$\|b(t,x)\| + \|\sigma(t,x)\|_{\mathsf{F}} \le C(1+\|x\|), \tag{2.6}$$

for all $t \in [0, T]$ and $x, y \in \mathbb{R}^d$. If, in addition,

$$\mathbb{E}\big[\|\xi\|^2\big] < \infty,$$

then (2.3) admits a unique strong solution $\{X_t\}_{t\in[0,T]}$ satisfying

$$\mathbb{E}\left[\sup_{0\le t\le T} \|X_t\|^2\right] < \infty.$$

When X_t admits a sufficiently regular probability density function $p(t, x)$, its evolution can be described by the **Fokker–Planck equation**. Let

$$a(t, x) = \sigma(t, x)\sigma(t, x)^\top.$$

Then $p(t, x)$ satisfies

$$\frac{\partial p}{\partial t}(t, x) = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i} \left(b_i(t, x)p(t, x)\right) + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \left(a_{ij}(t, x)p(t, x)\right). \tag{2.7}$$

The SDE formulation and the corresponding Fokker–Planck equation provide two related descriptions of stochastic dynamics. The SDE describes the evolution of individual sample paths, while the Fokker–Planck equation describes the evolution of their probability distribution.

2.2 Generative Modeling Problems

Let $Z \in \mathbb{R}^m$ be a random variable with a simple reference distribution p_Z, and let $X \in \mathbb{R}^d$ be a random variable with target distribution p_X. The generative modeling problem is to construct a transformation

$$G : \mathbb{R}^m \to \mathbb{R}^d$$

such that the distribution of $G(Z)$ approximates the target distribution p_X. In this setting, G is referred to as a generator.

In many applications, the reference distribution is chosen to be a standard Gaussian distribution because independent Gaussian samples can be generated efficiently. Once the generator G has been constructed, a new sample from the learned generative model is obtained by first drawing $z \sim \mathcal{N}(0, I_m)$ and then evaluating $G(z)$. Thus, the potentially expensive task of constructing the generator is performed during an offline stage, while sample generation can be carried out at a reduced online computational cost.

The target distribution p_X may be high-dimensional, non-Gaussian, or multimodal. A useful generative model should be able to represent such distributions while generating samples efficiently after training. In scientific computing, it is also important to understand the accuracy of the learned model. The discrepancy between the generated distribution and the target distribution may arise from limited training data, approximation errors in the model class, optimization errors during training, or numerical errors in the construction of the generative model. Theoretical analysis and numerical verification of these errors are therefore important for the reliable use of generative models in scientific applications.

2.3 Normalizing Flows

Let $X \in \mathbb{R}^d$ be a random vector with target density p_X, and let $Z \in \mathbb{R}^d$ be a random vector with a simple reference density p_Z of the same dimension as X. A normalizing flow seeks an invertible and differentiable function

$$f : \mathbb{R}^d \to \mathbb{R}^d$$

such that

$$Z = f(X). \tag{2.8}$$

The inverse mapping

$$X = f^{-1}(Z) \tag{2.9}$$

transforms samples from the reference distribution into samples from the target distribution.

Let $J_f(x)$ denote the Jacobian matrix of f evaluated at x. By the change-of-variables formula, the densities of X and Z satisfy

$$p_X(x) = p_Z\big(f(x)\big)\,\left|\det J_f(x)\right|. \tag{2.10}$$

Equivalently, for $z = f(x)$,

$$p_Z(z) = p_X\big(f^{-1}(z)\big)\,\left|\det J_{f^{-1}}(z)\right|. \tag{2.11}$$

Equation (2.10) also makes it possible to evaluate the target density through the reference density and the Jacobian determinant. Given samples $\{x_i\}_{i=1}^N$ from the target distribution, the parameters of f can be learned by minimizing the negative log-likelihood

$$\begin{aligned} \mathcal{L}_{\mathsf{NLL}} &= -\frac{1}{N}\sum_{n=1}^{N} \log p_X(x_n) \\ &= -\frac{1}{N}\sum_{n=1}^{N} \left[\log p_Z\big(f(x_n)\big) + \log\left|\det J_f(x_n)\right|\right]. \end{aligned} \tag{2.12}$$

After training, new samples can be generated by first drawing $z \sim p_Z$ and then applying the inverse map $x = f^{-1}(z)$. In conventional normalizing flows, f is usually constructed using specially designed neural network architectures so that f^{-1} and $\det J_f$ can be computed efficiently. The normalizing flow framework introduced here will be extended in Chapters 3 and 4 through pseudo-reversible neural network architectures.

2.4 Diffusion Models

Diffusion models construct generative models through a continuous-time stochastic process. Let $\{Z_t\}_{t\in[0,T]}$ be a diffusion process satisfying

$$dZ_t = b(t)Z_t\,dt + \sigma(t)\,dW_t, \qquad Z_0 \sim p_0, \tag{2.13}$$

where p_0 denotes the target density and W_t is an m-dimensional Brownian motion. The coefficients b and σ are selected such that the distribution p_t of Z_t gradually evolves from the target distribution p_0 to a simple reference distribution p_T, which is commonly chosen to be a standard Gaussian distribution. Notice that SDE in Eq. (2.13) can only be solved forward in time. To derive the transformation from p_T to p_0 we first introduce score function.

Definition 2.6. *Suppose that $p_t(z) > 0$ and is differentiable with respect to z. The **score function** of p_t is defined by*

$$s_t(z) = \nabla_z \log p_t(z). \tag{2.14}$$

The reverse-time SDE [3] is defined by

$$dZ_t = \left[b(t)Z_t - \sigma^2(t)s_t(Z_t)\right]dt + \sigma(t)\,d\overleftarrow{W}_t, \qquad t : T \to 0, \qquad Z_T \sim p_T, \tag{2.15}$$

where $\overleftarrow{W}_t$ denotes a reverse-time Brownian motion. The reverse SDE in Eq. (2.15) has the same marginal densities as the forward SDE in Eq. (2.13). Consequently, solving Eq. (2.15) from $t = T$ to $t = 0$ generates a random variable Z_0 with density p_0.

In practice, the score function s_t is unknown and is commonly approximated by a neural network. Chapter 5 considers a training-free construction of the score function and studies the numerical error associated with solving the diffusion model.

Chapter 3 A Pseudo-Reversible Normalizing Flow for Stochastic Dynamical Systems with Various Initial Distributions

3.1 Problem Setting

We consider the d-dimensional stochastic process X_t, corresponding to the solution of the stochastic differential equation

$$X_t = X_0 + \int_0^t b(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s \quad \text{with } X_0 \in \mathcal{D},\ t \in [0, T], \tag{3.1}$$

where $T > 0$, $\mathcal{D}$ is a bounded domain in $\mathbb{R}^d$, $b : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : [0, T] \times \mathbb{R}^d \to \mathbb{R}^{d\times m}$ are drift and diffusion coefficients, respectively, and $W_s := (W_s^1, \ldots, W_s^m)^\top$ is a m-dimensional standard Brownian motion. We assume that b and σ satisfy, for some constant C,

$$|b(t, x)| + |\sigma(t, x)| \le C(1 + |x|), \text{ for } x \in \mathbb{R}^d, t \in [0, T], \tag{3.2}$$

and

$$|b(t, x) - b(t, y)| + |\sigma(t, x) - \sigma(t, y)| \le C|x - y|, \text{ for } x, y \in \mathbb{R}^d, t \in [0, T]. \tag{3.3}$$

For an initial state $X_0 \in \mathcal{D}$, the SDE in Eq. (3.1) has a unique time continuous solution X_t [51]. As the state X_t in Eq. (3.1) depends on the initial condition X_0, we will write X_t in the conditional form $X_t|X_0$ when we need to emphasize this dependence.

In applications one is typically interested in evaluating quantities of interest, QoI, of the form

$$\text{QoI} = \int_{\mathbb{R}^d} F(x_t)\, p_{X_t}(x_t)dx_t\,, \tag{3.4}$$

where F denotes a physical property of the system, and p_{X_t} is the probability density of the final state. Because of the dependence of the final state on the initial condition, normalizing flow strategies focusing on the distribution $p_{X_t}(x_t)$, e.g., Refs. [9, 50, 26], might

not be efficient since every time the initial condition changes, the evaluation of QoI requires re-training of the neural network for $p_{X_t}(x_t)$ corresponding to the new initial condition.

To circumvent this problem we propose here a new method that focuses on the use of normalizing flows to create a surrogate model for the probability density $p_{X_t|X_0}(x_t|x_0)$, which gives the transition probability of being at x_t starting at x_0. Once a normalizing flow model is trained for $p_{X_t|X_0}(x_t|x_0)$, the computation of the QoI reduces to the direct evaluation of the integral

$$\text{QoI} = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} F(x_t)\, p_{X_t|X_0}(x_t|x_0) p_{X_0}(x_0) dx_t dx_0\,, \tag{3.5}$$

where $p_{X_0}(x_0)$ denotes the probability density of the initial condition. In this approach the computational cost of evaluating QoI for different initial conditions reduces to the cost of performing a quadrature without a retraining overhead. If $\{x_0^{(m)}\}$ for $m = 1 \ldots M$ is a set of M-samples drawn from $p_{X_0}(x_0)$, and $\{x_t^{(n)}\}$ is the corresponding set of N-samples drawn from $p_{X_t|X_0}(x_t|x_0)$, then the quadrature can be approximated as

$$\text{QoI} \approx \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} F\left(x_t^{(n)}|x_0^{(m)}\right), \tag{3.6}$$

3.2 The Pseudo-Reversible Normalizing Flow (PR-NF)

A normalizing flow model can be viewed as a nonlinear transformation that maps an unknown target random variable, denoted by X, to a standard random variable, e.g., the standard Gaussian, denoted by Z. Because we intend to take into account the initial condition X_0 in our normalizing flow model, we define the target random variable X as

$$X := (X_0, X_t|X_0) \in \mathbb{R}^{2d}. \tag{3.7}$$

A normalizing flow model includes a forward transformation mapping X to Z and an inverse transformation mapping Z to X, i.e.,

$$Z = h(X; \theta_h) \quad \text{and} \quad \widehat{X} = g(Z; \theta_g), \tag{3.8}$$

where θ_h, θ_g are parameters of the transformations and $\widehat{X} = X$ if $g = h^{-1}$. The function h moves (or flows) in the normalizing direction, i.e., from an unknown distribution to a standard distribution p_Z. The tradition of defining Z as the standard normal random variable $\mathcal{N}(0, \mathbf{I}_{2d})$ gives rise to the name "normalizing flow". Let $p_X(x)$ and $p_Z(z)$ denote the PDF of X and Z, respectively. The relation between the two PDFs can be obtained by the change of variables formula, i.e.,

$$p_X(x) = p_Z(z) \left| \frac{\partial z}{\partial x} \right| = p_Z(h(x)) |\text{det} J_h(x)|, \tag{3.9}$$

where det$J_h(x)$ is the determinant of the Jacobian matrix J_h of the forward map $h(\cdot)$.

3.2.1 The PR-NF Model's Architecture

The PR-NF model uses the pseudo-reversible neural network architecture [41, 71, 58], i.e., $g \approx h^{-1}$ in Eq. (3.8), to replace the exact reversible architectures used in most existing normalizing flow models, e.g., MAF [53], Real NVPs [19]. The typical exact reversible architectures rely on functions with either triangular Jacobians or Lipschitz continuity, where the Jacobian determinant can be efficiently approximated. However, the relaxation of the reversibility condition increases the flexibility of the model design. Specifically, we define $h(\cdot; \theta_h)$ and $g(\cdot; \theta_g)$ as two independent fully-connected neural networks, and the pseudo-reversibility is imposed by adding a soft constraint $\|\widehat{X} - X\|^2$ to the loss function. This architecture is different from autoencoders because the size of the bottleneck, i.e., the dimension of Z, is the same as the dimension of X and $\widehat{X}$. Figure 3.1

illustrates the performance of the PR-NF on a standard two-dimensional dataset. We observe that the left panel (i.e., the true distribution) and the right panel (i.e., the PR-NF generated distribution) are in good agreement, meaning the relaxation of the reversibility does not cause a significant error.

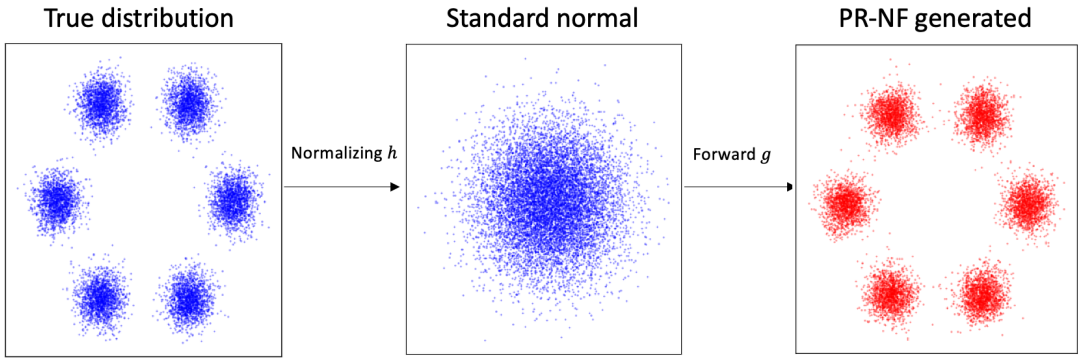

Figure 3.1: Illustration of the PR-NF's performance.

The novelty of the PR-NF model is to learn the conditional distribution in Eq. (3.5), such that we can use the trained PR-NF model to compute the QoI in Eq. (3.5) for arbitrary initial conditions, $p_{X_0}(x_0)$, without re-training. The key ingredient is to include the initial state X_0 into the input of the PR-NF model, i.e., Eq. (3.7). The network architecture is shown in Figure 3.2. Specifically, the mapping $h(\cdot)$ consists of two components, i.e., $h(x_0, x_t|x_0) = (h_0(x_0), h_1(x_0, x_t|x_0; \theta_h))$, where $h_1(x_0, x_t|x_0; \theta_h) : \mathbb{R}^{2d} \to \mathbb{R}^d$ is a fully connected neural network with tuning parameter θ_h, and $h_0(x_0) : \mathbb{R}^d \to \mathbb{R}^d$ is an identity mapping depending only on x_0. We emphasize that it is critical to include x_0 as one input of h_1 to characterize the conditional distribution $p_{X_t|X_0}(x_t|x_0)$. We denote by z_0 the output of $h_0(x_0)$ and z_t the output of $h_1(x_0, x_t|x_0; \theta_h)$. The inverse map g has a similar structure as the forward map, i.e., $g(z_0, z_t) = (g_0(z_0), g_1(z_0, z_t; \theta_g))$, where g_0 is an identity map and g_1 is a fully connected neural network with tuning parameter θ_g. We denote by $\widehat{x}_0$ and $\widehat{x}_t$ the outputs of g_0 and g_1, respectively. Note that $\widehat{x}_0 = z_0 = x_0$ because h_0 and g_0 are identity maps.

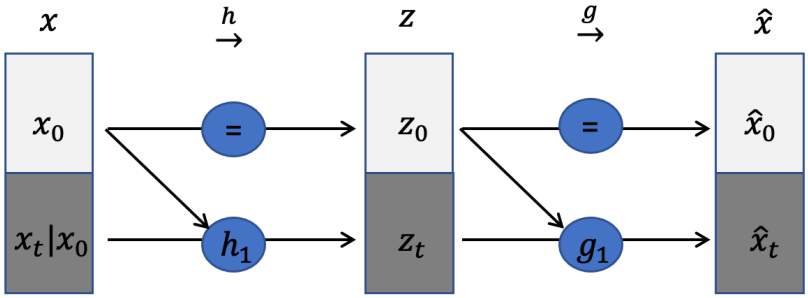

Figure 3.2: The network architecture of the proposed PR-NF model.

3.2.2 The Loss Function

The loss function is defined based on a training data set, i.e.,

$$\mathcal{V}_{\text{train}} = \left\{x^{(n)}\right\}_{n=1}^{N} = \left\{x_0^{(n)}, x_t^{(n)}|x_0^{(n)}\right\}_{n=1}^{N}, \tag{3.10}$$

where $\{x_0^{(n)}\}_{n=1}^{N}$ is generated from the uniform distribution over the bounded domain $\mathcal{D}$ in Eq. (3.1). For each initial state $x_0^{(n)}$, we numerically solve the SDE in Eq. (3.1) to obtain the sample $x_t^{(n)}|x_0^{(n)}$. The loss function consists of two components, i.e.,

$$\mathcal{L} = \mathcal{L}_1 + \lambda\mathcal{L}_2, \tag{3.11}$$

where $\mathcal{L}_1$ is the negative log-likelihood loss defined by

$$\mathcal{L}_1 = -\frac{1}{N}\sum_{n=1}^{N}\left(\log p_{Z_t}(h_1(x^{(n)};\theta_h)) + \log\left|\det J_h(x^{(n)};\theta_h)\right|\right), \tag{3.12}$$

with $p_{Z_t}(\cdot)$ the probability density function of the standard normal distribution, and $\mathcal{L}_2$ is the pseudo-reversibility loss that measures the difference between x and $\widehat{x}$, i.e.,

$$\mathcal{L}_2 = \frac{1}{N}\sum_{n=1}^{N}\left(\left\|x^{(n)} - g(h(x^{(n)};\theta_h);\theta_g)\right\|_2^2 + \left|\det J_g(h(x^{(n)}))\det J_h(x^{(n)}) - 1\right|\right), \tag{3.13}$$

and λ is a hyperparameter that will be discussed in Section 3.2.3.

The Jacobian determinant in $\mathcal{L}_1$ can be written as

$$|\det J_h(x)| = \left| \det \begin{pmatrix} \mathbf{I}_d, & 0 \\ \frac{\partial z_t}{\partial x_0}, & \frac{\partial z_t}{\partial x_t} \end{pmatrix} \right| = \left| \det \left(\frac{\partial z_t}{\partial x_t} \right) \right|, \tag{3.14}$$

where $\mathbf{I}_d$ is a $d \times d$ identity matrix. The simplification in Eq. (3.14) is based on the fact h_0 is an identity map with respect to x_0. Thus, although we doubled the size of the input of the PR-NF model, the size of the Jacobian matrix is still $d \times d$. The proposed PR-NF has $\mathcal{O}(d^3)$ complexity in Jacobian determinant computation. However, we observe that it is not a bottleneck with GPU-accelerated linear algebra libraries in PyTorch and TensorFlow, especially for a wide range of physical processes, i.e., the examples in Section 3.4, in the phase space with dimension up to six.

3.2.3 Hyperparameter Tuning

The hyperparameter λ in Eq. (3.11) determines the importance of the reversibility loss $\mathcal{L}_2$. Thus, a hyperparameter tuning is needed to achieve a good performance. In this work, we will perform grid search and use the cross-entropy as a metric to find the best value for λ. Specifically, we will generate a set of candidates, denoted by $\{\lambda_j\}_{j=1}^J$. For each λ_j, we train one PR-NF model, denoted by $(h^{(j)}(x), g^{(j)}(z))$. Then, we generate M samples of $\widehat{x}$ by running the inverse mapping, i.e.,

$$\left\{ \widehat{x}^{(j,m)} = g^{(j)}(z^{(m)}), m = 1, \ldots, M \right\}, \tag{3.15}$$

where $\{z^{(m)}\}_{m=1}^M = \{z_0^{(m)}, z_t^{(m)}\}_{m=1}^M$ with $z_0^{(m)}$ generated uniformly from $\mathcal{D}$ and $z_t^{(m)}$ generated from the standard normal distribution. Because it does not require solving the SDE in Eq. (3.1), we can generate a large number of samples. Next, we build a kernel density

estimator using the samples in Eq. (3.15), and compute the cross-entropy by

$$H(\lambda_j) = -\frac{1}{N}\sum_{n=1}^{N} \log(p_{\text{KDE}}(x_t^{(n)})), \tag{3.16}$$

where p_{KDE} is the kernel density estimator based on the samples in Eq. (3.15) and $\{x_t^{(n)}\}_{n=1}^N$ is the training data set in Eq. (3.10). The optimal hyperparameter λ is obtained by minimizing the cross-entropy, i.e., $\lambda = \text{argmin}\, H(\lambda_j)$.

3.2.4 Discussion on the Computational Cost

The algorithmic cost of sampling methods consists of two parts, the "offline cost" and the "online cost". The offline cost of an algorithm represents the running time for the training process. It is determined by the architecture of the neural network, the optimizer (numerical scheme), the size of the training dataset and the number of training epochs ("time" steps). An epoch in machine learning means one complete pass of the training dataset through the algorithm. We choose the number of epochs such that the loss function stably attains the minimum. The Monte Carlo method for sampling has no training process. However, it needs to be run repeatedly for each new initial condition. The PR-NF model is a generative model where the offline cost accounts for most of the total algorithmic cost. However, the offline cost is a one-time cost (no additional training is needed for future simulations). We denote the offline cost of the PR-NF method as C_{offline}. It is easy to utilize the GPU architecture to accelerate the computation for the training process under the PyTorch machine learning framework.

The online cost is the computational cost for sampling. Although the Monte Carlo method is conceptually simple and algorithmically straightforward, the associated computational cost can be staggeringly high. In general, the method requires many samples to get a good approximation, which may incur an arbitrarily large total runtime if the processing time of a single sample is high [64]. In contrast, the PR-NF model is a transformation

that maps samples from the standard normal distribution to the distribution of interest. For an initial distribution, the computational cost of the MC method is significantly greater than the online cost of the PR-NF method because it involves the sampling and numerical integration of the SDE in Eq. (3.1). The computational cost depends on how many samples need to generate. We denote C_{MC} and $C_{\text{PR-NF}}$ as the computational cost of an initial distribution for MC and PR-NF methods, respectively. We have $C_{\text{MC}} \gg C_{\text{PR-NF}}$. To test N_{init} different initial distributions, the cost of the PR-NF method $\mathcal{O}(C_{\text{offline}} + C_{\text{PR-NF}} \times N_{\text{init}})$ is cheaper than the MC method $\mathcal{O}(C_{\text{MC}} \times N_{\text{init}})$ when N_{init} is large.

3.3 Convergence Analysis of the PR-NF Model

This section provides the convergence analysis of the pseudo-reversible normalizing flow (PR-NF) model in Section 3.2. For notational simplicity, the target random variable and its probability density function are denoted by X and $p_X(x)$, respectively. Here, we limit our attention to single-hidden-layer neural networks. The work in [52] shows that for any pair of well-behaved[1] distributions $p_X(x)$ and $p_Z(z)$, there exists a diffeomorphism $z = f(x) = (f_1(x), \ldots, f_d(x))$ that can transform $p_X(x)$ to $p_Z(z)$, i.e.,

$$p_X(x) = p_Z(f(x))|\det J_f(x)|. \tag{3.17}$$

The proposed PR-NF model can be viewed as an approximation of the diffeomorphism, where $h(x)$ and $g(z)$ in Eq. (3.8) approximate f and f^{-1}, respectively.

We study the convergence of the PR-NF model in three stages. Section 3.3.1 shows the existence of a convergent neural network approximations to the target diffeomorphism f by exploiting the universal approximation theorem of fully connected neural networks. Section 3.3.2 shows that the convergence of the loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ in Eq. (3.11) is a necessary condition to obtain the convergent approximation discussed in

[1] A well-behaved distribution $p_X(x)$ means that $p_X(x) > 0$ for all $x \in \mathbb{R}^d$, and all conditional probabilities $\Pr(X_i \leq x_i | x_{<i})$ are differentiable, for $i = 1, 2, \ldots, d$.

Section 3.3.1, which verifies that appropriateness of the loss function used to train the PR-NF model. Section 3.3.3 shows that the convergence of $\mathcal{L}_1$ and $\mathcal{L}_2$ plus some mild assumptions on the h and g is sufficient to ensure the convergence of the KL divergence between the true and the approximate probability density function.

3.3.1 Existence of a Convergent Approximation to the Diffeomorphism

Assumption 3.1. *We assume that the determinant of the Jacobian matrix and any partial derivative of $f \in C^1(\mathbb{R}^d)$ are bounded, i.e.,*

$$\left|\frac{\partial f_i(x)}{\partial x_j}\right| < A_1 < \infty, \quad |\det J_f(x)| > A_2 > 0, \tag{3.18}$$

where A_1 and A_2 are positive constants and $i,\ j \in \{1, \ldots, d\}$. Additionally, the probability density function $p_X(x)$ is bounded and all the conditional probabilities are differentiable. There exists a positive constant K such that

$$p_X(x) < A_3 \exp(-\alpha \|x\|^2) \ \text{ for } \ \|x\| > K, \tag{3.19}$$

where $\alpha > 0$ and $A_3 > 0$ are positive constants and $\|x\|$ denotes the l^2 norm.

Under Assumption 3.1, Lemma 3.1 and Lemma 3.2 demonstrate the existence and reversibility of convergent flows h and g, respectively. To proceed, we introduce some multivariate notations: $\mathbb{Z}_+^d$ denote the lattice of nonnegative multi-integers in $\mathbb{R}^d$. For $m = (m_1, \ldots, m_d) \in \mathbb{Z}_+^d$, we set $|m| = m_1 + \ldots + m_d$, $x^m = x_1^{m_1} \cdots x_d^{m_d}$, and $D^m = \frac{\partial^{|m|}}{\partial x_1^{m_1} \cdots \partial x_d^{m_d}}$. We also define

$$C^k(\Omega) := \big\{ f : D^m f_i \in C(\Omega) \text{ for all } i \in \{1, \ldots, d\}, \text{ and } |m| \leq k \big\}.$$

Lemma 3.1 (Theorem 4.1 in [55])**.** *For any given $\varepsilon > 0$, there exists a single-hidden-layer neural network $h = (h_1, \ldots, h_d)$ with sufficient number of neurons to approximate a function*

$f \in C^k(\mathbb{R}^d)$ *in a compact set* $\Omega \subset \mathbb{R}^d$ *such that*

$$\max_{x\in\Omega} |D^m f_i(x) - D^m h_i(x)| < \varepsilon, \tag{3.20}$$

for all $i \in \{1, \dots, d\}$ *and* $m \in \mathbb{Z}_+^d$ *with* $|m| \le k$.

Lemma 3.2. *Under Assumption 3.1 and Lemma 3.1, the flow* h *approximating* $f \in C^k(\mathbb{R}^d)$ *is invertible for sufficiently small* $\varepsilon > 0$.

Proof. Let $\mathcal{S}_d$ be the set of all permutation of $\{1, \dots, d\}$, we have

$$|\det J_f(x) - \det J_h(x)| = \left| \sum_{\sigma\in\mathcal{S}_d} \left(\operatorname{sgn}(\sigma) \left(\prod_{i=1}^d \frac{\partial f_i}{\partial x_{\sigma_i}} - \prod_{i=1}^d \frac{\partial h_i}{\partial x_{\sigma_i}} \right) \right) \right| < C\varepsilon \tag{3.21}$$

for any $x \in \Omega$, where the last inequality holds by Lemma 3.1 with $k = 1$. According to Eq. (3.18), we have

$$A_2 < |\det J_f(x)| = \left| \sum_{\sigma\in\mathcal{S}_d} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^d \frac{\partial f_i}{\partial x_{\sigma_i}} \right) \right| < CA_1^d. \tag{3.22}$$

We choose $\varepsilon < \frac{A_2}{2C}$ such that

$$|\det J_h(x)| \le |\det J_f(x)| + |\det J_f(x) - \det J_h(x)| < \frac{A_2}{2} + CA_1^d, \tag{3.23}$$

$$|\det J_h(x)| \ge |\det J_f(x)| - |\det J_f(x) - \det J_h(x)| > \frac{A_2}{2} > 0. \tag{3.24}$$

By the inverse function theorem, h is invertible in Ω. □

Similarly, we can prove the invertibility of g for the function f^{-1} with Eq. (3.20). So far, we proved the existence of convergent neural network h and g to approximate f and f^{-1}, respectively.

3.3.2 Convergence of the Loss Function

We now prove that the convergence of the loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ in Eq. (3.11) is a necessary condition to obtain the convergent approximation discussed in Section 3.3.1. To proceed, we define two auxiliary random variables

$$\widetilde{X} = h^{-1}(Z) \text{ and } \widehat{X} = g(Z), \tag{3.25}$$

where Z follows the standard normal distribution. Using the change of variables formula, the probability density functions of $\widetilde{X}$ and $\widehat{X}$ are defined by

$$p_{\widetilde{X}}(x) = p_Z(h(x))|\det J_h(x)|, \quad p_{\widehat{X}}(x) = p_Z(g^{-1}(x))|\det J_{g^{-1}}(x)|. \tag{3.26}$$

For simplicity, we consider the loss functions in the continuous form, i.e.,

$$\mathcal{L}_1 = -\int_{\mathbb{R}^d} p_X(x) \log p_{\widetilde{X}}(x) dx, \tag{3.27}$$

and

$$\mathcal{L}_2 = \int_{\Omega} p_X(x) \left(\|g(h(x)) - x\|^2 + |\det J_g(h(x)) \det J_h(x) - 1| \right) dx, \tag{3.28}$$

where Ω is a compact set. For the variables $\widetilde{X}$ and $\widehat{X}$, we have the following assumptions.

Assumption 3.2. *We assume the density functions $p_X(x)$ and $p_{\widetilde{X}}(x)$ satisfy*

$$A_4 \exp(-\alpha\|x\|^2) < \frac{p_X(x)}{p_{\widetilde{X}}(x)} < A_4 \exp(\alpha\|x\|^2), \tag{3.29}$$

where α and A_4 are positive constants.

Lemma 3.3 and Theorems 3.1, 3.2 below will demonstrate that both $\mathcal{L}_1$ and $\mathcal{L}_2$ will approach the optimal constants in this case, with a difference between the loss and the optimal constant of the order $O(\varepsilon)$.

Lemma 3.3. *For arbitrarily small $\varepsilon > 0$, there exists a random variable $\widetilde{X}$ with the density function $p_{\widetilde{X}}(x)$ defined in Eq.* (3.26) *such that*

$$|p_X(x) - p_{\widetilde{X}}(x)| < C\varepsilon, \quad \textit{for} \quad x \in \Omega, \tag{3.30}$$

where $\Omega \subseteq \mathbb{R}^d$ is a compact set and $C > 0$ is a constant.

Proof. By Lemma 3.1 with $k = 0$, for any $x \in \Omega$ we have

$$\|f(x) - h(x)\| = \sqrt{\sum_{i=1}^{d} |f_i(x) - h_i(x)|^2} < C\varepsilon. \tag{3.31}$$

The standard normal random variable $Z \in \mathbb{R}^d$ has bounded gradient, i.e., $\|\nabla p_Z(z)\| < C$ for any $z \in \mathbb{R}^d$. Applying the mean value theorem, we have

$$|p_Z(f(x)) - p_Z(h(x))| = |\nabla p_Z(\xi) \cdot (f(x) - h(x))| \leq \|\nabla p_Z(\xi)\| \|f(x) - h(x)\| < C\varepsilon, \tag{3.32}$$

where ξ is between $f(x)$ and $h(x)$, ε is given in Lemma 3.1 and $C > 0$ is a constant. Because Z has bounded density function, we can exploit Eqs. (3.21), (3.23), (3.32) to derive

$$\begin{aligned}
&\left|p_X(x) - p_{\widetilde{X}}(x)\right| \\
&= |p_Z(f(x))|\det J_f(x)| - p_Z(h(x))|\det J_h(x)|| \\
&= |p_Z(f(x))(|\det J_f(x)| - |\det J_h(x)|) + |\det J_h(x)|(p_Z(f(x)) - p_Z(h(x)))| \\
&\leq |p_Z(f(x))|\,|\det J_f(x) - \det J_h(x)| + |\det J_h(x)|\,|p_Z(f(x)) - p_Z(h(x))| < C\varepsilon
\end{aligned} \tag{3.33}$$

for any $x \in \Omega$ which concludes the proof. □

Based on the results from Lemma 3.3, we have the convergence of the continuous loss function $\mathcal{L}_1$ defined in Eq. (3.27).

Theorem 3.1.

For arbitrarily small $\varepsilon > 0$, there exists a random variable $\widetilde{X}$ with the density function $p_{\widetilde{X}}(x)$ in Eq. (3.26) *such that $\mathcal{L}_1$ has the bound*

$$-\int_{\mathbb{R}^d} p_X(x) \log p_X(x) dx \leq \mathcal{L}_1 < -\int_{\mathbb{R}^d} p_X(x) \log p_X(x) dx + \varepsilon. \tag{3.34}$$

Proof. Using the inequality $\log t \leq t - 1$ for all $t > 0$, we have

$$\begin{aligned} -\int_{\mathbb{R}^d} p_X(x) \log p_X(x) dx - \mathcal{L}_1 &= \int_{\mathbb{R}^d} p_X(x) \log \frac{p_{\widetilde{X}}(x)}{P_X(x)} dx \\ &\leq \int_{\mathbb{R}^d} p_X(x) \left(\frac{p_{\widetilde{X}}(x)}{P_X(x)} - 1 \right) dx \\ &= \int_{\mathbb{R}^d} p_{\widetilde{X}}(x) - p_X(x) dx = 0. \end{aligned} \tag{3.35}$$

To prove the right side of Eq. (3.34), we need to show that

$$\int_{\mathbb{R}^d} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx < \varepsilon. \tag{3.36}$$

We define $B_d(r) := \{x \in \mathbb{R}^d : \|x\| \leq r\}$ and $V_{d-1}(r)$ be the volume of $\{x \in \mathbb{R}^d : \|x\| = r\}$. For every $\varepsilon_1 < \min\{A_3 \exp(-\alpha), A_3 \exp(-\alpha K^2)\}$, K is the constant in Assumption 3.1, we let $K_1 = \sqrt{-\frac{1}{\alpha} \log \frac{\varepsilon_1}{A_3}}$, then $K_1 > \max\{1, K\}$. According to Eq. (3.19) in Assumption 3.1, $p_X(x) < A_3 \exp(-\alpha \|x\|^2) \leq \varepsilon_1$ over the domain $B_d(K_1)^c$. Define $\Omega_{\varepsilon_1} := \{x : p_X(x) \geq \varepsilon_1\}$, then Ω_{ε_1} is contained in $B_d(K_1)$. Since $p_X(x)$ is continuous, Ω_{ε_1} is closed. Moreover, as a bounded closed set in $\mathbb{R}^d$, Ω_{ε_1} is compact. The KL divergence between distributions p_X

and $p_{\widetilde{X}}$ has the following bound

$$\begin{aligned} D_{\mathsf{KL}}(p_X \,\|\, p_{\widetilde{X}}) &= \int_{\mathbb{R}^d} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &\leq \int_{p_X(x)\geq \varepsilon_1} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &\quad + \int_{\{p_X(x)<\varepsilon_1\}\cap B_d(K_1)} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &\quad + \int_{\{p_X(x)<\varepsilon_1\}\cap B_d(K_1)^c} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &:= I_1 + I_2 + I_3. \end{aligned} \tag{3.37}$$

Next we investigate all terms on the right side of Eq. (3.37). By Lemma 3.3, for every $\varepsilon_2 > 0$, there exists variable $\widetilde{X}$ such that

$$|p_X(x) - p_{\widetilde{X}}(x)| < C\varepsilon_2, \tag{3.38}$$

for any $x \in \Omega_{\varepsilon_1}$. Choosing $\varepsilon_2 < \frac{\varepsilon_1}{2C}$, we have $p_{\widetilde{X}}(x) > \varepsilon_1/2$ for the case $p_X(x) \geq \varepsilon_1$. Applying the mean value theorem, I_1 has the bound

$$\begin{aligned} I_1 &= \int_{p_X(x)\geq\varepsilon_1} p_X(x) \left(\log(p_X(x)) - \log(p_{\widetilde{X}}(x))\right) dx \\ &\leq \int_{p_X(x)\geq\varepsilon_1} p_X(x) \left|\frac{1}{\xi}\right| \left|p_X(x) - p_{\widetilde{X}}(x)\right| dx \\ &\leq \frac{2C\varepsilon_2}{\varepsilon_1} \int_{p_X(x)\geq\varepsilon_1} p_X(x) dx \leq \frac{2C\varepsilon_2}{\varepsilon_1}, \end{aligned} \tag{3.39}$$

where $\left|\frac{1}{\xi}\right| \leq \frac{2}{\varepsilon_1}$ because ξ is between $p_X(x)$ and $p_{\widetilde{X}}(x)$. Since we define $\varepsilon_2 < \frac{\varepsilon_1}{2C_1}$, we are able to let $\varepsilon_2/\varepsilon_1 \to 0$ to guarantee $I_1 \to 0$.

Positive density functions $p_X(x)$, $p_{\widetilde{X}}(x)$ are continuous, thus $\log \frac{p_X(x)}{p_{\widetilde{X}}(x)}$ is continuous and bounded over $B_d(K)$, i.e., $\log \frac{p_X(x)}{p_{\widetilde{X}}(x)} < C_1$ for $x \in B_d(K)$. By Eq. (3.29) in Assumption 3.2, it follows that $\log \frac{p_X(x)}{p_{\widetilde{X}}(x)} < C_2 K_1^2$ for $x \in B_d(K_1) \setminus B_d(K)$. Hence, $\log \frac{p_X(x)}{p_{\widetilde{X}}(x)} <$

$\max\{C_1,\ C_2K_1^2\}$ for $x \in B_d(K_1)$. The term I_2 has the bound

$$\begin{aligned} I_2 &= \int_{\{p_X(x)<\varepsilon_1\}\cap B_d(K_1)} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &\leq \int_{\{p_X(x)<\varepsilon_1\}\cap B_d(K_1)} \left| p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} \right| dx \\ &\leq C\varepsilon_1 \int_{B_d(K_1)} \max\{1,\ K_1^2\} dx \\ &= C\varepsilon_1 \frac{\pi^{d/2}}{\Gamma(d/2+1)} \max\{K_1^d,\ K_1^{d+2}\} \\ &\leq C \max\{\varepsilon_1 (\log \frac{\varepsilon_1}{A_3})^{d/2},\ \varepsilon_1 (\log \frac{\varepsilon_1}{A_3})^{d/2+1}\}, \end{aligned} \tag{3.40}$$

where C is a constant and $I_2 \to 0$ as $\varepsilon_1 \to 0$. The last inequality holds due to $K_1 = \sqrt{-\frac{1}{\alpha} \log \frac{\varepsilon_1}{A_3}}$. Lastly we have the bound for I_3

$$\begin{aligned} I_3 &= \int_{\{p_X(x)<\varepsilon_1\}\cap B_d(K_1)^c} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &= \int_{B_d(K_1)^c} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx \\ &\leq C \int_{B_d(K_1)^c} \exp(-\alpha \|x\|^2) \|x\|^2 dx. \end{aligned} \tag{3.41}$$

According to the identity $\int_{B_d(R_2)\setminus B_d(R_1)} y(\|x\|) dx = V_{d-1}(1) \int_{R_1}^{R_2} y(r) r^{d-1} dr$, we have

$$\begin{aligned} I_3 &\leq C V_{d-1}(1) \int_{K_1}^{+\infty} \exp(-\alpha r^2) r^2 r^{d-1} dr \\ &= C \int_{K_1}^{+\infty} \exp(-\alpha r^2) r^{d+1} dr \\ &= C \left(\left[\frac{\exp(-\alpha r^2) r^d}{-2\alpha} \right]_{K_1}^{+\infty} + \frac{d+1}{2\alpha} \int_{K_1}^{+\infty} \exp(-\alpha r^2) r^d dr \right) \\ &= C \left(\left[\frac{\exp(-\alpha r^2) r^d}{-2\alpha} \right]_{K_1}^{+\infty} + \ldots + \frac{(d+1)!}{(2\alpha)^{d+1}} \int_{K_1}^{+\infty} \exp(-\alpha r^2) dr \right), \end{aligned} \tag{3.42}$$

where $I_3 \to 0$ as $K_1 \to +\infty$. Since $K_1 = \sqrt{-\frac{1}{\alpha} \log \frac{\varepsilon_1}{A_3}}$, we have $I_3 \to 0$ as $\varepsilon_1 \to 0$. □

In summary, Theorem 3.1 shows that the loss function $\mathcal{L}_1$ converges to the entropy of $p_X(x)$ as h converges to f. The following Theorem 3.2 proves the convergence of the continuous loss function $\mathcal{L}_2$ as defined in Eq. (3.28).

Theorem 3.2. *For arbitrarily small $\varepsilon > 0$, there exists a single-hidden-layer neural network g such that*

$$\mathcal{L}_2 < \varepsilon, \tag{3.43}$$

in any compact set $\Omega \subseteq \mathbb{R}^d$.

Proof. By Assumption 3.1, p_X is bounded. $\mathcal{L}_2$ has the following bound:

$$\begin{aligned}
\mathcal{L}_2 =& \int_\Omega p_X(x) \left(\|g(h(x)) - x\|^2 dx + \int_\Omega |\det J_g(h(x)) \det J_h(x) - 1| \right) dx \\
\leq & C \int_\Omega \|g(h(x)) - x\|^2 dx + \int_\Omega |\det J_g(h(x)) \det J_h(x) - 1| dx \\
= & C \Big(\int_\Omega \|g(h(x)) - h^{-1}(h(x))\|^2 dx \\
& + \int_\Omega |\det J_g(h(x)) (\det J_{h^{-1}}(h(x)))^{-1} - 1| dx \Big) \\
= & C \Big(\int_{h(\Omega)} \|g(y) - h^{-1}(y)\|^2 |\det J_{h^{-1}}(y)| dy \\
& + \int_{h(\Omega)} |\det J_g(y) - \det J_{h^{-1}}(y)| dy \Big) := C(I_1 + I_2).
\end{aligned} \tag{3.44}$$

Since h is continuous in a compact set Ω, by the extreme value theorem, h is bounded in Ω, i.e., there exists a constant M such that $|h(x)| \leq M$ for all $x \in \Omega$. Therefore, $h(\Omega) \subseteq B_d(M)$. Let g be the approximation of h^{-1}, and the term I_1 has the bound

$$\begin{aligned}
I_1 =& \int_{h(\Omega)} \|g(y) - h^{-1}(y)\|^2 |\det J_{h^{-1}}(y)| dy \\
\leq & \int_{B_d(M)} \|g(y) - h^{-1}(y)\|^2 |\det J_{h^{-1}}(y)| dy \\
= & \int_{B_d(M)} \left(\sum_{i=1}^d |g_i(y) - h_i^{-1}(y)|^2 \right) |\det J_{h^{-1}}(y)| dy \leq \int_{B_d(M)} d\varepsilon_1^2 C dy \leq C\varepsilon_1^2,
\end{aligned} \tag{3.45}$$

where the last inequality holds by Lemma 3.1 with $k = 0$ and the determinant of Jacobian of h^{-1} has the upper bound in Eq. (3.24). Let $\mathcal{S}_d$ be the set of all permutation of $\{1, \ldots, d\}$, we have

$$|\det J_g(y) - \det J_{h^{-1}}(y)| = \left| \sum_{\sigma \in \mathcal{S}_d} \left(\operatorname{sgn}(\sigma) \left(\prod_{i=1}^{d} \frac{\partial g_i(y)}{\partial x_{\sigma_i}} - \prod_{i=1}^{d} \frac{\partial h_i^{-1}(y)}{\partial x_{\sigma_i}} \right) \right) \right| < C\varepsilon_1 \quad (3.46)$$

for any $y \in B_d(M)$, where the last inequality holds by Lemma 3.1 with $k = 1$. We have the bound for I_2

$$I_2 = \int_{h(\Omega)} |\det J_g(y) - \det J_{h^{-1}}(y)| dy \leq \int_{B_d(M)} |\det J_g(y) - \det J_{h^{-1}}(y)| dy \leq C\varepsilon_1. \quad (3.47)$$

The convergence of $\mathcal{L}_2$ to 0 as ε_1 approaches 0 is established by Eqs. (3.44), (3.45) and (3.47). By choosing ε_1 sufficiently small, we complete the proof. □

3.3.3 Convergence of the KL Divergence

The analysis in Section 3.3.2 shows that the convergence of $\mathcal{L}_1$ and $\mathcal{L}_2$ is a necessary condition to obtain the convergent approximation discussed in Section 3.3.1, which verifies that appropriateness of the loss function used to train the PR-NF model. Here we show that the convergence of $\mathcal{L}_1$ and $\mathcal{L}_2$ plus some mild assumptions on the h and g is sufficient to ensure the convergence of the KL divergence between p_X and $p_{\widehat{X}}$.

Assumption 3.3. *We assume that the target random variable X has finite second-order moment, and there exists a positive constant A_5 such that*

$$\|\nabla p_{\widehat{X}}(x)\| \leq A_5(\|x\| + 1)p_{\widehat{X}}(x), \quad (3.48)$$

where $p_{\widehat{X}}(x)$ is defined in Eq. (3.26)*.*

Remark 3.1. *All these assumptions are valid for normally distributed random variables. So we can say that they are "normal-like" assumptions.*

Theorem 3.3. *For any given $\varepsilon > 0$, under the assumptions of Theorems 3.1, 3.2 and Assumption 3.3, the KL divergence between target variable X and the approximation $\widehat{X}$ defined in Eq.* (3.25) *satisfies $D_{\mathsf{KL}}(p_X \| p_{\widehat{X}}) < \varepsilon$.*

Proof. For arbitrarily small $\varepsilon_1 > 0$, under Theorems 3.1, we have

$$\mathcal{L}_1 \leq -\int_{\mathbb{R}^d} p_X(x) \log p_X(x) dx + \varepsilon_1, \tag{3.49}$$

and for arbitrarily small $\varepsilon_2 > 0$, under Theorems 3.2, we have

$$\mathcal{L}_2(\mathbb{R}^d) = \int_{\mathbb{R}^d} p_X(x) \left(\|g(h(x)) - x\|^2 + |\det J_g(h(x)) \det J_h(x) - 1| \right) dx < \varepsilon_2. \tag{3.50}$$

To simplify the notation, we denote $M(x) := g(h(x))$, Then Eq. (3.50) is rewritten as

$$\mathcal{L}_2(\mathbb{R}^d) = \int_{\mathbb{R}^d} p_X(x) (\|M(x) - x\|^2 + |\det J_M(x) - 1|) dx < \varepsilon_2. \tag{3.51}$$

Since $\widehat{X} = g(Z) = g(h(\widetilde{X})) = M(\widetilde{X})$, by the change of variables formula we have

$$p_{\widetilde{X}}(x) = p_{\widehat{X}}(M(x)) |\det J_M(x)|. \tag{3.52}$$

The KL divergence between distributions P_X and $P_{\widehat{X}}$ has the following bound

$$\begin{aligned} D_{\mathsf{KL}}(p_X \| p_{\widehat{X}}) &= \int_{\mathbb{R}^d} p_X(x) \log \frac{p_X(x)}{p_{\widehat{X}}(x)} dx \\ &\leq \int_{\mathbb{R}^d} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx + \int_{\mathbb{R}^d} p_X(x) \log \frac{p_{\widetilde{X}}(x)}{p_{\widehat{X}}(x)} dx \\ &= \int_{\mathbb{R}^d} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx + \int_{\mathbb{R}^d} p_X(x) \log \frac{p_{\widehat{X}}(M(x)) |\det J_M(x)|}{p_{\widehat{X}}(x)} dx \\ &= \int_{\mathbb{R}^d} p_X(x) \log \frac{p_X(x)}{p_{\widetilde{X}}(x)} dx + \int_{\mathbb{R}^d} p_X(x) \log \frac{p_{\widehat{X}}(M(x))}{p_{\widehat{X}}(x)} dx \\ &\quad + \int_{\mathbb{R}^d} p_X(x) \log |\det J_M(x)| dx := I_1 + I_2 + I_3. \end{aligned} \tag{3.53}$$

According to Eq. (3.49), we have $I_1 < \varepsilon_1$. Apply the mean value theorem, I_2 has the bound

$$\begin{aligned} I_2 &= \int_{\mathbb{R}^d} p_X(x) \log \frac{p_{\widehat{X}}(M(x))}{p_{\widehat{X}}(x)} dx \\ &= \int_{\mathbb{R}^d} p_X(x) \Big(\log(p_{\widehat{X}}(M(x))) - \log(p_{\widehat{X}}(x)) \Big) dx \\ &= \int_{\mathbb{R}^d} p_X(x) \left(\frac{\nabla p_{\widehat{X}}(\xi)}{p_{\widehat{X}}(\xi)} \cdot (M(x) - x) \right) dx, \end{aligned} \tag{3.54}$$

where ξ is between $M(x)$ and x. By Eq. (3.48) in Assumption 3.3, we have

$$\begin{aligned} I_2 \leq & C \int_{\mathbb{R}^d} p_X(x) (\|\xi\| + 1) \|M(x) - x\| dx \\ \leq & C \int_{\mathbb{R}^d} p_X(x) (\|M(x) - x\| + \|x\| + 1) \|M(x) - x\| dx \\ = & C \Big(\int_{\mathbb{R}^d} p_X(x) \|M(x) - x\|^2 dx + \int_{\mathbb{R}^d} p_X(x) \|x\| \|M(x) - x\| dx \\ & + \int_{\mathbb{R}^d} p_X(x) \|M(x) - x\| dx \Big) \\ \leq & C \Big[\varepsilon_2 + \left(\int_{\mathbb{R}^d} p_X(x) \|x\|^2 dx \right)^{\frac{1}{2}} \left(\int_{\mathbb{R}^d} p_X(x) \|M(x) - x\|^2 dx \right)^{\frac{1}{2}} \\ & + \left(\int_{\mathbb{R}^d} p_X(x) dx \right)^{\frac{1}{2}} \left(\int_{\mathbb{R}^d} p_X(x) \|M(x) - x\|^2 dx \right)^{\frac{1}{2}} \Big] \leq C(\varepsilon_2 + \varepsilon_2^{\frac{1}{2}}). \end{aligned} \tag{3.55}$$

Since $\log|t| \leq |t-1|$ for every $t \in \mathbb{R}$, I_3 has the bound

$$I_3 = \int_{\mathbb{R}^d} p_X(x) \log|\det J_M(x)| dx \leq \int_{\mathbb{R}^d} p_X(x) |\det J_M(x) - 1| dx \leq \varepsilon_2. \tag{3.56}$$

Therefore,

$$D_{\mathsf{KL}}(p_X \| p_{\widehat{X}}) \leq \varepsilon_1 + C(\varepsilon_2 + \varepsilon_2^{\frac{1}{2}}). \tag{3.57}$$

Let $\varepsilon_1 < \varepsilon/2$ and $\varepsilon_2 < \min\{\varepsilon^2/(16C^2), 1\}$, then we have

$$D_{\mathsf{KL}}(p_X \| p_{\widehat{X}}) \leq \varepsilon_1 + C(2\varepsilon_2^{\frac{1}{2}}) \leq \frac{\varepsilon}{2} + C\frac{\varepsilon}{2C} \leq \varepsilon. \tag{3.58}$$

Thus, $D_{\mathsf{KL}}(p_X \| p_{\widehat{X}}) \leq \varepsilon$. We complete the proof. □

3.4 Numerical Examples and Applications

In this section we present numerical experiments demonstrating the superior performance of the PR-NF model in comparison with the standard MC method. The example in Sec.3.4.1 benchmarks the accuracy of our method for a problem with known analytical ground-truth solution. In Sec. 3.4.2 we present an application to plasma physics and in Sec.3.4.3 an application to fluid mechanics.

Since in these cases there are not known analytical solutions, the ground-truth corresponds to MC simulations with sufficiently large samples. In all numerical simulations, the PyTorch machine learning framework has been implemented with CUDA GPU. The PR-NF model uses the Adam optimizer [43] and the Tanh activation function.

3.4.1 Verification of Algorithm Accuracy

In this subsection, we use a one-dimensional nonlinear SDE and a ten-dimensional linear SDE to verify the accuracy of the proposed algorithm. We first consider the following

one-dimensional SDE

$$X_t = X_0 + \int_0^t b(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s \quad \text{with } X_0 \in [0, L], \tag{3.59}$$

where $b(s, X_s) = 2\sqrt{X_s} + 1$, $\sigma(s, X_s) = 2\sqrt{X_s}$, $L = 5$, $t = 0.1$. The analytical solution of this SDE is

$$X_t = (\sqrt{X_0} + t + W_t)^2, \tag{3.60}$$

where X_0 is the initial condition, and the corresponding conditional distribution is

$$p_{X_t|X_0}(x|x_0) = \frac{1}{2\sqrt{2\pi t x}} \left[\exp\left(\frac{-(\sqrt{x} - \sqrt{x_0} - t)^2}{2t} \right) + \exp\left(\frac{-(\sqrt{x} + \sqrt{x_0} + t)^2}{2t} \right) \right]. \tag{3.61}$$

The distribution of X_t subject to an initial distribution $p_{X_0}(x_0)$ can be computed by the convolution with $p_{X_t|X_0}(x|x_0)$. Since the analytical solution is known, we use this example to test the accuracy of the propose method.

The training dataset consists of $N_{\text{train}} = 20000$ samples with initial positions $\{X_0^{(n)}\}_{n=1}^{N_{\text{train}}}$ sampled from a uniform distribution over the domain $\mathcal{D} = [0, 5]$, and terminal positions $\{X_t^{(n)}\}_{n=1}^{N_{\text{train}}}$ generated by MC simulations of Eq. (3.59). The neural network has $N_{\text{layer}} = 1$ hidden layer with $N_{\text{neuron}} = 256$ neurons. The hyperparameter λ in the loss function is selected according to the method described in Sec. 3.2.3. The left panel of Fig. 3.3 shows the cross entropy $H(\lambda)$ defined in Eq. (3.16) with different λ. As it shown, the PR-NF model attains the minimum of $H(\lambda)$ when $\lambda = 50$. The middle panel and right panel of Fig. 3.3 correspond to the accuracy performance of the proposed PR-NF model with two cases $\lambda = 1$ and $\lambda = 50$, respectively. The red curve is the exact density function and the histogram is synthesized by the PR-NF model. A good agreement is shown in the right panel ($\lambda = 50$). In contrast, $\lambda = 1$ is too small to guarantee the reversibility. Without the tuning process of λ, the PR-NF model may fail to balance the $\mathcal{L}_1$ and $\mathcal{L}_2$, e.g., the middle panel of Fig. 3.3. The minimization of cross entropy $H(\lambda)$ in Section 3.2.3 is an effective

and appropriate approach to choose λ. An optimal parameter λ is necessary to obtain an accurate surrogate model.

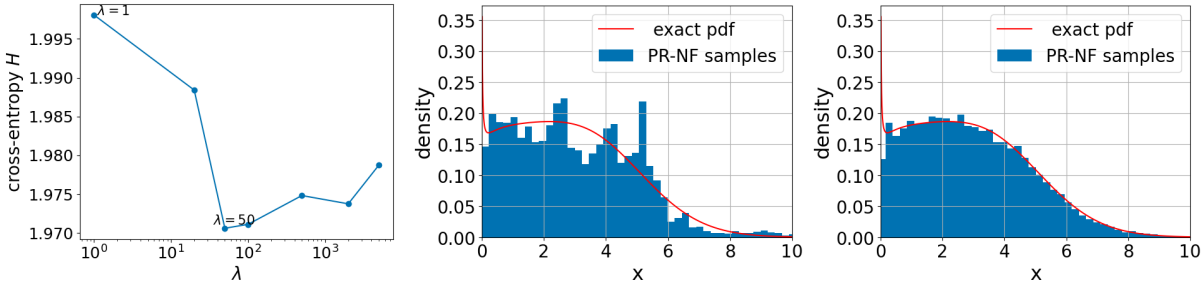

Figure 3.3: Cross entropy $H(\lambda)$ and fitting performance for $\lambda = 1$ and $\lambda = 50$.

We now use the well-trained PR-NF model ($\lambda = 50$) to sample X_t with variable initial distributions. The top row of Fig. 3.4 shows four different initial distributions, denoted as "δ func", "bar", "sin2", and "ricker". The bottom row of Fig. 3.4 shows corresponding histogram of X_t generated by the PR-NF model, where a good agreement with the exact density is observed. The results demonstrate that the well-trained PR-NF model can handle different initial conditions without additional training process. Figure 3.5 shows the decay of the loss function of training dataset (left panel) and the decay of the Kullback–Leibler (KL) divergence of the four initial distributions (right panel). The KL divergence formula is defined as the relative entropy from the approximate density (generated from PR-NF) p_{approx} to the exact density p_{exact}, i.e.,

$$D_{\text{KL}}(p_{\text{exact}} \,\|\, p_{\text{approx}}) = \int_{-\infty}^{\infty} p_{\text{exact}}(x) \log \left(\frac{p_{\text{exact}}(x)}{p_{\text{approx}}(x)} \right) dx, \tag{3.62}$$

where D_{KL} can be approximated by the Riemann sum over a uniform mesh or the mean value of $\log \left(\frac{p_{\text{exact}}(x)}{p_{\text{approx}}(x)} \right)$ over the training dataset $\mathcal{X}$ based on the exact solution in Eq. (3.60). We monitor the KL divergence for different distributions during the training process. The consistency of decays between loss function and KL divergence illustrates that the loss function $\mathcal{L}$ defined in Eq. (3.11) involving the reversibility error is effective to be regarded as a loss function for normalizing flow.

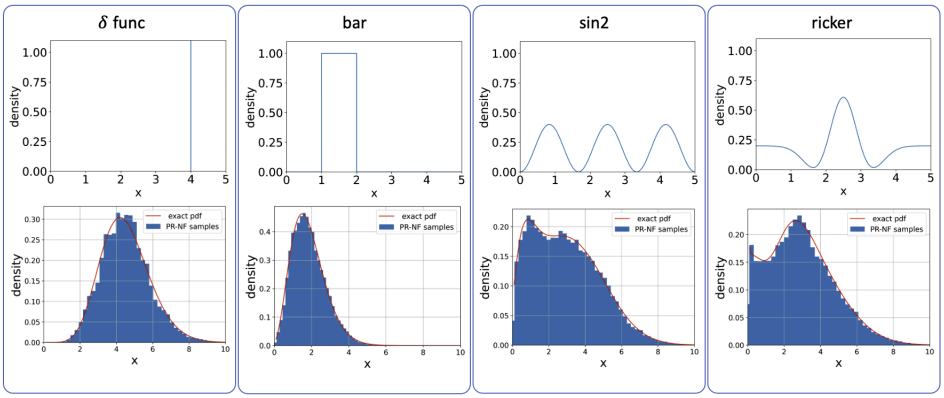

Figure 3.4: The accuracy performance of the well-trained PR-NF model for test dataset.

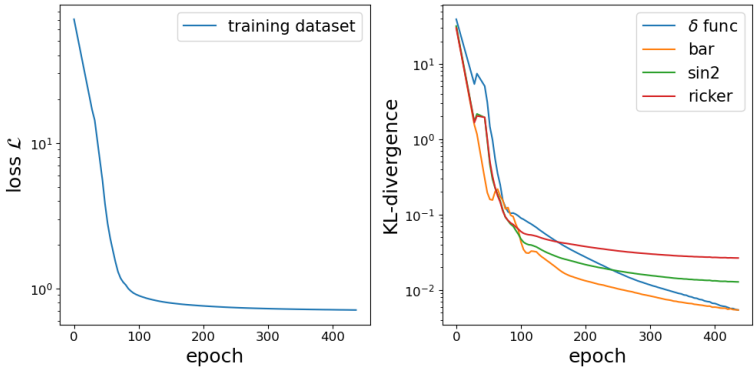

Figure 3.5: The decay of loss function of training dataset and the decay of KL divergence of four test dataset.

To illustrate the application of our method to high-dimensional problems, consider the ten-dimensional SDE

$$X_t = X_0 + \int_0^t X_s ds + \int_0^t K X_s dW_s \quad \text{with } X_0 \in \mathcal{D}, \tag{3.63}$$

where $t = 1$, $\mathcal{D} = [0,1]^{10}$, $K = \frac{1}{2}\begin{pmatrix} 1 & 1 & & & \\ & \ddots & \ddots & & \\ & & & \ddots & \\ & & & 1 & 1 \end{pmatrix}_{10\times 10}$. In this case the analytical solution X_t is given by

$$X_t = \exp((\mathbf{I}_{10} - K^2/2)t + KW_t)X_0. \tag{3.64}$$

The training dataset consists of $N_{\text{train}} = 20000$ samples where X_0 are sampled from an uniform initial distribution over the domain $\mathcal{D}$. The neural network has $N_{\text{hidden}} = 1$ hidden layer with $N_{\text{neuron}} = 256$ neurons. The left panel of Fig. 3.6 shows the cross entropy as function of λ, with a minimum at $\lambda = 100$.

We use the well-trained PR-NF model to test four initial distributions, one is the normal distribution $x_{\text{test}} \sim \mathcal{N}(0.5, 0.1 \cdot \mathbf{I}_{10})$, other three are nonlinear transformations of the normal distribution, square:= x^2_{test}, log:= $\ln(|x_{\text{test}}| + 1)$, and sin:= $\sin(x^2_{\text{test}})$. The middle panel of Fig. 3.6 shows the decay of loss function of the training dataset. And the decay of KL divergence of four test dataset is shown in the right panel. A good consistency is observed.

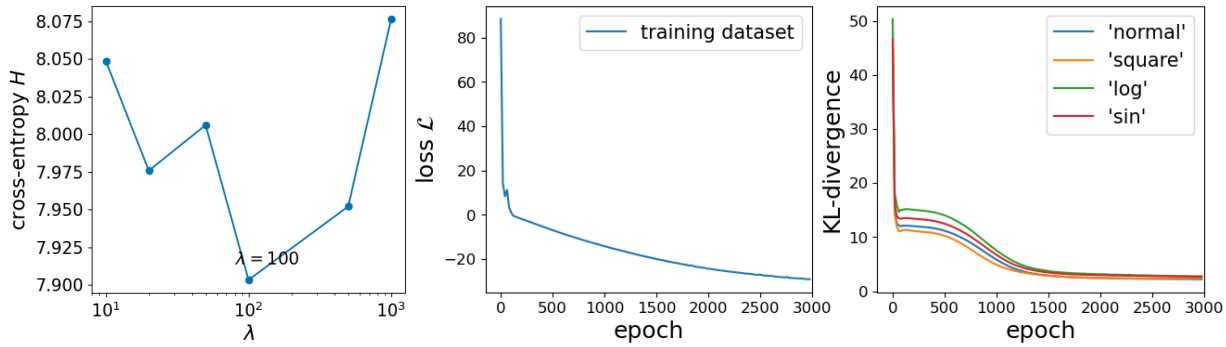

Figure 3.6: Numerical results for the ten-dimensional linear SDE.

3.4.2 Generation of Runaway Electrons in Magnetically Confined Nuclear Fusion Plasmas

Modeling and simulation of plasmas is an extremely complex computational problem involving, among other aspects, the trajectories of the plasma particles (electrons and ions), the evolution of external and self-generated electromagnetic fields, as well as energy and particle sources and sinks. Particle-based simulations (usually known as Monte Carlo simulations in the plasma physics literature) are based on the solution of stochastic differential equations of the form in Eq.(3.1) with X_t denoting the particles' coordinates in the d-dimensional phase space. In the full-orbit description $d = 6$, but reduced models with

$d < 6$ are commonly used. The deterministic term, $b\,ds$, is given by the electromagnetic forces, and the stochastic term, $\sigma\,dW_s$, is used to model collisional and diffusive processes. This is a strongly multi-scale problem since there is a big disparity of spatiotemporal scales in the evolution of the orbits and the electromagnetic fields. In addition, to avoid statistical sampling errors, these equations need to be solved for very large number of particles and large sets of different initial conditions. The goal of this section is to discuss how the proposed PR-NF method can be used to overcome some of these computational challenges in the case of magnetically confined high temperature plasmas in toroidal geometry. This configuration (known as tokamak) is the leading approach to achieve the elusive goal of controlled nuclear fusion for electricity production. The specific problem we consider is the simulation of relativistic runaway (RE) electrons produced during magnetic disruptions, see for example Ref. [10] and references therein. Understanding this problem is critical for the safe operation of nuclear fusion reactors.

The model of interest consists of the following set of SDEs

$$\begin{cases} dp = \left[E\xi \, - \dfrac{\gamma p}{\tau}(1-\xi^2) - C_F(p) + \dfrac{1}{p^2}\dfrac{\partial}{\partial p}\left(p^2 C_A\right)\right] dt + \sqrt{2C_A}\, dW_p, \\ d\xi = \left[\dfrac{E\,(1-\xi^2)}{p} + \dfrac{\xi(1-\xi^2)}{\tau\gamma} - 2\xi\dfrac{C_B}{p^2}\right] dt + \dfrac{\sqrt{2C_B}}{p}\,\sqrt{1-\xi^2}\, dW_\xi\,. \end{cases} \tag{3.65}$$

This two-dimensional reduced model tracks the evolution of the magnitude of the relativistic momentum, p, and $\xi = \cos\eta$, where η is the pitch angle between the particle velocity and the magnetic field which is assumed fixed and given.

dW_p and dW_ξ are independent standard Brownian motions, E is the electric field, and τ the radiation damping time scale. The collision operator includes the momentum diffusion, C_A, pitch angle scattering, C_B, and Coulomb drag, C_F, which are defined as

$$C_A(p) = \bar{\nu}_{ee}\,\bar{v}_T^2\,\frac{\psi(y)}{y}, \quad C_F(p) = 2\,\bar{\nu}_{ee}\,\bar{v}_T\,\psi(y),$$

$$C_B(p) = \frac{1}{2}\,\bar{\nu}_{ee}\,\bar{v}_T^2\;\frac{1}{y}\left[Z + \phi(y) - \psi(y) + \frac{y^2}{2}\delta^4\right],$$

$$\phi(y) = \frac{2}{\sqrt{\pi}}\int_0^y e^{-s^2}ds, \quad \psi(y) = \frac{1}{2y^2}\left[\phi(y) - y\frac{d\phi}{dy}\right], \quad y = \frac{1}{\bar{v}_T}\frac{p}{\gamma},$$

$$\gamma = \sqrt{1 + (\tilde{\delta}p)^2}, \quad \tilde{\delta} = \frac{\tilde{v}_T}{c} = \sqrt{\frac{2\tilde{T}}{mc^2}}, \quad \delta = \frac{\hat{v}_T}{c} = \sqrt{\frac{2\hat{T}_f}{mc^2}}, \quad \bar{v}_T = \sqrt{\frac{\hat{T}_f}{\tilde{T}}},$$

$$\bar{\nu}_{ee} = \left(\frac{\tilde{T}}{\hat{T}_f}\right)^{3/2}\frac{\ln\hat{\Lambda}}{\ln\tilde{\Lambda}}, \quad \ln\tilde{\Lambda} = 14.9 - \frac{1}{2}\ln 0.28 + \ln\tilde{T}, \quad \ln\hat{\Lambda} = 14.9 - \frac{1}{2}\ln 0.28 + \ln\hat{T}_f,$$

with Z and c denoting the ion effective charge and the speed of light, respectively. Further details on this model can be found in Ref. [18, 79] and references therein.

We are interested in the hot-tail acceleration of electrons due to the rapid cooling of the plasma (thermal quench) during a magnetic disruption [69, 54]. We use a fast cooling model for the temperature and an electric field from Ohm's law with a Spitzer resistivity temperature dependent model $E = E_0\left[\frac{\tilde{T}}{\hat{T}_f}\right]^{3/2}$, with typical model parameters $Z = 1$, $\tau = 6 \times 10^3$, $E_0 = 1/2000$, $\tilde{T} = 3$, $\hat{T}_f = 0.05$ and $mc^2 = 500$.

The integration domain is $p \in (p_{\text{min}}, p_{\text{max}})$ and $\theta \in (0, \pi)$, where $(p_{\text{min}}, p_{\text{max}}) = (0.5, 5)$, and $t_{\text{max}} = 26$. The different cases of initial conditions are obtained by sampling a family of Maxwellian distributions

$$f(p, \xi, t_0) = \frac{2p^2}{\pi^{1/2}p_0^3}e^{-(p/p_0)^2}, \quad \text{where } p_0 = \sqrt{\frac{\hat{T}_0}{\tilde{T}}}, \tag{3.66}$$

normalized as $\int_0^1\int_{-1}^1\int_{p_{\text{min}}}^{p_{\text{max}}} f_0(p,\xi)dpd\xi = 1$, and parameterized by $\hat{T}_0$. Exploration of the hot-tail generation of RE in this case requires the solution of this problem for different values of $\hat{T}_0$. In the standard Monte Carlo approach this is a computational intense problem because each time $\hat{T}_0$ changes the problem needs to be recomputed for the new initial condition. However, as it will be shown below, this is not the case with the proposed PR-NF method.

In this problem the training dataset $\mathcal{V}$ consists of $N_{\text{train}} = 30000$ samples with initial positions, $\{(p_0, \xi_0)^{(n)}\}$, $n = 1, \ldots, N_{\text{train}}$, uniformly distributed in the integration domain and terminal positions, $(p_{t_{\text{max}}}, \xi_{t_{\text{max}}})^{(n)}$, obtained by a direct Monte Carlo simulation of Eqs. (3.65). The neural network architecture of the PR-NF model uses $N_{\text{hidden}} = 1$ hidden layer and $N_{\text{neuron}} = 512$ neurons. To choose the hyperparameter λ, which controls the relative importance of the pseudo-reversivility and the negative log-likelihood losses, we follow the same procedure as in Section 3.4.1. Namely, we select the value of λ for which the cross entropy, $H(\lambda)$, has a minimum, which in this case corresponds to $\lambda = 100$.

To check the accuracy of the PR-NF model we consider a test set of $N_{\text{test}} = 20000$ samples drawn from the Maxwellian distribution in Eq. (3.66) with $\hat{T}_0 = 10$. Fig. 3.7 compares the final probability density function (pdf) at $t_{\text{max}} = 26$ computed with the PR-NF method and the direct Monte Carlo simulation. To construct the pdf from the N_{test} data points, we use a Gaussian kernel density estimation. The plots on the top row show the one-dimensional marginal pdfs for the pitch angle θ (left) and momentum p (right). The plots on the bottom row show the two-dimensional contour plots of the $\log_{10}$ of the pdfs for the PR-NF (left) and the MC (right) simulations. The local pointwise differences between PR-NF and MC methods are of the order 10^{-2}, which implies a small error in the evaluation of quantities of interest like the production of runaway electrons shown in Fig. 3.8.

To illustrate the performance of the PR-NF method in the computation of the final state for different initial conditions, we consider the quantity of interest

$$n_{\text{RE}} = \int_{-1}^{1} \int_{p^*}^{p_{\text{max}}} f_{t_{\text{max}}}(p, \xi) dp d\xi, \tag{3.67}$$

representing the total fraction of runaway electrons produced by the hot-tail mechanism during the thermal quench, where $f_{t_{\text{max}}}$ is the density function of electrons at $t_{\text{max}} = 26$, $p^* = 1.75$ is the RE energy threshold and p_{max} is a numerical upper bound. We consider a family of Maxwellian initial conditions of the form in Eq. (3.66) with $\hat{T}_0 = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.

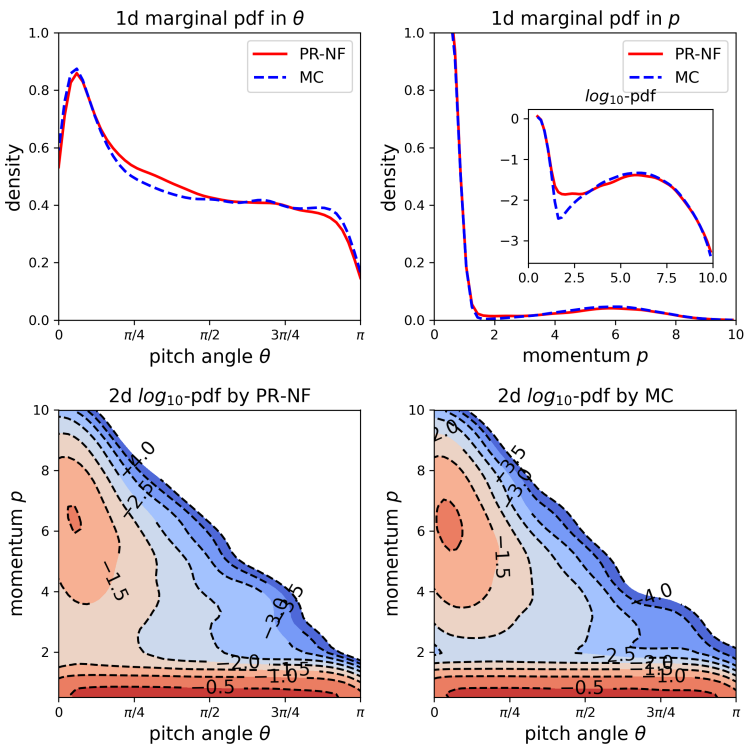

Figure 3.7: Final state of runaway electrons probability density function.

Figure 3.8 compares n_{RE} computed using a direct MC simulation that requires to solving the whole problem from the start for each value of $\hat{T}_0$, with the PR-NF method that reduces the computation to the direct evaluation of the final state using the trained neural network. This numerical simulation is based on $N_{\text{sample}} = 20000$ samples for different $\hat{T}_0$. By using a single PR-NF model, we can obtain the orbits of particles for ten initial conditions without any additional simulations or training. However, the Monte Carlo method has to repeatedly simulate the particles' orbits ten times. For the simulation, the time step size of the MC method is $\Delta t = 0.001$ and the number of epochs of the PR-NF method is $N_{\text{epoch}} = 20000$. We record the running wall-clock time (measured using the python function "time.time()") between these two methods. For the results in Fig. 3.8, the error between the PR-NF and MC method (viewed as the ground-truth) is acceptable. The accuracy of PR-NF model is enough to predict and analyze the profile of fraction runaway electrons production rate n_{RE}. Moreover, the total running time of the MC method of these ten cases of Maxwellian

distribution is around $C_{\text{MC}} = 4000$ sec. The total running time of the PR-NF method consists of two parts, the offline cost is around $C_{\text{offline}} = 1354$ sec and the online cost is around $C_{\text{online}} = 12$ sec. The PR-NF model is faster than the MC method, especially on the online cost. This efficiency advantage is more valuable when we handle a large number of Maxwellian distributions.

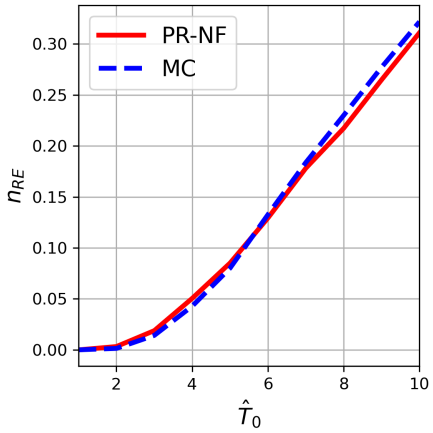

Figure 3.8: Quantity of interest n_{RE} in Eq. (3.67) with respect to $\hat{T}_0$ in Eq. (3.66).

3.4.3 Passive Scalar Advection-Diffusion Transport in a 3D Chaotic Flow

Understanding passive scalar transport is a problem of significant interest in fluid dynamics in general and environmental engineering, oceanography, and atmospheric sciences in particular. By passive we mean that the transported scalar exactly follows the given velocity field without modifying it.

As a specific example to illustrate the application of the PR-NF method to this problem, we consider the ABC (Arnold-Beltrami-Childress) flow [20] which is a three-dimensional incompressible velocity field which is an exact solution of Euler's equation. In Cartesian coordinates this velocity field has components $\mathbf{v} = (v_x, v_y, v_z)$ with $v_x = A\sin z + C\cos y$, $v_y = B\sin x + A\cos z$ and $v_z = C\sin y + B\cos x$. For a wide range

of the A, B and C parameters, this flow is not integrable. That is, it exhibits chaotic advection and the sensitive dependence on initial condition gives rise to rapid mixing and long-range transport [63].

In addition to advection, passive scalar transport is affected by diffusion. In particle based simulations this important effect can be modeled by adding a stochastic term. Following this approach we consider the following set of SDEs

$$\begin{cases} dx = P_e\,[A\,\sin z + C\,\cos y]\,dt + \,dW_x\,, \\ dy = P_e\,[B\,\sin x + A\,\cos z]\,dt + \,dW_y\,, \\ dz = P_e\,[C\,\sin y + B\,\cos x]\,dt + \,dW_z\,. \end{cases} \tag{3.68}$$

where dW_x, dW_y and dW_z are independent Brownian motions, and the Peclé number, P_e is a dimensionless parameter controlling the relative importance of advection and diffusion. When, $P_e \ll 1$, transport is dominated by diffusion and when, $P_e \gg 1$, transport is dominated by advection.

The initial conditions, $\{x_0^{(n)}, y_0^{(n)}, z_0^{(n)}\}_{n=1}^{N_{\text{train}}}$, of the training dataset consists of an ensemble of $N_{\text{train}} = 30000$ random points uniformly distributed in the cubic domain $\mathcal{D} = [0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$. The terminal positions, $\{x_{t_{\text{max}}}^{(n)}, y_{t_{\text{max}}}^{(n)}, z_{t_{\text{max}}}^{(n)}\}_{n=1}^{N_{\text{train}}}$, of the training dataset are obtained by a direct Monte Carlo simulation of Eq. (3.68). In the calculation presented we use $t_{\text{max}} = 2$, and model parameters $P_e = 3$, $A = 1$, $B = 1$, and $C = 0.25$. For the PR-NF neural network model we use $N_{\text{hidden}} = 1$ hidden layer with $N_{\text{neuron}} = 512$ neurons. Once the training process is done, the PR-NF model can be used as a surrogate model to predict the final state for various initial conditions, without the need to integrate Eqs. (3.68).

As a first test, we apply the PR-NF to predict the final state of an initial condition of the form

$$f(x, y, z, t_0) = H(x, y, z)\,\exp\left[-\left(\frac{x - x_c}{\sigma_x}\right)^2 - \left(\frac{y - y_c}{\sigma_y}\right)^2 - \left(\frac{z - z_c}{\sigma_z}\right)^2\right], \tag{3.69}$$

modeling, for example, a Gaussian-shaped cloud of a pollutant, with $\sigma_x = \pi/3$, $\sigma_y = \pi/5$, $\sigma_z = \pi/4$, localized at $(x_c, y_c, z_c) \in \mathcal{D}$, where $H(x, y, z) = 1$ if $(x, y, z) \in \mathcal{D}$ and $H(x, y, z) = 0$ otherwise. Figure 3.9 shows a good agreement between the final pdf of the passive scalar, $f(x, y, z, t_{\text{max}})$, computed with the PR-NF surrogate model and the pdf computed using direct Monte Carlo simulation for the case $(x_c, y_c, z_c) = (\pi/2, \pi, \pi)$. Since both methods are particle-based, once the discrete particle data have been obtained, the corresponding pdfs are constructed using a standard Gaussian-kernel estimation. To ease the comparison of these $3d$ pdfs, Fig. 3.9 shows the $1d$ marginal distributions, e.g., $g(x) = \int dy \int dz f(x, y, z, t_{\text{max}})$, and the $2d$ marginal distributions, e.g., $g(x, y) = \int dz f(x, y, z, t_{\text{max}})$. The mixing due to the combination of chaotic advection and diffusion rapidly displaces the scalar outside the $\mathcal{D}$ domain.

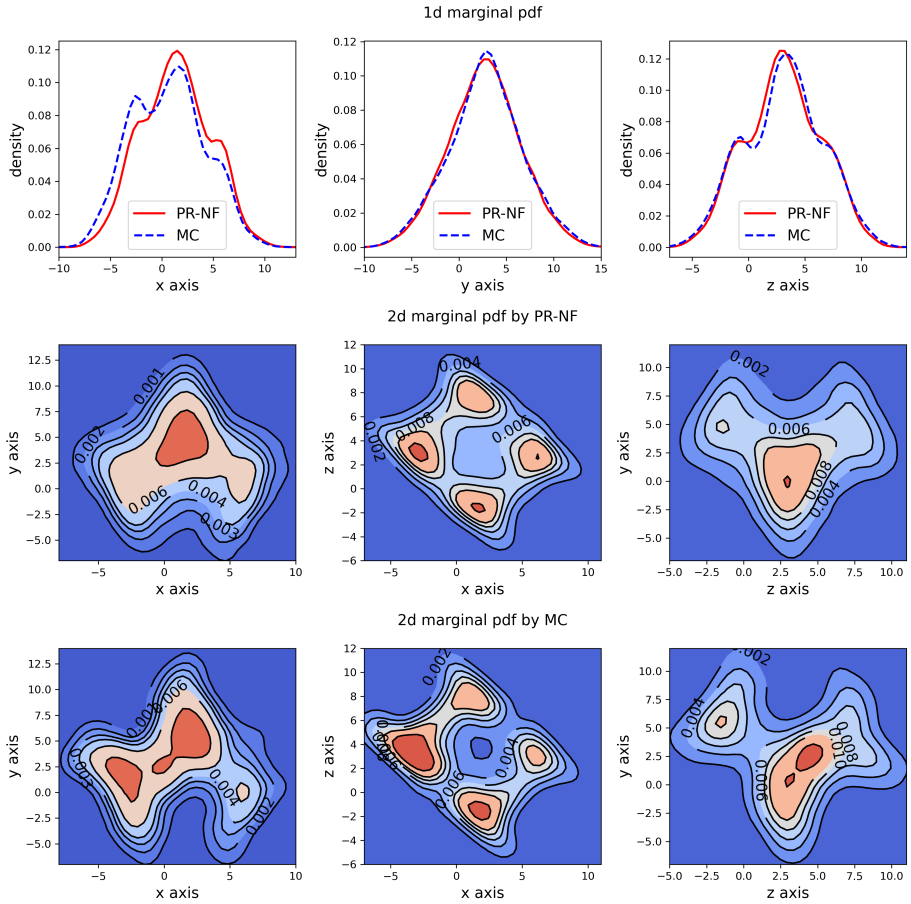

Figure 3.9: Final state of the probability density function in the ABC fluid velocity model.

As a second application we consider a "target" transport problem of interest, for example, to environmental fluid dynamics. The problem consists of finding the concentration of the scalar in a target domain, $\mathcal{T} = [x_{min}, x_{max}] \times (-\infty, \infty) \times [z_{min}, z_{max}]$, resulting from the transport of a concentrated pollutant cloud released inside the $\mathcal{D} = [0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$ domain. The initial cloud will be described using the Gaussian model in Eq. (3.69) with $\sigma_x = \pi/3$, $\sigma_y = \pi/5$, $\sigma_z = \pi/4$. The y-coordinate of the cloud will be fixed at $y_c = \pi$ and the goal is to compute the total scalar density in $\mathcal{T}$ as function of the (x_c, z_c) coordinates of the center of the initial cloud in $\mathcal{D}$. The quantity of interest in this case is

$$n_{\mathcal{T}}(x_c, z_c) = \int_{x_{\text{min}}}^{x_{\text{max}}} \int_{-\infty}^{\infty} \int_{z_{\text{min}}}^{z_{\text{max}}} f(x, y, z, t_{\text{max}}) dz dy dx \,, \tag{3.70}$$

and for the calculation presented here we will use $x_{\text{min}} = 0$, $x_{\text{max}} = \pi$, $z_{\text{min}} = 2\pi$, and $z_{\text{max}} = 3\pi$.

To perform this calculation a $N_x \times N_z$ uniform grid is constructed in the $(x_c, z_c) \in [0, 2\pi] \times [0, 2\pi]$ space and the $N_x N_z = N_{\text{ic}}$ grid nodes are used to define the ensemble $\{x_c^i, z_c^i\}_{i=1}^{N_{\text{ic}}}$. For each element of the ensemble, the standard direct Monte Carlo approach requires the solution of the system of SDE in Eq. (3.68) for N_{test} particles with initial conditions sampled from the Gaussian cloud in Eq. (3.69) centered at $(x_c^i, y_c = \pi, z_c^i)$. In total, this approach requires $N_{\text{ic}} N_{\text{test}}$ solutions of the SDEs. On the other hand, the PR-NF method only requires to solving the SDE in Eq. (3.68) for N_{train} initial conditions and, once the training is done, the computation of the $N_{\text{ic}} N_{\text{test}}$ initial conditions can be done by simply evaluating the surrogate neural network model.

In the case discussed here we use $N_x = N_z = 21$, and $N_{\text{test}} = 20000$. With the Monte Carlo method this requires solving 8.82×10^6 SDEs. For $t_{\text{final}} = 2$, with step size $\Delta t = 0.001$, the total run-time to produce the results reported in Fig. 3.10 was $C_{\text{MC}} = 5320$ sec. On the other hand, the training of the PR-NF neural network, like in the previous calculation, is done using $N_{\text{train}} = 30000$, $N_{\text{hidden}} = 1$ hidden layer with $N_{\text{neuron}} = 512$

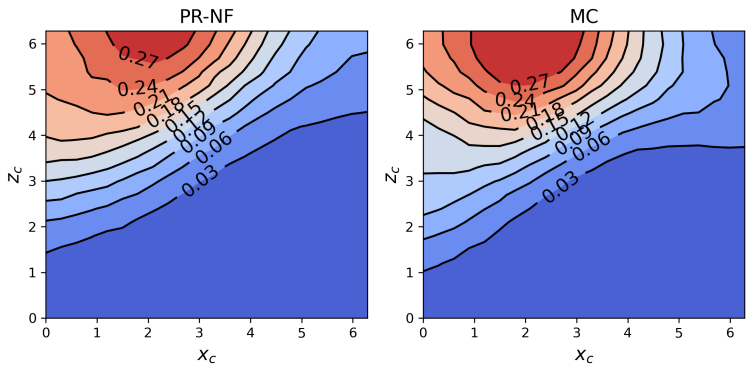

Figure 3.10: Contour plots of quantity of interest $n_\tau(x_c, z_c)$ in Eq. (3.70) computed using the PR-NF surrogate model and the direct MC method.

neurons, and $N_{\text{epoch}} = 20000$. For these parameters, the offline cost of the training is around $C_{\text{offline}} = 2200$ sec. But, once the training is done, the evaluation of the different initial conditions has minimal cost. In particular, the total PR-NF run-time to produce the results reported in Fig. 3.10 is of the order $C_{\text{online}} = 50$ sec. The contour plots of $n_{\mathcal{T}}$ shown in Fig. 3.10 provided evidence that the much more efficient PR-NF surrogate model reproduces well the results obtained with the direct, time-consuming, MC method for the target transport problem.

Chapter 4 Conditional Pseudo-Reversible Normalizing Flow for Surrogate Modeling in Quantifying Uncertainty Propagation

4.1 Problem Setting

This section introduces the surrogate modeling problem under consideration. We consider a physical model of the form

$$Y = f(X) + \varepsilon(X), \tag{4.1}$$

where $X \in \mathbb{R}^d$ is a d-dimensional random vector denoting the input of the model, $f : \mathbb{R}^d \to \mathbb{R}^s$ is a continuously differentiable function that represents the deterministic component of the model, $Y \in \mathbb{R}^s$ is a s-dimensional random vector denoting the output of the model, and $\varepsilon(X)$ denotes the additive random noise satisfying $\mathbb{E}[\varepsilon(X)] = 0$ and $\mathbb{E}[\varepsilon^2(X)] = v(X)$ with $v(\cdot)$ being a deterministic and bounded function. The goal is to efficiently compute two types of quantities of interest (QoIs), i.e.,

$$\text{(Forward QoI)} \quad Q_{\text{forward}}(x) = \int_{\mathbb{R}^s} q_{\text{forward}}(y)\, p(y|x)\, dy \;\text{ for }\; X = x, \tag{4.2}$$

where $q_{\text{forward}}(y)$ is a function of the model's output and $p(y|x)$ is the conditional probability density function (PDF) describing how the uncertainty of $\varepsilon(x)$ is propagated to Y for $X = x$, and

$$\text{(Inverse QoI)} \quad Q_{\text{inverse}}(y) = \int_{\mathbb{R}^d} q_{\text{inverse}}(x)\, p(x|y)\, dx \;\text{ for }\; Y = y, \tag{4.3}$$

where $q_{\text{inverse}}(x)$ is a function of the model's input and $p(x|y)$ is the conditional PDF of X for any fixed value of the model's output $Y = y$.

In practice, it is common to define a prior distribution of X to be one of the classical distributions, e.g., uniform or normal distributions, based on knowledge of the physical problem under consideration. Thus, we assume that the input X follows a prior PDF,

denoted by

$$X \sim p(x), \tag{4.4}$$

and we are capable of efficiently generating unlimited number of samples of X from the prior $p(x)$. The definitions of $p(x)$, $f(\cdot)$ and $\varepsilon(\cdot)$ can uniquely determine the joint distribution $p(x, y)$ and the marginal distribution $p(y) = \int p(x, y)dx$, such that the conditional PDF $p(x|y)$ in Eq. (4.3) is defined by

$$p(x|y) = \frac{p(x, y)}{p(y)}, \tag{4.5}$$

based on the conditional probability formula. Note that Eq. (4.5) is only the definition of the condition PDF $p(x|y)$, but we do not know how to draw samples from $p(x|y)$.

4.1.1 Challenges of Building Surrogate Models for the Conditional PDFs

When the physical model $f(X)$ is computationally expensive to evaluate, computing the QoIs in Eqs. (4.2) and (4.3) may become computationally infeasible. In this scenario, building a surrogate model to replace the time-consuming physics model is a widely used strategy to significantly improve the computational efficiency in computing the QoIs. Existing surrogate modeling approaches, e.g., polynomial chaos, kernel methods, or neural networks, mainly focus on approximating the deterministic function $f(X)$ in Eq. (4.1). However, the strategy of approximating $f(X)$ has the following disadvantages:

- It requires the ability to generate samples of the noise $\varepsilon(X)$ in order to obtain samples of the conditional PDF $p(y|x)$. When we do not know the distribution of $\varepsilon(X)$, we cannot use the deterministic surrogate of $f(X)$ to compute the forward QoI in Eq. (4.2).
- It requires another sampling method, e.g., Markov Chain Monte Carlo (MCMC), to generate samples from the conditional PDF $p(x|y)$ to compute the inverse QoI in Eq. (4.3).

To address these challenges, we intend to utilize a normalizing flow based generative model to directly learn how to generate samples from the conditional PDFs $p(y|x)$ in

Eq. (4.2) and $p(x|y)$ in Eq. (4.3). The trained normalizing flow models will serve as the surrogate models for the conditional PDFs, such that we can generate unlimited samples of the conditional PDFs to efficiently compute the QoIs in Eqs. (4.2) and (4.3).

4.2 The Pseudo-Reversible Normalizing Flow (PR-NF) for Learning Conditional PDFs

In this section, we explain in detail the proposed conditional generative model for computing the QoIs. To proceed, we define the training dataset as

$$\mathcal{D}_{\text{train}} := \left\{x^{(n)}, y^{(n)}\right\}_{n=1}^{N} \quad \text{with} \quad y^{(n)} = f\left(x^{(n)}\right) + \varepsilon^{(n)}, \tag{4.6}$$

where $y^{(n)}$ is the output of the physical model in Eq. (4.1) for the input $x^{(n)}$. Hereafter, we assume a purely data-driven scenario, which means we will need to train a surrogate model of the conditional PDFs of interest without using any information about the model f and the noise ε. Taking $p(y|x)$ as an example, our objective is to build and train a conditional generative model, denoted by

$$Y|X \approx \widehat{Y} = G(Z, X; \theta_G), \tag{4.7}$$

where Z follows the standard normal distribution $\mathcal{N}(0, \mathbf{I}_p)$, X is the input of the physics model in Eq. (4.1), θ_G denotes the trainable parameters of G. After training, the output of the generator approximates the target conditional random variable $Y|X$, such that the forward QoI in Eq.(4.2) can be efficiently computed by drawing samples from $\mathcal{N}(0, \mathbf{I}_p)$ and pushing through the generator G. A similar generator can be constructed for the conditional PDF $p(x|y)$. In this effort, we construct the conditional generative model in Eq. (4.7) by the PR-NF architecture introduced in the next subsection.

4.2.1 The Architecture of Conditional PR-NF Model

The standard pseudo-reversible architecture is similar to auto-encoders except that the bottleneck layer has the same width as the input and output layers. Compared to exactly reversible neural networks, the pseudo-reversible architecture has two advantages. First, it allows the use of simple fully-connected networks which greatly simplifies the implementation of normalizing flow models. Second, it allows us to conduct rigorous convergence analysis of normalizing flow models, which has a great significance from the mathematics perspective. In this section, we will explain how to extend the standard NF to the conditional PR-NF.

For notational simplicity, we introduce a new random vector W defined by

$$W := (X, Y|X) \in \mathbb{R}^{d+s}, \tag{4.8}$$

which is the concatenation of X and $Y|X$. A normalizing flow model includes an encoding transport map h and a decoding transport map g, i.e.,

$$Z = h(W; \theta_h) \quad \text{and} \quad \widehat{W} = g(Z; \theta_g), \tag{4.9}$$

where $\widehat{W}$ is the approximation of W provided by the normalizing flow model, and θ_h, θ_g are trainable parameters of the transformations. In standard normalizing flow models, e.g., MAF[53] and Real NVPs[19], the transport maps h and g are usually defined by exact reversible neural networks, such that $g = h^{-1}$ and $\widehat{W} = W$. In practice, the enforcement of reversibility could limit the expressive power and make it difficult to conduct any theoretical analysis. In PR-NF, h and g are defined by two independent fully-connected neural networks and the pseudo-reversibility is enforced by incorporating a soft constraint $\|\widehat{W} - W\|_2^2$ into the loss function introduced in Section 4.2.2. Let $p_W(w)$ and $p_Z(z)$ represent the PDFs of variables W and Z, respectively. The connection between these two

PDFs can be established using the change of variables formula, i.e.,

$$p_W(w) = p_Z(z)\left|\frac{\partial z}{\partial w}\right| = p_Z(h(w))|\det J_h(w)|, \tag{4.10}$$

where $\det J_h(w)$ is the determinant of the Jacobian matrix J_h of the encoding map $h(\cdot)$.

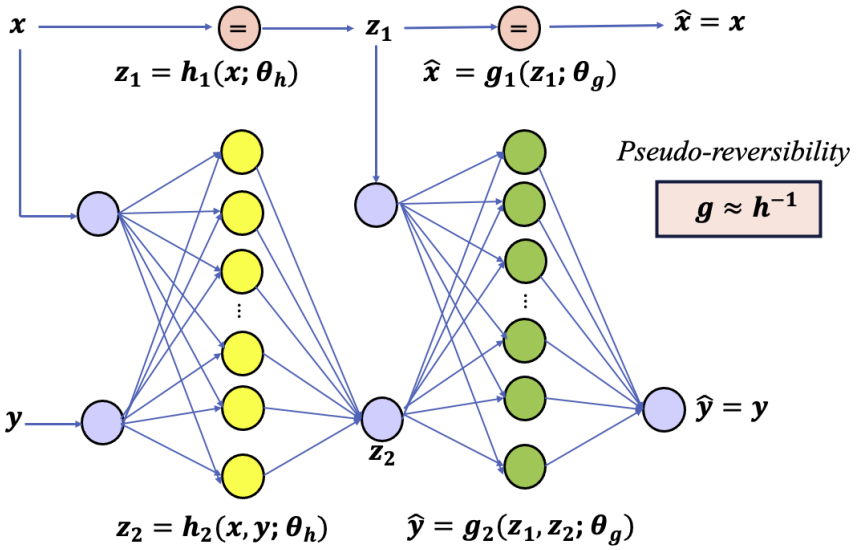

Figure 4.1: The architecture of the proposed conditional pseudo-reversible normalizing flow (PR-NF) model.

The architecture of the conditional PR-NF neural network is illustrated in Figure 4.1. Specifically, the transport map $h(\cdot)$ consists of two components: $h(x, y|x) = (h_1(x), h_2(x, y|x; \theta_h))$. Here, $h_2(x, y|x; \theta_h)$ represents a fully connected neural network with parameter θ_h, mapping from $\mathbb{R}^{d+s}$ to $\mathbb{R}^s$. On the other hand, $h_1(x)$ serves as an identity mapping, dependent on x. To describe the conditional relationship between x and y within the conditional distribution $p(y|x)$, both are included as input features in the neural network. This allows the neural network to learn the distribution of one input variable, y, conditioned on the other, x. We denote the output of $h_1(x)$ as z_1 and the output of $h_2(x, y|x; \theta_h)$ as z_2. The inverse map g exhibits a similar structure to the forward map, expressed as $g(z_1, z_2) = (g_1(z_1), g_2(z_1, z_2; \theta_g))$. Here, g_1 serves as an identity map, while g_2 is a fully connected neural network with parameter θ_g. The outputs of g_1 and g_2 are denoted as $\widehat{x}$

and $\widehat{y}$, respectively. It's worth noting that $\widehat{x} = z_1 = x$ because both h_1 and g_1 are identity mappings.

4.2.2 The Loss Function for Training the Conditional PR-NF

We can rewrite the training set in Eq. (4.6) using the simplified notation in Eq. (4.8), i.e.,

$$\mathcal{D}_{\text{train}} = \left\{w^{(n)}\right\}_{n=1}^{N} = \left\{x^{(n)}, y^{(n)}|x^{(n)}\right\}_{n=1}^{N}, \tag{4.11}$$

where $y^{(n)}|x^{(n)} = y^{(n)}$ because $y^{(n)}$ is obtained by pushing $x^{(n)}$ through the physical model in Eq. (4.1). The loss function consists of two components, i.e.,

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_2, \tag{4.12}$$

where $\mathcal{L}_1$ is the negative log-likelihood loss defined by

$$\mathcal{L}_1 = -\frac{1}{N}\sum_{n=1}^{N}\left(\log p_{Z_2}(h_2(w^{(n)};\theta_h)) + \log\left|\det J_h(w^{(n)};\theta_h)\right|\right), \tag{4.13}$$

with $p_{Z_2}(\cdot)$ is the probability density function of the standard normal distribution, and $\mathcal{L}_2$ is the pseudo-reversibility loss that measures the difference between w and $\widehat{w}$, i.e.,

$$\mathcal{L}_2 = \frac{1}{N}\sum_{n=1}^{N}\left(\left\|w^{(n)} - g(h(w^{(n)};\theta_h);\theta_g)\right\|_2^2 + \left|\det J_g(h(w^{(n)}))\det J_h(w^{(n)}) - 1\right|\right), \tag{4.14}$$

where the hyperparameter λ determines and balances the importance of the reversibility loss $\mathcal{L}_2$ in the total loss $\mathcal{L}$.

A process of hyperparameter tuning is required to achieve optimal model performance. To proceed, we conduct a grid search and employ cross-entropy as our metric to determine the optimal value for λ. To do this, we create a set of candidate values denoted as $\{\lambda_j\}_{j=1}^{J}$. For each λ_j, we will train a PR-NF model represented as $(h^{(j)}(w), g^{(j)}(z))$.

Subsequently, we will generate M samples of $\widehat{w}$ by applying the inverse mapping, i.e.,

$$\left\{\widehat{w}^{(j,m)} = g^{(j)}(z^{(m)}), m = 1, \ldots, M\right\}, \tag{4.15}$$

where $\{z^{(m)}\}_{m=1}^{M} = \{z_1^{(m)}, z_2^{(m)}\}_{m=1}^{M}$, with $z_1^{(m)}$ generated uniformly from $\mathcal{D}$ and $z_2^{(m)}$ sampled from the standard normal distribution. This approach allows us to generate a sufficient number of samples without additional training data. Subsequently, we construct a kernel density estimator using the samples outlined in Eq.(4.15), followed by computing the cross-entropy as follows:

$$H(\lambda_j) = -\frac{1}{N}\sum_{n=1}^{N} \mathsf{log}(p_{\mathsf{KDE}}(w_t^{(n)})), \tag{4.16}$$

where p_{KDE} represents the kernel density estimator, which is constructed based on the samples in Eq.(4.15), while $\{w_t^{(n)}\}_{n=1}^{N}$ is the training dataset in Eq.(4.11). The optimal hyperparameter λ is determined by minimizing the cross-entropy, formally expressed as $\lambda = \mathsf{argmin} H(\lambda_j)$. The numerical experiments in Sec 4.4 use the optimal hyperparameter λ that minimizes the cross-entropy.

4.2.3 The Computational Cost of Training the Conditional PR-NF Model

The computational challenges in training normalizing flow models are associated with calculation of the Jacobian determinant. In the conditional PR-NF model, the Jacobian determinant in $\mathcal{L}_1$ from Eq. (4.13) is expressed as:

$$|\mathsf{det} J_h(w)| = \left|\mathsf{det}\begin{pmatrix} \mathbf{I}_d, & 0 \\ \frac{\partial z_2}{\partial x}, & \frac{\partial z_2}{\partial y} \end{pmatrix}\right| = \left|\mathsf{det}\left(\frac{\partial z_2}{\partial y}\right)\right|, \tag{4.17}$$

where $\mathbf{I}_d$ is a $d \times d$ identity matrix. The simplification in Eq. (4.17) is based on the fact that h_1 functions as an identity map of x. As a result, the size of the Jacobian matrix J_h

depends only on the dimension of the output variable y. Because the PR-NF model does not have a special architecture (e.g., MAF[53] and Real NVPs[19]) that leads to explicit calculation of the Jacobian determinant, we employ the QR decomposition to compute the determinant, which exhibits a computational complexity of $\mathcal{O}(s^3)$. However, with the advancement of GPU acceleration techniques, the cost of QR decomposition becomes an acceptable approach for moderately high-dimensional problems.

Fig. 4.2 displays the computational cost (wall-clock time) of both PyTorch backpropagation and the calculation of the Jacobian determinant using 5,000 samples, comparing CUDA GPU and CPU versions. While we maintain a constant input parameter dimension of x at $d = 20$, we systematically vary the dimension of the observation y from $s = 10, 20, \ldots, 160$. The architecture employs a fully-connected neural network with one hidden layer and 512 hidden neurons. It's worth noting that as the dimension s increases, the GPU proves to be effective in accelerating the PR-NF model, particularly in the case of backpropagation, which benefits from the GPU acceleration techniques. On the other hand, we realize that the computational cost will become unaffordable when the dimension s is extremely high. In this scenario, dimension reduction methods, e.g., auto-encoders, can be used to identify low-dimensional manifold of the training dataset $\mathcal{D}_{\text{train}}$, and the conditional PR-NF is then applied in the latent space. We demonstrate this strategy in solving the geologic carbon storage problem in Section 4.4.3.

4.3 Convergence Analysis

In this section, we present an error analysis of the proposed PR-NF model introduced in Section 4.2. Here we limit our attention to the forward problem $Y = f(X) + \varepsilon(X)$[1]. We start by showing the convergence of the loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ in Lemma 4.1. Then,

[1]The observation Y is a function in term of X. In some places we use Y instead of $Y|X$ for notational simplicity.

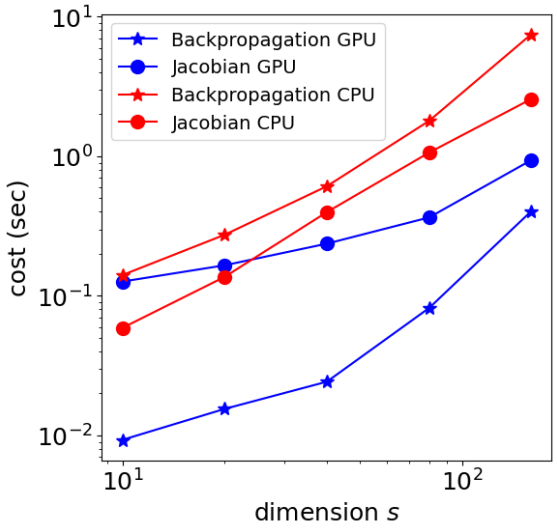

Figure 4.2: The computational cost (wall-clock time) of both PyTorch Backpropagation and the calculation of the Jacobian determinant with 5000 samples.

in Theorem 4.1, we establish the Monte Carlo integration error related to these loss functions. Finally, in Theorem 4.2, we show that with certain mild assumptions, the convergence of the loss function is sufficient to guarantee the convergence of the KL divergence between the ground truth distribution p_Y and its approximation $p_{\widehat{Y}}$ obtained through the PR-NF model.

4.3.1 Convergence of The Loss Function in Continuous Form

This section is to prove the convergence of loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ in Eq. (4.12). Equation 4.1 in Ref [52] shows that for any pair of well-behaved[2] distributions $p_W(w)$ and $p_Z(z)$, there exists a diffeomorphism $z = k(w) = (k_1(w), \ldots, k_{d+s}(w))$ that can transform $p_W(w)$ to $p_Z(z)$, i.e.,

$$p_W(w) = p_Z(k(w))|\det J_k(w)|. \tag{4.18}$$

In our analysis, we have the following Assumption 4.1 for the diffeomorphism k.

[2]A well-behaved distribution $p_X(x)$ means that $p_X(x) > 0$ for all $x \in \mathbb{R}^d$, and all conditional probabilities $\Pr(X_i \leq x_i | x_{<i})$ are differentiable, for $i = 1, 2, \ldots, d$.

Assumption 4.1. *We assume that the determinant of the Jacobian matrix and any partial derivative of $k \in C^1(\mathbb{R}^{d+s})$ are bounded, i.e.,*

$$\left|\frac{\partial k_i(w)}{\partial w_j}\right| < A_1 < \infty, \quad |\det J_k(w)| > A_2 > 0, \tag{4.19}$$

where A_1 and A_2 are positive constants and $i,\ j \in \{1, \ldots, d+s\}$. Additionally, the probability density function $p_W(w)$ is bounded, all the conditional probabilities are differentiable, and the decay of p_W at the tail is Gaussian like, i.e., there exists a positive constant K such that

$$p_W(w) < A_3 \exp(-\alpha \|w\|^2) \ \text{ for } \ \|w\| > K, \tag{4.20}$$

where $\alpha > 0$ and $A_3 > 0$ are positive constants and $\|\cdot\|$ denotes the l^2 norm.

The proposed PR-NF model can be viewed as an approximation of the diffeomorphism. Specifically, two networks $h(w)$ and $g(z)$ in Eq. (4.9) are approximations of k and k^{-1}, respectively. We first define auxiliary random variables

$$\widetilde{W} = h^{-1}(Z) \ \text{ and } \ \widehat{W} = g(Z), \tag{4.21}$$

where Z follows the standard normal distribution. Using the change of variables formula, the probability density function of $\widetilde{W}$ and $\widehat{W}$ are defined by

$$p_{\widetilde{W}}(w) = p_Z(h(w))|\det J_h(w)|, \quad p_{\widehat{W}}(w) = p_Z(g^{-1}(w))|\det J_{g^{-1}}(w)|. \tag{4.22}$$

For simplicity, we consider the loss functions in the continuous form, i.e.,

$$\begin{aligned} \widetilde{\mathcal{L}}_1 &= -\int_{\mathcal{D}_W} p_W(w)\left(\log p_{Z_2}(h_2(w)) + \log|\det J_h(w)|\right) dw \\ &= -\int_{\mathcal{D}_W} p_W(w)(\log p_{\widetilde{W}}(w) - \log p_{Z_1}(z_1))dw. \end{aligned} \tag{4.23}$$

and

$$\widetilde{\mathcal{L}}_2 = \int_{\mathcal{D}_W} p_W(w) \left(\|g(h(w)) - w\|^2 + |\det J_g(h(w)) \det J_h(w) - 1| \right) dw, \tag{4.24}$$

where $\mathcal{D}_W$ is the domain of variable W. For variables W and $\widetilde{W}$ we have the following Assumption 4.2.

Assumption 4.2. *We assume density functions $p_W(w)$ and $p_{\widetilde{W}}(w)$ satisfy*

$$A_4 \exp(-\alpha \|w\|^2) < \frac{p_W(w)}{p_{\widetilde{W}}(w)} < A_4 \exp(\alpha \|w\|^2), \tag{4.25}$$

where α and A_4 are positive constants.

Under the Assumptions 4.1 and 4.2. Applying Theorems 3.1 and 3.2 with the target random vector in Chapter 3 replaced by W, we have the following Lemma 4.1 for the convergence of the loss functions $\widetilde{\mathcal{L}}_1$ and $\widetilde{\mathcal{L}}_2$.

Lemma 4.1. *Under the Assumptions 4.1, 4.2. For an arbitrarily small $\varepsilon > 0$, there exist two independent single-hidden-layer neural networks h and g such that*

$$0 \leq \widetilde{\mathcal{L}}_1 + \int_{\mathcal{D}_W} p_W(w)(\log p_W(w) - \log p_{Z_1}(z_1))dw < \varepsilon, \text{ and } \quad \widetilde{\mathcal{L}}_2 < \varepsilon. \tag{4.26}$$

4.3.2 Monte Carlo Integration Error of Loss Function

In practice, we are not able to define the loss function in the continuous form since the neural network is data-based optimization solver. Hence, we employ the Monte Carlo method (discrete form) to approximate the loss functions $\widetilde{\mathcal{L}}_1$ and $\widetilde{\mathcal{L}}_2$. To proceed, we first introduce the following two lemmas.

Lemma 4.2. *Let $f \in L^1(\mathbb{R}^d, \mathbb{R})$ and dataset $\{x^{(i)}\}_{i=1}^n$ follow the probability density function $p_X(x)$. Then we have*

$$\mathbb{E}\left|\int_{\mathbb{R}^d} f(x)p_X(x)dx - \frac{1}{n}\sum_{i=1}^n f(x^{(i)})\right|^2 = \frac{1}{n}\mathsf{Var}(f(X)). \tag{4.27}$$

Proof. We define a family of independent and identically distributed (i.i.d.) variables $\{X^{(i)}\}_{i=1}^n$ that following distribution $p_{X^{(i)}}(x^{(i)})$. Then we have

$$\begin{aligned}
&\mathbb{E}\left|\int_{\mathbb{R}^d} f(x)p_X(x)dx - \frac{1}{n}\sum_{i=1}^n f(x^{(i)})\right|^2 \\
=&\mathbb{E}\left|\sum_{i=1}^n \frac{\mathbb{E}[f(X^{(i)})] - f(X^{(i)})}{n}\right|^2 \\
=&\frac{1}{n^2}\left[\sum_{i=1}^n \mathsf{Var}(f(X^{(i)})) + \sum_{i\neq j=1}^n \mathsf{Cov}(f(X^{(i)}), f(X^{(j)}))\right] = \frac{\mathsf{Var}(f(X))}{n}.
\end{aligned} \tag{4.28}$$

We complete the proof. □

Lemma 4.3. *Let $f \in L^2(\mathbb{R}^d, \mathbb{R})$ be a Lipschitz continuous function with constant K, i.e. $|f(x) - f(y)| \leq K\|x - y\|$. For a random variable $X = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^d$, we have the following inequality*

$$\mathsf{Var}(f(X)) \leq K^2 \sum_{i=1}^d \mathsf{Var}(X_i). \tag{4.29}$$

Proof. Let Y be an independent and identically distributed (i.i.d.) as X. Since $f(X)$ and $f(Y)$ are independent and have the same variance, we have

$$\begin{aligned} 2\mathsf{Var}(f(X)) &= \mathsf{Var}(f(X)) + \mathsf{Var}(f(Y)) \\ &= \mathsf{Var}(f(X) - f(Y)) = \mathbb{E}\left[|f(X) - f(Y)|^2\right] \\ &\leq K^2 \mathbb{E}\left[\|X - Y\|^2\right] \quad \text{by Lipschitz continuity} \\ &= K^2 \sum_{i=1}^{d} \mathbb{E}\left[|X_i - Y_i|^2\right] = K^2 \sum_{i=1}^{d} \mathbb{E}\left[(X_i)^2 - 2X_iY_i + (Y_i)^2\right] \\ &= K^2 \sum_{i=1}^{d} \left(2\mathbb{E}[(X_i)^2] - 2(\mathbb{E}[X_i])^2\right) = 2K^2 \sum_{i=1}^{d} \mathsf{Var}(X_i). \end{aligned} \tag{4.30}$$

We complete the proof. □

Based on the results of Lemmas 4.2 and 4.3, we have the error estimate of the loss functions $\widetilde{\mathcal{L}}_1$ and $\widetilde{\mathcal{L}}_2$.

Theorem 4.1. *Loss functions $\mathcal{L}_1$ in Eq.* (4.13) *and $\mathcal{L}_2$ in Eq.* (4.14) *are the discrete form of $\widetilde{\mathcal{L}}_1$ and $\widetilde{\mathcal{L}}_2$, respectively. For the forward problem $Y = f(X) + \varepsilon(X)$, $X \in \mathbb{R}^d$ and $Y \in \mathbb{R}^s$ with N training samples, we have*

$$\max\left\{\mathbb{E}|\widetilde{\mathcal{L}}_1 - \mathcal{L}_1|^2, \mathbb{E}|\widetilde{\mathcal{L}}_2 - \mathcal{L}_2|^2\right\} \leq \frac{C}{N}\left[\sum_{i=1}^{s}\left[\mathbb{E}[v_{ii}(X)] + \mathsf{Var}(f_i(X))\right] + \sum_{i=1}^{d}\mathsf{Var}(X_i)\right], \tag{4.31}$$

where constant C depends on the Lipschitz constant K in Lemma 4.3, and $v(x) = \mathsf{Var}[\varepsilon(x)]$.

Proof. According to Lemma 4.2 and Lemma 4.3, we have

$$\begin{aligned} \mathbb{E}|\widetilde{\mathcal{L}}_1 - \mathcal{L}_1|^2 \leq & \frac{C_1}{N}\sum_{i=1}^{d+s}\mathsf{Var}(W_i) = \frac{C_1}{N}\left[\sum_{i=1}^{d}\mathsf{Var}(X_i) + \sum_{i=1}^{s}\mathsf{Var}(Y_i)\right] \\ = & \frac{C_1}{N}\left[\sum_{i=1}^{d}\mathsf{Var}(X_i) + \sum_{i=1}^{s}\mathbb{E}[Y_i^2] - \sum_{i=1}^{s}\mathbb{E}[Y_i]^2\right] := \frac{C_1}{N}\left(I_1 + I_2 + I_3\right). \end{aligned} \tag{4.32}$$

For the term I_2, we have

$$\begin{aligned} I_2 &= \sum_{i=1}^{s} \int_{\mathcal{D}_{Y_i}} y_i^2 P(Y_i = y_i) dy_i \\ &= \sum_{i=1}^{s} \int_{\mathcal{D}_{Y_i}} \int_{\mathcal{D}_X} y_i^2 P(Y_i = y_i | X = x) P(X = x) dx dy_i \\ &= \int_{\mathcal{D}_X} P(X = x) \int_{\mathcal{D}_{Y_i}} \sum_{i=1}^{s} y_i^2 P(Y_i = y_i | X = x) dy_i dx \\ &= \int_{\mathcal{D}_X} P(X = x) \sum_{i=1}^{s} [v_{ii}^2(x) + f_i^2(x)] dx = \sum_{i=1}^{s} \mathbb{E}\left[v_{ii}(X) + f_i^2(X)\right]. \end{aligned} \tag{4.33}$$

where $\mathcal{D}_X$ and $\mathcal{D}_Y$ are domains of variables X and Y. For the term I_3, we have

$$\begin{aligned} I_3 &= \sum_{i=1}^{s} \left(\int_{\mathcal{D}_{Y_i}} y_i P(Y_i = y_i) dy_i \right)^2 \\ &= \sum_{i=1}^{s} \left(\int_{\mathcal{D}_{Y_i}} \int_{\mathcal{D}_X} y_i P(Y_i = y_i | X = x) P(X = x) dx dy_i \right)^2 \\ &= \sum_{i=1}^{s} \left(\int_{\mathcal{D}_X} P(X = x) \int_{\mathcal{D}_{Y_i}} y_i P(Y_i = y_i | X = x) dy_i dx \right)^2 \\ &= \sum_{i=1}^{s} \left(\int_{\mathcal{D}_X} P(X = x) f_i(x) dx \right)^2 = \sum_{i=1}^{s} \left(\mathbb{E}\left[f_i(X)\right] \right)^2. \end{aligned} \tag{4.34}$$

Hence, the estimate for loss function $\mathcal{L}_1$ in Eq. (4.31) is proved by combining inequalities in Eqs. (4.32), (4.33) and (4.34). Similarly, we establish the estimates for $\mathcal{L}_2$. We complete the proof. □

4.3.3 Convergence of the KL Divergence

The section is to show that the convergence of $\widetilde{\mathcal{L}}_1$ and $\widetilde{\mathcal{L}}_2$ with certain assumptions implies the convergence of the KL-divergence between the conditional distribution $p_{Y|X}$ and $p_{\widehat{Y}|X}$.

Assumption 4.3. *We assume that the target random variable $W = (X, Y|X)$ has the finite second-order moment, and there exists a positive constant A such that*

$$\|\nabla p_{\widehat{W}}(w)\| \leq A(\|w\| + 1)p_{\widehat{W}}(w), \tag{4.35}$$

where $p_{\widehat{W}}(w)$ is the distribution of the approximation $\widehat{W} = (X, \widehat{Y}|X)$.

Theorem 4.2. *Under the assumptions of Lemma 4.1, Theorem 4.1 and Assumption 4.3, for any given $\varepsilon > 0$, with sufficient training samples, there exist two neural networks h and g such that*

$$\mathbb{E}\left(\int_{\mathcal{D}_X} D_{\mathsf{KL}}(p_{Y|X}\|p_{\widehat{Y}|X})dx\right) < \varepsilon. \tag{4.36}$$

Proof. In the training process, we assume the input variable X follows a uniform distribution, i.e., $P_X(x) = \frac{1}{|\mathcal{D}_X|}$ for all $x \in \mathcal{D}_X$. By the definition of KL divergence we have

$$\begin{aligned}
&\int_{\mathcal{D}_X} D_{\mathsf{KL}}(p_{Y|X}\|p_{\widehat{Y}|X})dx \\
=&|\mathcal{D}_X|\int_{\mathcal{D}_X} P_X(x)\left(\int_{\mathcal{D}_Y} P_{Y|X}(y|x)\log\frac{P_{Y|X}(y|x)}{P_{\widehat{Y}|X}(y|x)}dy\right)dx \\
=&|\mathcal{D}_X|\int_{\mathcal{D}_X}\left(\int_{\mathcal{D}_Y} P_X(x)P_{Y|X}(y|x)\log\frac{P_{Y|X}(y|x)P_X(x)}{P_{\widehat{Y}|X}(y|x)P_X(x)}dy\right)dx \\
=&|\mathcal{D}_X|\int_{\mathcal{D}_W} P_W(w)\log\frac{P_W(w)}{P_{\widehat{W}}(w)}dw = |\mathcal{D}_X|D_{\mathsf{KL}}(p_W\|p_{\widehat{W}}).
\end{aligned} \tag{4.37}$$

Following the Theorem 3.3 in chapter 3, we have

$$D_{\mathsf{KL}}(p_W\|p_{\widehat{W}}) < \widetilde{\mathcal{L}}_1 + \int_{\mathcal{D}_W} p_W(w)(\log p_W(w) - \log p_{Z_1}(z_1))dw + A(\widetilde{\mathcal{L}}_2 + \widetilde{\mathcal{L}}_2^{\frac{1}{2}}), \tag{4.38}$$

where A is the constant in Assumption 4.3.

We choose ε_1 and ε_2 such that $\varepsilon_1 < \varepsilon/(2|\mathcal{D}_X|)$ and $\varepsilon_2 < \min\{\varepsilon^2/(16A^2|\mathcal{D}_X|^2), 1\}$. Let

$$\mathcal{L}_1 < -\int_{\mathbb{R}^d} p_W(w)(\log p_W(w) - \log p_{Z_1}(z_1))dw + \varepsilon_1/2,$$

and $\mathcal{L}_2 < \varepsilon_2/2$. By Theorem 4.1 we can choose an integer n large enough such that

$$\max\{\mathbb{E}|\widetilde{\mathcal{L}}_1 - \mathcal{L}_1|^2, \mathbb{E}|\widetilde{\mathcal{L}}_2 - \mathcal{L}_2|^2\} < \min\{\varepsilon_1^2/4, \varepsilon_2^2/4\},$$

then we have

$$\begin{aligned}
&\mathbb{E}\left|\widetilde{\mathcal{L}}_1 + \int_{\mathbb{R}^d} p_W(w)(\log p_W(w) - \log p_{Z_1}(z_1))dw\right| \\
<&\mathbb{E}\left|\mathcal{L}_1 + \int_{\mathbb{R}^d} p_W(w)(\log p_W(w) - \log p_{Z_1}(z_1))dw\right| + \mathbb{E}|\widetilde{\mathcal{L}}_1 - \mathcal{L}_1| \\
<&\frac{\varepsilon_1}{2} + (\mathbb{E}|\widetilde{\mathcal{L}}_1 - \mathcal{L}_1|^2)^{\frac{1}{2}} = \frac{\varepsilon_1}{2} + \sqrt{\frac{\varepsilon_1^2}{4}} = \varepsilon_1,
\end{aligned} \tag{4.39}$$

and

$$\mathbb{E}(A(\widetilde{\mathcal{L}}_2 + \widetilde{\mathcal{L}}_2^{\frac{1}{2}})) = A(\mathbb{E}(\widetilde{\mathcal{L}}_2) + \mathbb{E}(\widetilde{\mathcal{L}}_2^{\frac{1}{2}})) \leq A(\mathbb{E}(\widetilde{\mathcal{L}}_2) + (\mathbb{E}(\widetilde{\mathcal{L}}_2)^{\frac{1}{2}}) < A(\varepsilon_2 + \varepsilon_2^{\frac{1}{2}}). \tag{4.40}$$

The last inequality holds due to

$$\mathbb{E}(\widetilde{\mathcal{L}}_2) \leq \mathbb{E}(\mathcal{L}_2) + \mathbb{E}|\widetilde{\mathcal{L}}_2 - \mathcal{L}_2| \leq \mathbb{E}(\mathcal{L}_2) + (\mathbb{E}|\widetilde{\mathcal{L}}_2 - \mathcal{L}_2|^2)^{\frac{1}{2}} < \frac{\varepsilon_2}{2} + \frac{\varepsilon_2}{2} = \varepsilon_2. \tag{4.41}$$

Thus, by (4.37)-(4.40) we have

$$\begin{aligned}
\mathbb{E}\left(\int_{\mathcal{D}_X} D_{\mathsf{KL}}(p_{Y|X}\|p_{\widehat{Y}|X})dx\right) =&|\mathcal{D}_X|\mathbb{E}\left(D_{\mathsf{KL}}(p_W\|p_{\widehat{W}})\right) \\
<&|\mathcal{D}_X|\left(\varepsilon_1 + A(\varepsilon_2 + \varepsilon_2^{\frac{1}{2}})\right) \leq |\mathcal{D}_X|\left(\varepsilon_1 + A(2\varepsilon_2^{\frac{1}{2}})\right) \\
<&|\mathcal{D}_X|\left(\frac{\varepsilon}{2|\mathcal{D}_X|} + 2A\left(\frac{\varepsilon^2}{16A^2|\mathcal{D}_X|^2}\right)^{\frac{1}{2}}\right) = \varepsilon.
\end{aligned} \tag{4.42}$$

We complete the proof. □

4.4 Numerical Examples

In this section we present numerical experiments to demonstrate the performance of the proposed conditional PR-NF model. The synthetic example in Sec. 4.4.1 benchmarks the accuracy of our model for problems with known analytical ground-truth solutions. In Sec. 4.4.2 we present the implementation of the PR-NF model on high-dimensional uncertainty problems. Sec. 4.4.3 presents an application to a real-world geologic carbon storage problem. In all numerical simulations, the PyTorch machine learning framework has been implemented with CUDA GPU. The source code is publicly available at `https://github.com/mlmathphy/PRNF_uncertainty`. The numerical results presented in this section can be accurately replicated by utilizing the code available on our GitHub repository.

4.4.1 Verification of Algorithm Accuracy

We consider the following uncertainty propagation problem

$$y = f(x) + \varepsilon(x),\ x \in \mathcal{D} = [0, 1], \tag{4.43}$$

where the deterministic function $f(x)$ is considered as two expressions:

$$\textbf{Quadratic} : f(x) = 4(x - 0.5)^2 \text{ or } \textbf{Sin} : f(x) = \sin(2\pi x). \tag{4.44}$$

The uncertainty $\varepsilon(x)$ has four options (homoscedastic and heteroscedastic):

$$\begin{aligned} &\textbf{Gaussian} : \varepsilon(x) \sim \mathcal{N}(0, 0.15) \text{ or } \varepsilon(x) \sim \mathcal{N}(0, 0.2|f(x)|). \\ &\textbf{Laplace} : \varepsilon(x) \sim \text{Laplace}(0, 0.1) \text{ or } \varepsilon(x) \sim \text{Laplace}(0, 0.15|f(x)|). \end{aligned} \tag{4.45}$$

The training dataset is based on 20K samples of variable $\{x^{(i)}\}_{i=1}^{N_{\text{train}}}$ uniformly generated in $\mathcal{D}$. For each $x^{(i)}$, $i = 1, \ldots, N_{\text{train}}$, we generated the corresponding observation $y^{(i)}$ using

the equation $y^{(i)} = f(x^{(i)}) + \varepsilon(x^{(i)})$. The pair $\{x^{(i)}, y^{(i)}\}$, $i = 1, \ldots, N_{\text{train}}$, consists of the training dataset.

In the context of the forward problem, x serves as the input, and y represents the output. Conversely, in the inverse problem, the roles of x and y are reversed. We utilize the same training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{N_{\text{train}}}$ and build two independent single hidden layer neural networks for forward and inverse problems, respectively. In the training process, we choose hyperparameter λ, which controls the relative importance of the pseudo-reversivility and the negative log-likelihood losses, such that the cross entropy $H(\lambda)$ in Eq. (4.16) has a minimum. In both forward and inverse problems we choose $\lambda = 80$. The single hidden layer neural network has 256 neurons and uses Tanh activation function. The neural networks are trained by the Adam optimizer with 2000 epochs.

The PR-NF Model is Capable of Sampling and Evaluating the Conditional Distribution $p(y|x)$

We use the forward uncertainty propagation (x serves as the input, and y represents the output) as an example. This section is to demonstrate that the well-trained PR-NF model can serve as a surrogate model to

- generate sufficient samples of observation variable y for any input variable $x \in \mathcal{D}$;
- evaluate the conditional distribution $p(y|x)$ based on samples of y.

The Kullback–Leibler (KL) divergence is applied as a metric to test the accuracy of models.

Fig. 4.3 shows the Kullback–Leibler divergence between the ground-truth $p(y|x)$ and the approximation $p(\widehat{y}|x)$ at $x \in (-1, 2)$ formulated as

$$D_{\text{KL}}(x) = \int_{\mathbb{R}^s} p(y|x) \log\left(\frac{p(y|x)}{p(\widehat{y}|x)}\right) dy, \tag{4.46}$$

where the density $p(\widehat{y}|x)$ is calculated based on 20K samples $\{\widehat{y}^{(i)}\}_{i=1}^{N_{\text{sample}}}$ generated from the well-trained PR-NF model and the KL-divergence is numerically approximated by Riemann sum over a uniform mesh. It is shown that the PR-NF model achieves a good agreement in KL-divergence when the input x is in the training dataset domain, i.e., $x \in \mathcal{D}$. However, the KL-divergence gets worse when $x \notin \mathcal{D}$, which means the PR-NF model does not have the prediction property for adapting beyond its initial training domain.

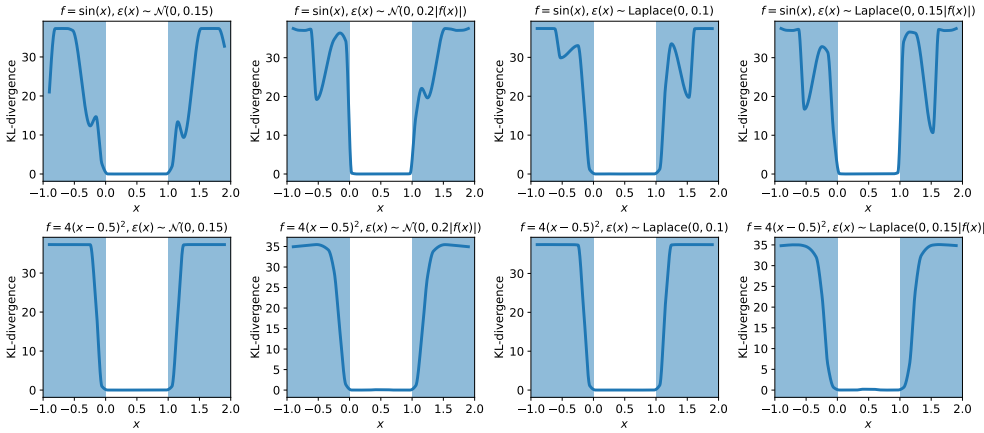

Figure 4.3: The Kullback–Leibler (KL) divergence between the ground-truth density $p(y|x)$ and the approximation $p(\widehat{y}|x)$ from the PR-NF model for any $x \in (-1, 2)$.

Building upon the conclusion from Fig. 4.3, we investigate further into the performance of the PR-NF model regarding the evaluation of the density function $p(y|x)$ at specific values of x. Figs. 4.4 and 4.5 show the case of $f(x) = \sin(2\pi x)$ and $f(x) = 4(x-0.5)^2$, respectively. Each row represents different additive noise and each column corresponds to different test point x. In alignment with the findings depicted in Fig. 4.3, we select four test points, $x = -0.8, 0.2, 0.8, 1.8$, with two points falling outside the domain $\mathcal{D}$ and two points inside. In each plot, the orange curve is the ground-truth $p(y|x)$ and the blue histogram is the approximation $p(\widehat{y}|x)$ generated by the PR-NF model with $N_{\text{sample}} = 20$K evaluation samples. Very good agreements are observed for inside test point $x \in \mathcal{D}$ (2nd & 3rd columns) and the PR-NF model captures different types of uncertainties including the heteroscedastic noises, but it does not work for test points outside of domain $\mathcal{D}$ (1st & 4th columns). The center of distribution function $p(y|x)$ locates at $f(x)$ and the shape of

distribution is determined by the noise $\varepsilon(x)$. Here we highlight that the well-trained PR-NF model can evaluate $p(y|x)$ at any $x \in \mathcal{D}$ without any additional training processes.

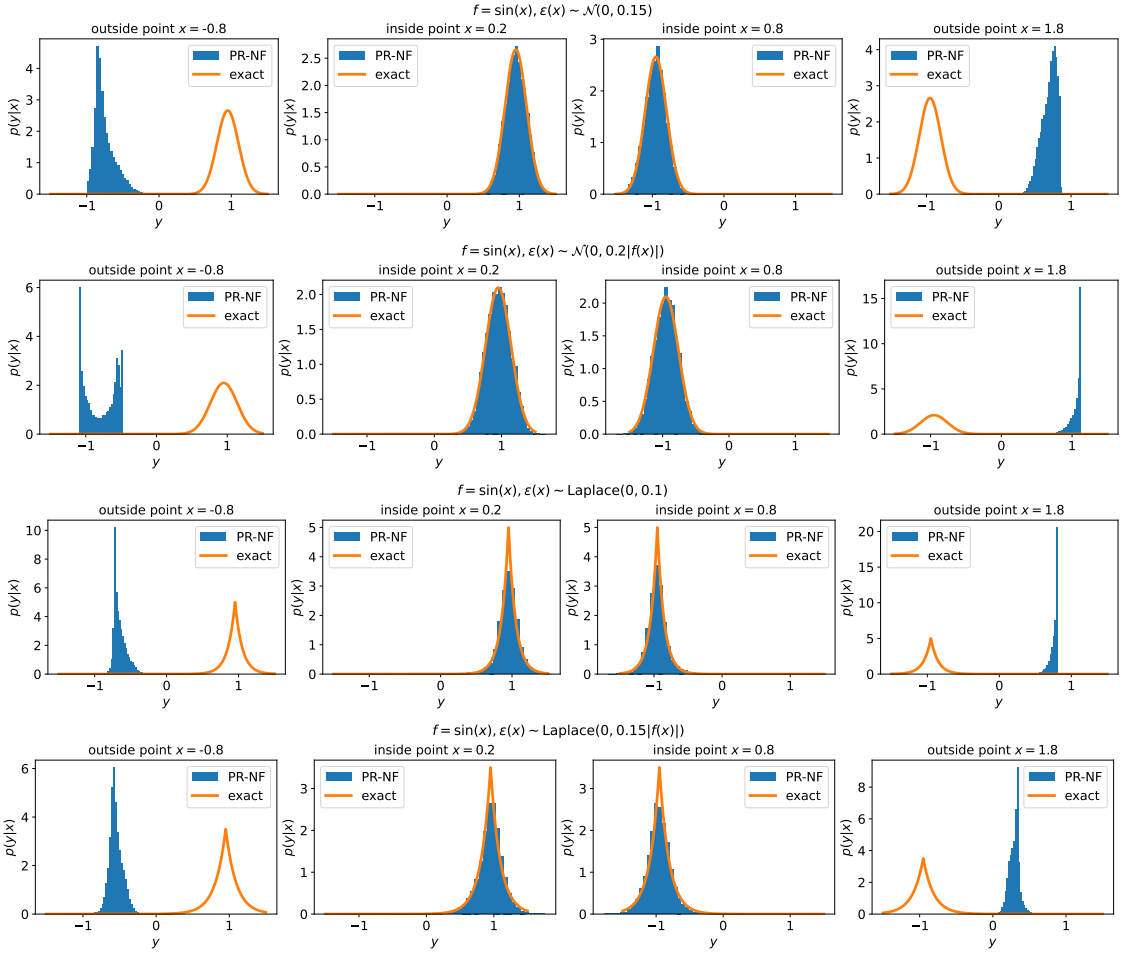

Figure 4.4: The accuracy performance of the well-trained PR-NF model on evaluating $f(x) = \sin 2\pi x$ with four different additive noises.

The PR-NF Model Can Evaluate the Bimodal Conditional Distribution $p(x|y)$

Fig. 4.6 illustrates the training data of two physical models: $f(x) = \sin 2\pi x$ (left panel) and $f(x) = 4(x - 0.5)^2$ (right panel), each subject to uncertainty $\varepsilon \sim \mathcal{N}(0, 0.15)$. It is observed that for each observation y, the corresponding parameter variable x exhibits a probability distribution characterized by two distinct peaks (bimodal distribution). Due to the lack of a one-to-one relationship from variable y to x, we are not able to derive a function of x

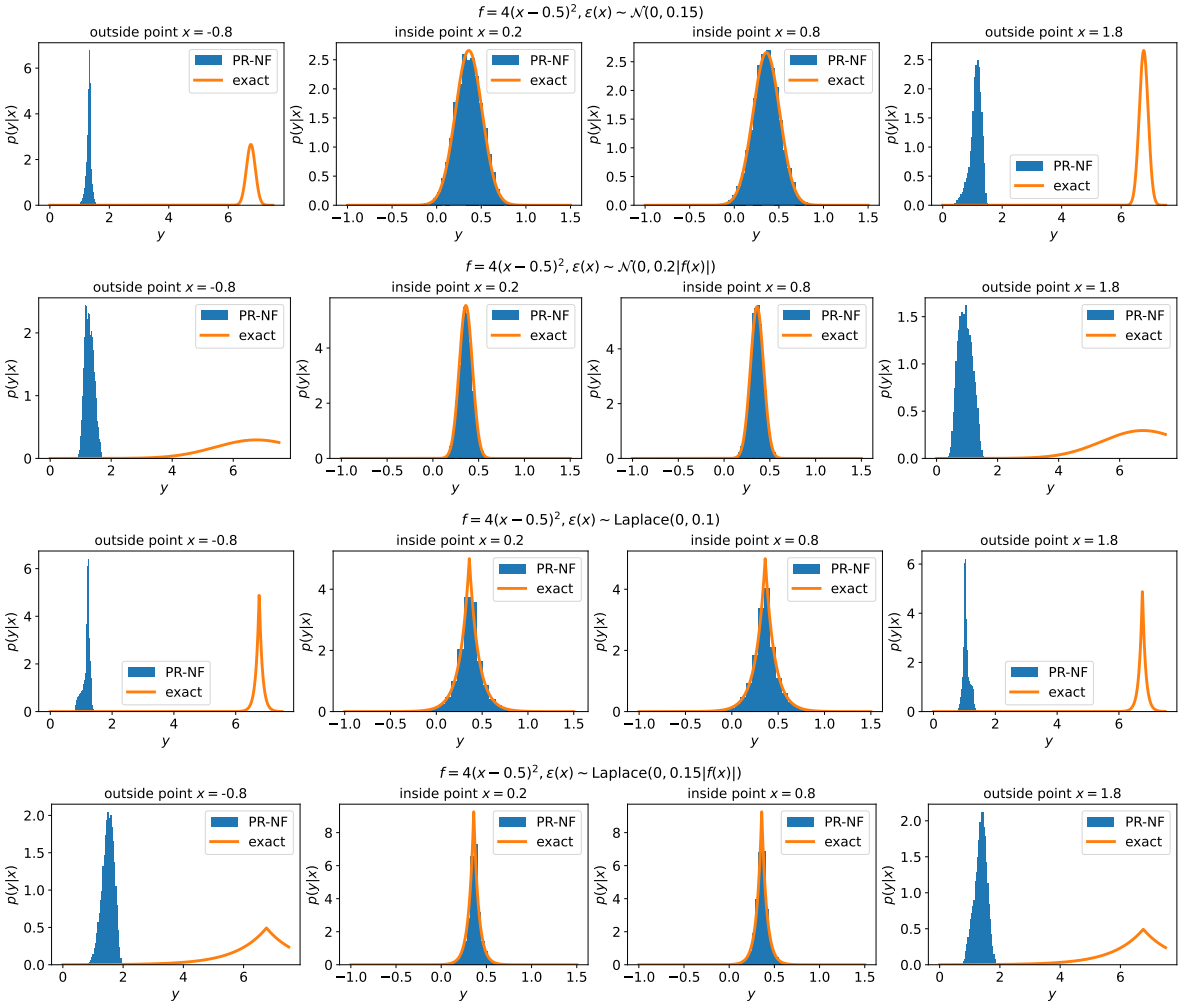

Figure 4.5: The accuracy performance of the well-trained PR-NF model on evaluating $f(x) = 4(x - 0.5)^2$ with four different additive noises.

in terms of y. Traditional neural network models often struggle to effectively capture the process from y to x since mean squared error(MSE) fails to capture two peaks.

Fig. 4.7 shows the conditional distribution $p(x|y)$ constructed by the well-trained PR-NF model at three test points $y = -0.5, 0, 0.5$, where $f(x) = \sin 2\pi x$ and $\varepsilon \sim \mathcal{N}(0, 0.15)$. Even though given the explicit expression of function f and uncertainty ε, we are not able to derive the analytical expression of $p(x|y)$. The ground truth PDF is constructed using Gaussian kernel density estimation with sufficient samples. Very good agreements on the distribution are observed. Similarly, Fig. 4.8 shows the conditional distribution $p(x|y)$ at test points $y = 0.25, 0.5, 0.75$, where $f(x) = 4(x - 0.5)^2$ and $\varepsilon \sim \mathcal{N}(0, 0.15)$. It is observed

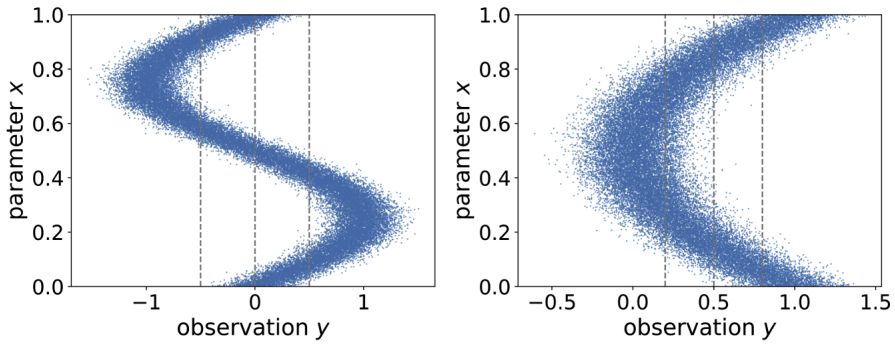

Figure 4.6: The training data $\{x^{(i)}, y^{(i)}\}_{i=1}^{N_{\text{train}}}$ of functions $f(x) = \sin(2\pi x)$ and $f(x) = 4(x - 0.5)^2$, each subject to $\varepsilon \sim \mathcal{N}(0, 0.15)$.

that the PR-NF model is not sensitive to the input-output function relationship and has the capacity to capture the bimodal distribution. It utilizes the identical training data and similar neural network configuration (with a simple input and output position switch) for both forward and inverse uncertainty problems.

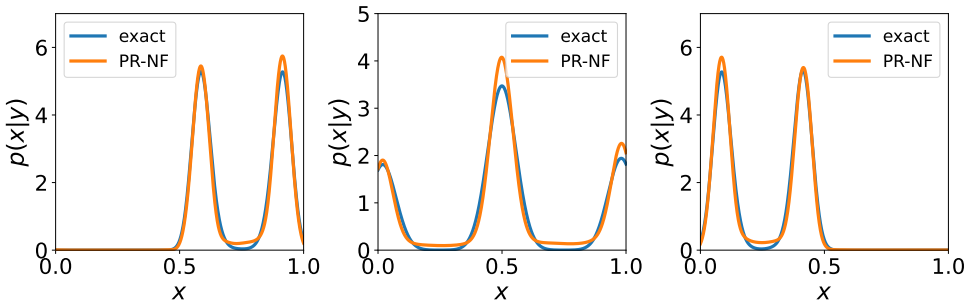

Figure 4.7: The PR-NF model evaluates the conditional distribution $p(x|y)$ at three points $y = -0.5, 0, 0.5$, where $f(x) = \sin(2\pi x)$ and $\varepsilon \sim \mathcal{N}(0, 0.15)$.

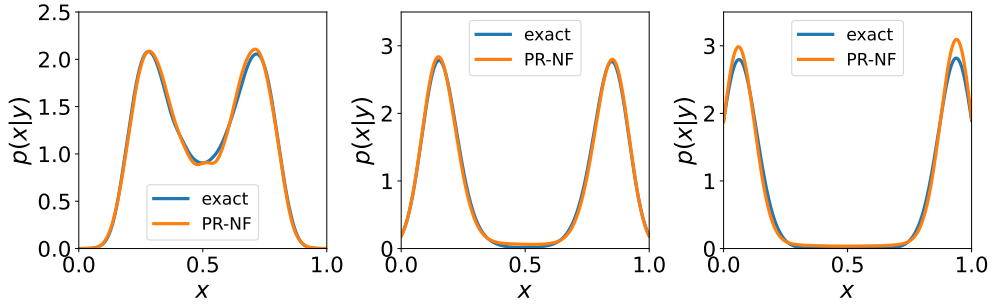

Figure 4.8: The PR-NF model evaluates the conditional distribution $p(x|y)$ at three points $y = 0.25, 0.5, 0.75$, where $f(x) = 4(x - 0.5)^2$ and $\varepsilon \sim \mathcal{N}(0, 0.15)$.

4.4.2 High-Dimensional Regression Problem

This numerical example aims to illustrate how the PR-NF model effectively quantifies uncertainty in high-dimensional problems. We consider a physical model that has the parameter variable $x \in \mathcal{D} = [0,1]^d$ with $d = 20$ and the observation variable $y \in \mathbb{R}^s$ with $s = 5, 10, 20$, as follows

$$y = f(x) + \varepsilon(x), \tag{4.47}$$

where the deterministic function $f(x) = \mathbf{A}x$ and values of the coefficient matrix $\mathbf{A} \in \mathbb{R}^{s\times d}$ are selected from the uniform distribution $\mathcal{U}[0,1]$. The additive noise ε has three options:

- A multivariate normal distribution without correlation i.e., $\varepsilon(x) \sim \mathcal{N}(0, 0.1 \cdot \mathbf{I}_s)$.
- A multivariate Gaussian mixture distribution that consists of two same weight Gaussian distribution components, $\varepsilon_1(x) \sim \mathcal{N}(0.1, 0.1 \cdot \mathbf{I}_s)$ and $\varepsilon_2(x) \sim \mathcal{N}(-0.1, 0.1 \cdot \mathbf{I}_s)$.
- A multivariate normal distribution with correlation, i.e., $\varepsilon(x) \sim \mathcal{N}(0, \Sigma_{s\times s})$, where the covariance matrix $\Sigma_{s\times s}$ is a symmetric positive definite matrix and its values are randomly selected from $\mathcal{N}(0,1)$.

In this problem the training dataset $\mathcal{V}$ consists of $N_{\text{train}} = 30$K samples. The hyperparameter λ is chosen as $\lambda = 80$ for which the cross entropy, $H(\lambda)$, has a minimum. We use a fully-connected neural network with one hidden layer to build a surrogate model for the conditional distribution $P(y|x)$. We evaluate the PR-NF model by the following metrics:

- Error of the normalized mean value:

$$\text{Err}_{\text{mean}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\|f(x^{(i)}) - \widehat{f}(x^{(i)})\|_2}{\|f(x^{(i)})\|_2}, \tag{4.48}$$

  where $f(x^{(i)})$ is computed as the mean of $y(x^{(i)})$.

- Error of the standard deviation, and covariance matrix that for the third additive noise (the correlated normal distribution):

$$\mathsf{Err}_{\mathsf{std}} = \frac{1}{N_{\mathsf{test}}} \sum_{i=1}^{N_{\mathsf{test}}} \frac{\|\sigma(y(x^{(i)})) - \sigma(\widehat{y}(x^{(i)}))\|_2}{\sqrt{s}}, \quad \mathsf{Err}_{\mathsf{cov}} = \frac{1}{N_{\mathsf{test}}} \sum_{i=1}^{N_{\mathsf{test}}} \frac{\|\Sigma(y(x^{(i)})) - \Sigma(\widehat{y}(x^{(i)}))\|_F}{s}. \quad (4.49)$$

- Average of the KL divergence:

$$\mathsf{Avg}_{\mathsf{KL}} = \frac{1}{N_{\mathsf{test}}} \sum_{i=1}^{N_{\mathsf{test}}} D_{\mathsf{KL}}(p(x^{(i)}) \,\|\, \widehat{p}(x^{(i)})), \quad (4.50)$$

  where the KL divergence is defined as the relative entropy from the approximate density (generated from PR-NF) p_{approx} to the exact density p_{exact}, i.e.,

$$D_{\mathsf{KL}}(p(x^{(i)}) \,\|\, \widehat{p}(x^{(i)})) = \int_{\mathbb{R}^s} p(y|x^{(i)}) \log \left(\frac{p(y|x^{(i)})}{p(\widehat{y}|x^{(i)})} \right) dy, \quad (4.51)$$

  where D_{KL} is approximated by the Riemann sum over a uniform mesh.

These average metrics are averaged among $N_{\mathsf{test}} = 100$ test points, which are randomly sampled in domain $\mathcal{D}$. For each test point $x^{(i)}$, we generate $N_{\mathsf{sample}} = 20\mathsf{K}$ samples of $y(x^{(i)})$ for the evaluation. $\|\cdot\|_2$ and $\|\cdot\|_F$ are L^2 and Frobenius norm among dimensionality, respectively.

To assess the robustness of the PR-NF model, we conducted ten (without repeats) random simulations, each varying the hyperparameter λ and the number of neurons N_{neuron} from the set

$$\{(\lambda, N_{\mathsf{neuron}}) | \lambda \in \{1, 50, 100, 200\}, N_{\mathsf{neuron}} \in \{400, 600, 800, 1000\}\}.$$

As the PR-NF employs a single hidden layer neural network, the number of hidden layers is not a hyperparameter and remains fixed at one. Figures 4.9, 4.10, and 4.11 depict the decay of training loss alongside the above defined average metrics of testing dataset,

each presented with a 95% confidence interval. Each figure represents a different dimensional case ($s = 5, 10, 20$), with each row corresponding to different types of additive noise. The decay trends observed in the figures exhibit strong performance, particularly noteworthy is stable in error decay following a short transition period, evident by the narrow confidence intervals. These numerical findings underscore the capability of the proposed PR-NF model in addressing various high-dimensional uncertainty challenges. The PR-NF model consistently converges towards the minimum loss within a reasonable parameter range, demonstrating its robustness and stability. Regarding computational efficiency, with the GPU-acceleration, even for the case $s = 20$ the training process takes about 10 minutes and the evaluation is completed in seconds.

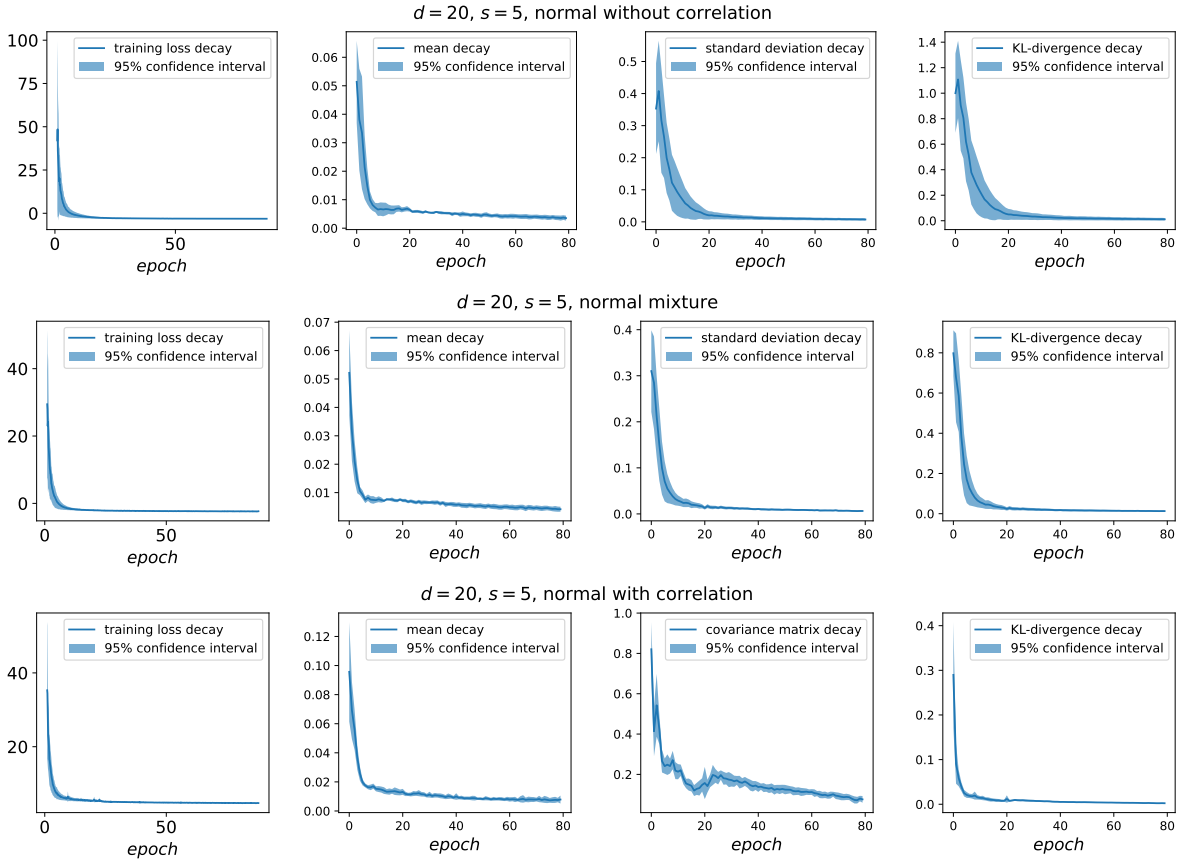

Figure 4.9: The decay of training loss and average metrics with $d = 20$, $s = 5$.

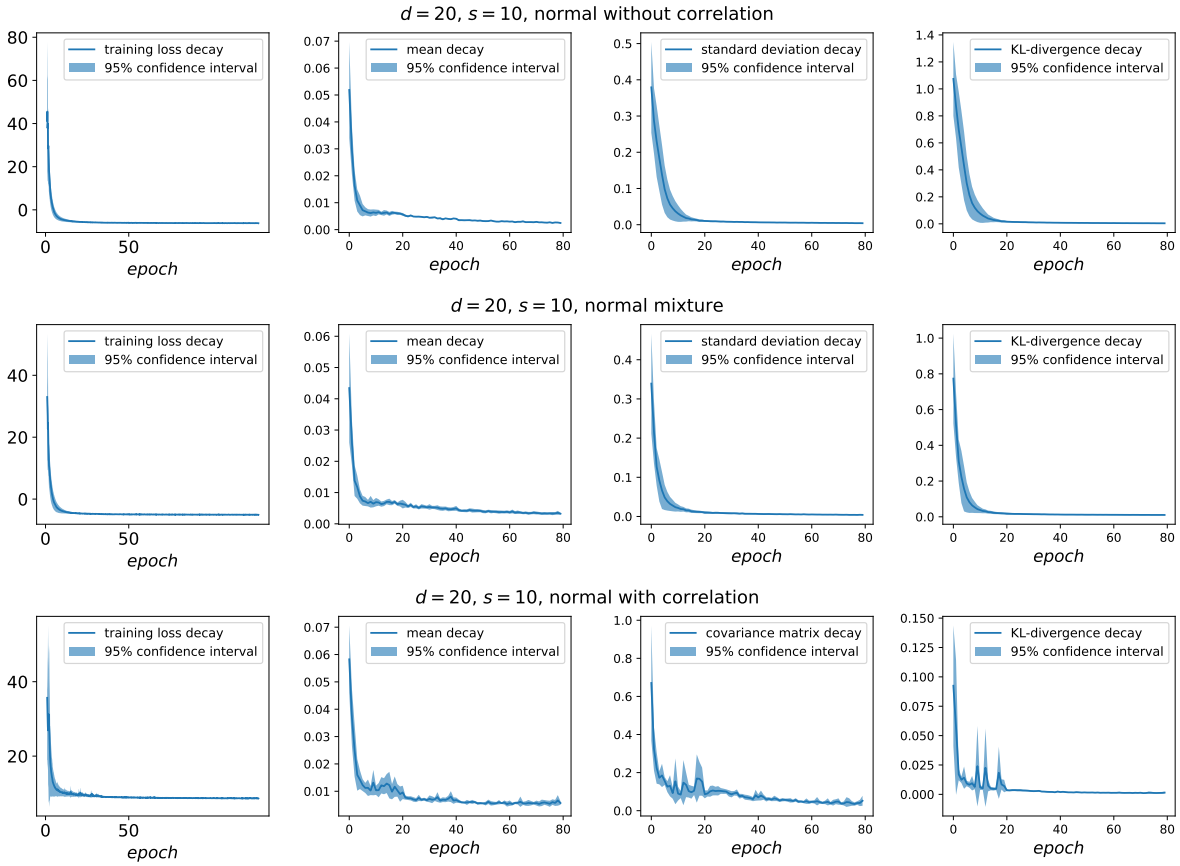

Figure 4.10: The decay of training loss and average metrics with $d = 20$, $s = 10$.

The optimal numerical results within the parameter set are recorded in Table 4.1. As expected, the PR-NF model effectively evaluates the distribution of the ground truth across various types of additive noise. Notably, even with increasing dimensionality, the PR-NF model consistently maintains a high level of performance in approximation.

4.4.3 Application to a Geologic Carbon Storage Problem

In addressing the greenhouse effect caused by trillions of tons of CO_2 in the atmosphere since the industrial age, it becomes evident that merely reducing emissions from the industrial, power, and transportation sectors is insufficient to combat climate change. Therefore, alongside deploying clean energy technologies for a decarbonized future, there is a critical need for carbon dioxide removal approaches [2]. One promising option is the

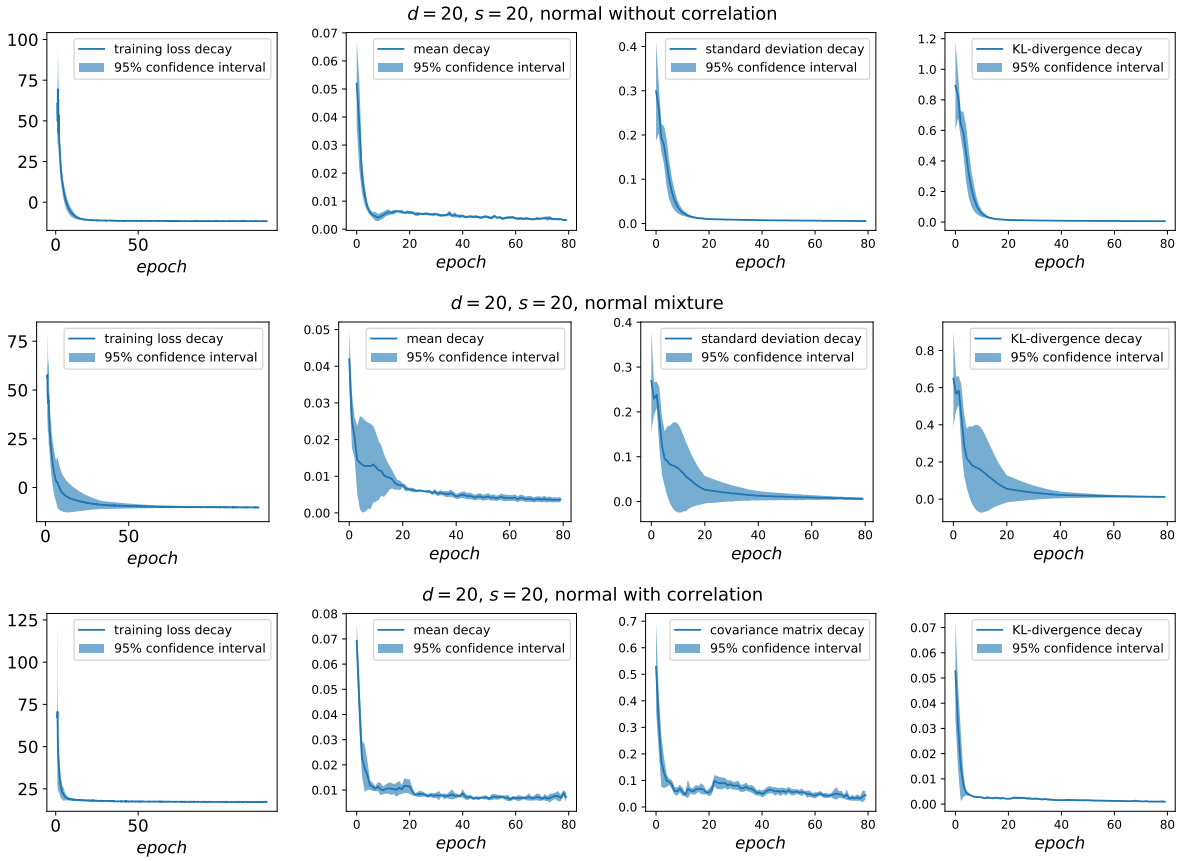

Figure 4.11: The decay of training loss and average metrics with $d = 20$, $s = 20$.

sequestration of CO_2 in geological reservoirs beneath the Earth's surface. This methodology involves capturing CO_2 emissions originating from stationary anthropogenic sources, which encompass fossil-fueled power plants and various industrial processes, followed by secure storage in diverse geologic formations, which may comprise saline-bearing formations, depleted oil and gas reservoirs, unmineable coal seams, and organic-rich shales [25].

In subsurface storage, CO_2 is injected in its supercritical phase to depths where temperature and pressure conditions maintain the CO_2 in this phase, thereby optimizing storage volume utilization within reservoir pore spaces [29]. To ensure the safe and effective deployment of large-scale geologic carbon storage, a comprehensive risk management strategy is required to minimize and mitigate potential risks during CO_2 injection phases. A key aspect of this risk management approach is the continuous monitoring

s	Noise distribution	Err_{mean}	Err_{std}	Err_{cov}	Avg_{KL}
5	Normal without correlation	4.31e-03	9.78e-03	N/A	2.51e-02
5	Normal mixture	4.25e-03	9.00e-03	N/A	1.15e-02
5	Normal with correlation	9.00e-03	N/A	4.51e-02	1.49e-03
10	Normal without correlation	2.36e-03	6.55e-03	N/A	5.06e-03
10	Normal mixture	2.75e-03	4.75e-03	N/A	3.98e-03
10	Normal with correlation	6.80e-03	N/A	3.93e-02	1.67e-03
20	Normal without correlation	2.42e-03	3.06e-03	N/A	7.04e-04
20	Normal mixture	2.76e-03	7.46e-03	N/A	1.34e-03
20	Normal with correlation	7.99e-03	N/A	3.26e-02	8.79e-04

Table 4.1: Optimal numerical results for $d = 20$ derived from the simulations of Figs. 4.9, 4.10, and 4.11. For correlated normal noise, Err_{cov} is reported in place of Err_{std}.

of pressure fields over time to secure the containment of CO_2 within the reservoir [24]. For example, pressure buildup can pose a risk of leakage, especially when it reaches the fracture initiation pressure, potentially inducing leakage paths in caprock or faulting seals. These paths may serve as conduits for CO_2 migration from the designated storage location, potentially contaminating underground drinking water resources or escaping into the atmosphere [12]. Conventionally, obtaining a reliable estimation of 3D pressure distributions over time in the subsurface relies on the inverse modeling process. Firstly, a forward reservoir model is developed and then calibrated using observational data through inverse modeling. Subsequently, this calibrated model is utilized to predict pressure distribution. However, this approach for real-time forecasting encounters challenges due to the computational expense of inverse modeling and the unreliability of predictions stemming from limited observations.

To generate the training dataset, we developed a physics-based reservoir simulation model representing a fluvial depositional environment, specifically the Gulf of Mexico High Island 24L, to simulate the migration of CO_2 in subsurface formations, as illustrated in Figure 4.12a. The geological model is structured with grid cells $54 \times 48 \times 54$ along the x, y, and z axes, with cell dimensions of 750 feet in the horizontal plane and 10 feet vertically. Four vertical injection wells are strategically positioned to attain cumulative CO_2 storage. Additionally, three production wells are positioned to prevent reservoir

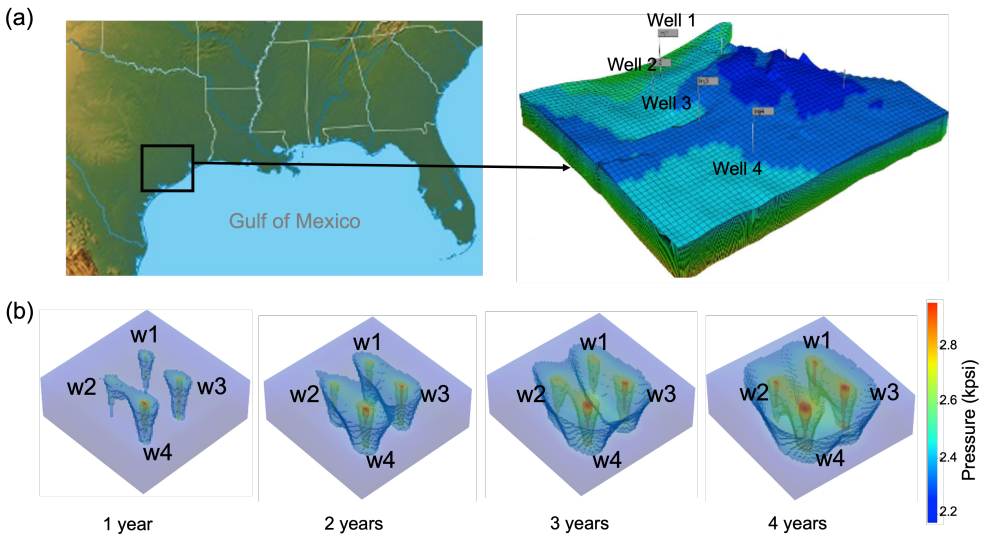

Figure 4.12: Workflow for forecasting three-dimensional pressure fields during a 30-year CO_2 injection process in the Gulf of Mexico High Island 24L reservoir using sparse observations from four injection wells and physics-based reservoir simulations.

over-pressurization, maintaining a constant bottom-hole pressure constraint. We utilized CMG-GEM for reservoir simulations [17], generating 108 simulations collected every two months over 30 years, each consisting of 180 time steps. The input parameters include geological model realizations, cumulative CO_2 injection targets, and injection allocations for well pairs, while the output comprises pressure distributions.

In this study, we employ the developed PR-NF method to directly forecast the 3D pressure fields over time using observation data obtained from the injection wells, thereby avoiding the need for computationally demanding inverse modeling, as shown in Figure 4.12b. To implement this approach, we initially extract observation data from four injection wells, including every other layer between 33 and 51, resulting in a total of 40 observation variables. Subsequently, we employ a dimension reduction technique by utilizing a convolutional autoencoder to map the high-dimensional 3D pressure fields into a lower-dimensional latent space with 20 dimensions. Next, we forecast the 3D pressure fields in the latent space at the time step $t+1$ based on the observations at time step t. Finally, the predicted latent variables are transformed back to their original 3D space.

Figure 4.13 illustrates the forecasted pressure fields across layers 20, 30, and 40 at the end of the injection process for a sample in the testing set. The black triangles represent the four injection well locations. The first column represents the ground truths obtained through reservoir simulations, the second column illustrates the predictions generated by the developed PR-NF method, and the final column describes the error between these two sets of data. The comparative analysis between the synthetic truths and prediction results highlights minimal disparities in the pressure distribution. This observation demonstrates the PR-NF method's adeptness in capturing the spatial evolution characteristics of pressure fields, affirming its reliability as an effective monitoring tool for CO_2 storage operations.

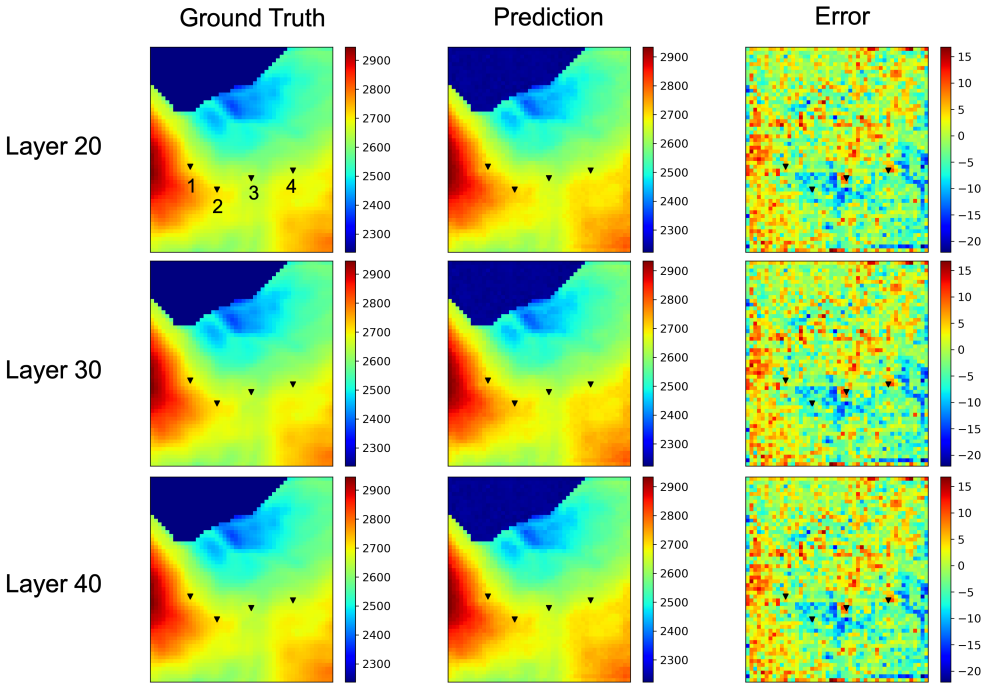

Figure 4.13: Predictions of pressure fields across three layers at the termination of the injection process.

To further explore the predictive capabilities of the PR-NF method, we analyze the temporal pressure distribution at four injection wells. Figure 4.14 presents a comparison between the predicted pressure and the actual pressure values over the injection process,

using the same testing sample illustrated in Figure 4.13. The remarkable consistency observed in the pressure predictions at each well location confirms the PR-NF method's ability to accurately forecast temporal pressure distribution during CO_2 injection. Notably, our proposed method not only enhances forecasting accuracy but also significantly expedites the forecasting process. For instance, the CMG reservoir simulator necessitates approximately 4.5 hours to complete a simulation, but the well-trained PR-NF model requires minutes to forecast 3D pressure fields. This accelerated forecasting capability represents a valuable enhancement to conventional forecasting workflow.

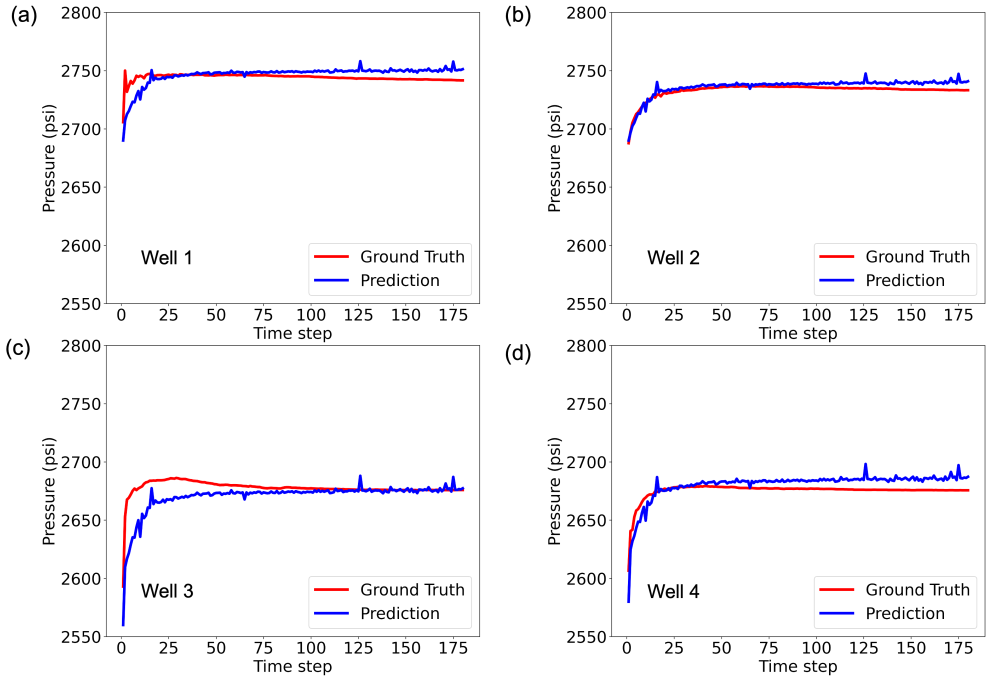

Figure 4.14: Predictions of pressure over time at four injection wells for layer 30.

Chapter 5 Error Estimates of a Training-Free Diffusion Model for High-Dimensional Sampling

5.1 Problem Setting

We aim to estimate the numerical approximation error of a training-free diffusion model for supervised learning of a generative model for producing unlimited samples from a d-dimensional random variable, defined by

$$X \in \mathbb{R}^d \text{ and } X \sim p_X(x), \tag{5.1}$$

where p denotes the probability density function of the target random variable X. We are given a training dataset, denoted by

$$\mathcal{D}_{\text{train}} = \{x_1, x_2, \ldots, x_J\} \subset \mathbb{R}^d, \tag{5.2}$$

which contains J independent samples from the target distribution $p_X(x)$. From the computational perspective, the objective is to construct a parametric generative model of the following form

$$X = F(Y;\theta) \text{ with } Y \in \mathbb{R}^d, \tag{5.3}$$

which is a transport map that transforms the reference random variable Y (e.g., a standard Gaussian random variable) into the target random variable X. Once the parameter θ is determined through model training, unlimited samples of X can be produced by first sampling from Y and then passing those samples through the trained generative model in Eq. (5.3).

A key challenge in training the generative model $F(y;\theta)$ arises from the absence of labeled training data, which means there is no corresponding sample of Y for each sample $x_j \in \mathcal{D}_{\text{train}}$ in Eq. (5.2). Consequently, the model $F(y;\theta)$ cannot be trained using

supervised learning based on a mean square error (MSE) loss. Instead, an indirect loss derived from the change-of-variables formula, i.e., $p_X(x) = p_Y(F^{-1}(x))|\det \mathsf{D}(F^{-1}(x))|$, is commonly used to train normalizing flow models in an unsupervised manner. The limitation of the unsupervised learning approach is that it requires specially designed reversible architectures for $F(y;\theta)$ to enable efficient backpropagation through the computation of $|\det \mathsf{D}(F^{-1}(x))|$. To address this challenge, our previous work [48] introduced a training-free diffusion model to generate labeled data such that the generative model $F(y;\theta)$ can be trained in a supervised manner using the MSE loss, which eliminates the need for having a reversible architecture for $F(y;\theta)$ solely for training purposes.

5.1.1 Overview of the Training-Free Diffusion Model

We briefly review the algorithm to be analyzed in this paper, namely the training-free score-based diffusion model proposed in [48], which generates labeled data for the supervised learning of the generative model in Eq. (5.3). Our algorithm is based on the score-based diffusion model, which comprises a forward process and a reverse process. The forward process maps the target random variable X to a standard Gaussian variable $Y \sim \mathcal{N}(0, \mathbf{I}_d)$ through the following SDE:

$$dZ_t = b(t)Z_t dt + \sigma(t)dW_t, \quad Z_0 = X,\ Z_1 = Y,\ t \in [0,1], \tag{5.4}$$

where W_t is a standard Brownian motion. Although there are many choices for the drift and diffusion coefficients $b(t)$ and $\sigma(t)$ in Eq. (5.4), we focus on analyzing the diffusion model using the following definition of the coefficients, i.e.,

$$b(t) = \frac{d\log \alpha_t}{dt}, \quad \sigma^2(t) = \frac{d\beta_t^2}{dt} - 2\frac{d\log \alpha_t}{dt}\beta_t^2, \tag{5.5}$$

where the processes α_t and β_t are defined by

$$\alpha_t = 1 - t, \quad \beta_t^2 = t, \quad t \in [0, 1]. \tag{5.6}$$

The reverse SDE for generating new samples of the target distribution is defined by

$$dZ_t = \left[b(t)Z_t - \sigma^2(t)S(Z_t, t)\right] dt + \sigma(t)d\overleftarrow{W}_t, \quad Z_0 = X,\ Z_1 = Y, \tag{5.7}$$

where $\overleftarrow{W}_t$ is a reverse Brownian motion, $b(t)$ and $\sigma(t)$ adopt the same definition as in Eq. (5.5), and $S(Z_t, t)$ is the score function defined by

$$S(z_t, t) := \nabla_z \log q_{Z_t}(z_t), \tag{5.8}$$

with $q_{Z_t}(z_t)$ denoting the marginal probability density function of Z_t at time t in Eq. (5.4). When the score function in Eq. (5.8) is known or can be approximated, we can generate samples of X by solving the reverse SDE in Eq. (5.7) starting from a sample of $Y \sim \mathcal{N}(0, \mathbf{I}_d)$ as the terminal condition. Therefore, the central task in using the diffusion model is to approximate the score function.

The training-free score estimator. Traditional diffusion models rely on training neural networks to approximate the score function, for example via score matching [36]. In contrast, we leverage the analytical structure of the forward process to derive a training-free score estimator. Substituting $q_{Z_t}(z_t) = \int_{\mathbb{R}^d} q_{Z_t|Z_0}(z_t|z_0)q_{Z_0}(z_0)dz_0$ into Eq. (5.8) and using the fact that

$$q_{Z_t|Z_0}(z_t|z_0) = \phi(z_t; \alpha_t z_0, \beta_t^2 \mathbf{I}_d), \tag{5.9}$$

with $\phi(z_t; \alpha_t z_0, \beta_t^2 \mathbf{I}_d)$ being the probability density function of the Gaussian distribution $\mathcal{N}(\alpha_t z_0, \beta_t^2 \mathbf{I}_d)$, we obtain

$$S(z_t, t) = \int_{\mathbb{R}^d} \left(-\frac{z_t - \alpha_t z_0}{\beta_t^2}\right) w_t(z_t, z_0) q_{Z_0}(z_0) dz_0, \tag{5.10}$$

where the weight function is defined by

$$w_t(z_t, z_0) := \frac{q_{Z_t|Z_0}(z_t|z_0)}{\int_{\mathbb{R}^d} q_{Z_t|Z_0'}(z_t|z_0')q_{Z_0}(z_0')dz_0'}. \tag{5.11}$$

Given the dataset $\mathcal{D}_{\text{train}}$ in Eq. (5.2) containing samples from $q_{Z_0}(z_0) := p_X(x)$, we approximate the integral in Eq. (5.10) using the following Monte Carlo estimation:

$$S(z_t, t) \approx \bar{S}(z_t, t) := \sum_{j=1}^{J} \left(-\frac{z_t - \alpha_t x_j}{\beta_t^2} \right) \bar{w}_t(z_t, x_j), \tag{5.12}$$

where

$$\bar{w}_t(z_t, x_j) := \frac{q_{Z_t|Z_0}(z_t|x_j)}{\sum_{j'=1}^{J} q_{Z_t|Z_0}(z_t|x_{j'})}, \tag{5.13}$$

with $q_{Z_t|Z_0}(z_t|x_j)$ given by the Gaussian density function in Eq. (5.9). With the score estimation in Eq. (5.12), we can directly solve the reverse SDE in Eq. (5.7) to generate new samples of the target distribution $p_X(x)$. Nevertheless, the stochastic nature of the reverse process in Eq. (5.7) cannot be used to generate labeled data for supervised learning of the map in Eq. (5.3), which will be addressed in the next subsection.

5.1.2 Supervised Learning of the Generative Model

We describe how to use the score representation given in Section 5.1.1 to generate labeled data and use such data to train the generative model in Eq. (5.3). Starting from the forward SDE in Eq. (5.4), let $q_{Z_t}(z)$ denote the marginal probability density of Z_t and write the score function as in Eq. (5.8), i.e., $S(z, t) := \nabla_z \log q_{Z_t}(z)$. The density function q_{Z_t} satisfies the Fokker–Planck equation:

$$\partial_t q_{Z_t}(z) = -\nabla_z \cdot \big(b(t)\, z\, q_{Z_t}(z)\big) + \frac{1}{2}\sigma^2(t)\Delta_z q_{Z_t}(z). \tag{5.14}$$

Using $\Delta_z q_{Z_t} = \nabla_z \cdot \big(\nabla_z q_{Z_t}\big) = \nabla_z \cdot \big(S(z,t)\, q_{Z_t}(z)\big)$, we can rewrite Eq. (5.14) in the flux form, i.e.,

$$\partial_t q_{Z_t}(z) = -\nabla_z \cdot \left(\left[b(t)\, z - \frac{1}{2}\, \sigma^2(t)\, S(z,t) \right]\, q_{Z_t}(z) \right). \tag{5.15}$$

Any deterministic flow $\dot{z}(t) = v(z(t),t)$ with the same marginals satisfies the continuity equation:

$$\partial_t q_{Z_t}(z) \;=\; -\nabla_z \cdot \big(v(z,t)\, q_{Z_t}(z)\big). \tag{5.16}$$

Matching Eq. (5.15) and Eq. (5.16) yields the following form of the velocity term:

$$v(z,t) = b(t)\, z - \frac{1}{2}\, \sigma^2(t)\, S(z,t),$$

and the probability-flow ODE driven by the velocity term is introduced as

$$dz_t = \left[b(t) z_t - \frac{1}{2} \sigma^2(t) S(z_t,t) \right] dt, \tag{5.17}$$

whose trajectories have the same marginal distribution as the reverse SDE in Eq. (5.7). The reverse ODE defines a much smoother functional relationship between the initial state and the terminal state than that defined by the reverse SDE in Eq. (5.7), which can be used to generate the desired labeled data.

To generate labeled data to train the generator in Eq. (5.3), we first draw M samples of Y, denoted by $\mathcal{Y} = \{y_1, \ldots, y_M\}$ from the standard normal distribution. For $m = 1, \ldots, M$, we solve the reverse ODE in Eq. (5.17) and collect the state $Z_0|y_m$, where the score function is computed using Eq. (5.12) and Eq. (5.13), and the dataset $\mathcal{D}_{\text{train}}$ in Eq. (5.2). The labeled training dataset is denoted by

$$\mathcal{D}_{\text{label}} := \{(y_m, x_m) : x_m = Z_0|y_m \text{ for } m = 1, \ldots, M\}, \tag{5.18}$$

where x_m is obtained by solving the reverse ODE in Eq. (5.17). After obtaining $\mathcal{D}_{\text{label}}$, we can use it to train the generative model $F(y;\theta)$ in Eq. (5.3) using supervised learning with the MSE loss. It has been numerically demonstrated that the supervised learning approach for training the generative models have shown promising potential in several uncertainty quantification tasks [76, 47, 23, 49]. However, there is no error estimate to show the convergence rate of the discretization scheme for the training-free diffusion model, specifically for the reverse ODE in Eq. (5.17), which motivated us to conduct the error analysis presented in Section 5.2.

5.2 Error Estimates of the Training-Free Diffusion Model

The algorithm described in Section 5.1.2 indicates that the accuracy of the generator $F(y;\theta)$ in Eq. (5.3), trained via supervised learning, is strongly influenced by the quality of the labeled data $\mathcal{D}_{\text{label}}$ in Eq. (5.18), which is obtained by solving the reverse ODE in Eq. (5.17). The main contribution of this paper is to provide error estimates for the discretization of the reverse ODE. Our error analysis seeks to answer the following central questions:

- *How does the error from the score estimation in Eq.* (5.12) *accumulate through the time-stepping scheme and affect the final convergence rate?*
- *Does the singularity of the drift and diffusion coefficients* $b(t)$ *and* $\sigma(t)$ *defined in Eq.* (5.5) *affects the accuracy and stability in solving the reverse ODE?*
- *How does the discretization error of the reverse ODE depend on the dimensionality* d *of the target random variable* $X \in \mathbb{R}^d$ *in Eq.* (5.1)*?*

To address the above questions, we first introduce an important assumption used throughout the rest of the error analysis.

Assumption 5.1. *Given the training dataset $\mathcal{D}_{\text{train}}$ in Eq.* (5.2)*, we assume that the probability density function of the state Z_0 in the reverse ODE in Eq.* (5.17) *at $t = 0$ is a Gaussian mixture model defined by*

$$q_{Z_0}(z_0) = \sum_{j=1}^{J} P_\xi(j)\, p_{Z_0|\xi}(z_0|j) = \frac{1}{J} \sum_{j=1}^{J} \phi(z_0; z_0^j, \Sigma), \tag{5.19}$$

where ξ is a discrete random variable taking values in $\{1, \ldots, J\}$, $P_\xi(j) := 1/J$ is the probability of the event $\xi = j$, z_0^j denotes the j-th sample from the training dataset $\mathcal{D}_{\text{train}}$, and $p_{Z_0|\xi}(z_0|j) := \phi(z_0; z_0^j, \Sigma)$ represents the probability density function of a Gaussian distribution with mean z_0^j and covariance matrix Σ.

The Gaussian mixture model in Eq. (5.19) can be interpreted as an "inflated" version of the empirical distribution represented by the samples in the training dataset $\mathcal{D}_{\text{train}}$. Assumption 5.1 is usually referred to as the *inflation strategy*, which is widely used in ensemble-based data assimilation methods [21, 22]. The inflation approach serves two complementary purposes in uncertainty quantification. First, it acts as a regularization mechanism that alleviates the underestimation of uncertainty commonly encountered in high-dimensional systems with limited ensemble size. By artificially broadening the ensemble spread, inflation mitigates variance collapse and promotes better representation of the true uncertainty. Second, from a generative AI perspective, inflation helps alleviate model memorization by encouraging diversity in the learned representations. By introducing controlled stochasticity or variance inflation into the training dataset, we can reduce overfitting to training samples and enhances generalization, enabling more reliable synthesis of unseen states or scenarios.

Based on Assumption 5.1, we can split the total discretization error of the reverse ODE in Eq. (5.17) into the following three parts:

$$\begin{aligned} & z_0^{\text{true}}(z_1) - F(z_1;\theta) \\ &= \underbrace{z_0^{\text{true}}(z_1) - z_0(z_1)}_{\text{Data error}} + \underbrace{z_0(z_1) - z^K(z_1)}_{\substack{\text{Reverse ODE} \\ \text{discretization error}}} + \underbrace{z^K(z_1) - F(z_1;\theta)}_{\text{Supervised learning error}}, \end{aligned} \tag{5.20}$$

where $z_0^{\text{true}}(z_1)$ is ground truth solution of the reverse ODE in Eq. (5.17) with $S(z_t, t)$ being the score function in Eq. (5.10) defined by the target density function $p_X(x)$ in Eq. (5.1), $F(z_1;\theta)$ is the predicted solution of the generative model in Eq. (5.3) trained by the supervised learning approach discussed in Section 5.1.2; $z_0(z_1)$ is the solution of the reverse ODE in Eq. (5.17) with $S(z_t, t)$ being the score function in Eq. (5.10) defined by the Gaussian mixture model in Eq. (5.19), $z^K(z_1)$ is the approximation of $z_0(z_1)$ using the numerical discretization scheme introduced in Section 5.2.1 with K discrete time steps.

We observe that data error in Eq.(5.20) is solely determined by Assumption 5.1 and the size of the training set $\mathcal{D}_{\text{train}}$ and the supervised learning error is solely determined by classic function approximation theory. Importantly, neither of these error components depend on the diffusion model itself. Consequently, in order to address the three questions posed at the beginning of Section 5.2 concerning the training-free diffusion model, we focus the remainder of Section 5.2 on analyzing the reverse ODE discretization error. On the other hand, the error decomposition in Eq. (5.20) plays a crucial role in achieving only mild dependence of the resulting error bounds on the dimensionality, and in rendering the bounds fully numerically verifiable.

5.2.1 Discretization of the Reverse ODE

We describe the discretization of the reverse ODE in Eq. (5.17) and show that the singularity in the drift and diffusion coefficients can be removed once the score function is

incorporated, so it poses no numerical difficulty. Substituting the expressions for $b(t)$ and $\sigma(t)$ from Eq. (5.5) into Eq. (5.17), we obtain

$$dz_t = \left[\frac{1}{t-1}z_t - \frac{1+t}{2(1-t)}S(z_t,t)\right]dt, \tag{5.21}$$

where both the drift and diffusion coefficients are singular as $t \to 1$. Since the solutions to the reverse ODE in Eq. (5.21) and the forward SDE in Eq. (5.4) share the same marginal distributions, we have

$$Z_t = (1-t)Z_0 + \sqrt{t}\varepsilon_0, \tag{5.22}$$

where the random variable $\varepsilon_0 \sim \mathcal{N}(0, \mathbf{I}_d)$ is independent of Z_0 and the latent variable ξ in Eq. (5.19). The variable Z_t, conditionally on ξ, is expressed as

$$Z_t|\xi = (1-t)Z_0|\xi + \sqrt{t}\varepsilon_0. \tag{5.23}$$

Then we can write the random vector $(Z_0|\xi, Z_t|\xi)^T$ jointly as

$$\begin{pmatrix} Z_0|\xi \\ Z_t|\xi \end{pmatrix} = \begin{pmatrix} \mathbf{I}_d & 0 \\ (1-t)\mathbf{I}_d & \sqrt{t}\mathbf{I}_d \end{pmatrix} \begin{pmatrix} Z_0|\xi \\ \varepsilon_0 \end{pmatrix}. \tag{5.24}$$

Combining Eq. (5.19) and Eq. (5.24), we have

$$\begin{pmatrix} Z_0|\xi = j \\ Z_t|\xi = j \end{pmatrix} \sim \begin{pmatrix} \mathbf{I}_d & 0 \\ (1-t)\mathbf{I}_d & \sqrt{t}\mathbf{I}_d \end{pmatrix} \mathcal{N}\left(\begin{pmatrix} z_0^j \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma & 0 \\ 0 & \mathbf{I}_d \end{pmatrix}\right), \tag{5.25}$$

from which we have $Z_t|(\xi = j) \sim \mathcal{N}((1-t)z_0^j, (1-t)^2\Sigma + t\mathbf{I}_d)$. The marginal probability density of Z_t is the equally weighted mixture defined by

$$\begin{aligned} p_{Z_t}(z_t) = & \sum_{j=1}^{J} P_{Z_t|\xi}(z_t \mid j) P_\xi(j) \\ = & \frac{1}{J} \sum_{j=1}^{J} \phi\left(z_t; (1-t)z_0^j, (1-t)^2\Sigma + t\mathbf{I}_d\right), \end{aligned} \tag{5.26}$$

where $\phi(z_t; (1-t)z_0^j, (1-t)^2\Sigma + t\mathbf{I}_d)$ denotes the probability density function of a Gaussian distribution with mean $(1-t)z_0^j$ and covariance matrix $(1-t)^2\Sigma + t\mathbf{I}_d$.

According to Eq. (5.26), we can derive the exact score function S in Eq. (5.8) as

$$\begin{aligned} S(z_t, t) = & -\left((1-t)^2\Sigma + t\mathbf{I}_d\right)^{-1} z_t \\ & + (1-t)\left((1-t)^2\Sigma + t\mathbf{I}_d\right)^{-1} \sum_{j=1}^{J} z_0^j \, w_j(z_t, t). \end{aligned} \tag{5.27}$$

For each component index $j = 1, \ldots, J$, we define a simplified notation

$$e_j(z_t, t) := \exp\Big(-\tfrac{1}{2}\big(z_t - (1-t)z_0^j\big)^\top \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{-1} \big(z_t - (1-t)z_0^j\big)\Big), \tag{5.28}$$

such that the corresponding normalized weights in Eq. (5.27) can be expressed by

$$w_j(z_t, t) = \frac{e_j(z_t, t)}{\sum_{j'=1}^{J} e_{j'}(z_t, t)}. \tag{5.29}$$

On the other hand, because Σ is positive definite, $((1-t)^2\Sigma + t\mathbf{I}_d)$ is positive definite for all $t \in [0, 1]$, and hence invertible. In particular, the score function has no singularity at

$t = 0$. The associated probability flow ODE in Eq. (5.21) becomes

$$\begin{aligned} dz_t =& \left[\frac{1}{t-1} z_t - \frac{1+t}{2(1-t)} S(z_t, t)\right] dt \\ =& \Bigg[\frac{1}{t-1} z_t - \frac{1+t}{2(1-t)} \Bigg(- ((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} z_t \\ & + (1-t)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j w_j(z_t, t)\Bigg)\Bigg] dt \\ =& \Bigg[\frac{1}{2}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} z_t \\ & - \frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j w_j(z_t, t)\Bigg] dt \end{aligned} \tag{5.30}$$

with the initial condition $z_1 \sim \mathcal{N}(0, \mathbf{I}_d)$. Although the right-hand side of Eq. (5.30) appears singular as $t \to 1$, substituting the score expression in Eq. (5.27) shows that the $(t-1)^{-1}$ terms are fully compensated by corresponding terms in the score. Consequently, the drift coefficients extend continuously to $t = 1$, and the apparent singularity is entirely removable at the coefficient level.

To compute numerical solutions of the ODE in Eq. (5.30), we adopt the explicit Euler method. The resulting discretization scheme takes the form

$$\begin{aligned} z^{k+1} =& z^k - \Bigg[\frac{1}{2}(\mathbf{I}_d - 2kh\Sigma)(k^2h^2\Sigma + (1-kh)\mathbf{I}_d)^{-1} z^k \\ & - \frac{2-kh}{2}(k^2h^2\Sigma + (1-kh)\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j w_j(z^k, (1-kh))\Bigg] h, \end{aligned} \tag{5.31}$$

where $K = 1/h,\ k = 0, 1, \ldots, K-1$ and the initial state z^0 is sampled from the standard normal distribution $\mathcal{N}(0, \mathbf{I}_d)$.

Remark 5.1. *We emphasize that the error splitting approach in Eq.* (5.20) *and the Gaussian mixture assumption in Eq.* (5.19) *ensures that the exact score function of the diffusion*

model can be analytically derived, i.e., Eq. (5.27)*, such that the exact score can be used in solving the reverse ODE to avoid the accumulation of the score estimation error. In fact, by comparing the Monte Carlo estimator in Eq.* (5.12) *and the exact score in Eq.* (5.27)*, we can quickly derive that Eq.* (5.27) *converges to Eq.* (5.12) *as* $\Sigma \to 0$*. In other words, Eq.* (5.12) *can also be viewed as the exact score when the target distribution is the empirical distribution defined by* $\mathcal{D}_{\text{train}}$ *without any smoothing. This answers the first question presented at the beginning of Section 5.2. On the other hand, Eq.* (5.30) *answers the second question regarding the singularity of the diffusion model presented in Section 5.2. We will answer the third question in the next subsection.*

5.2.2 Error Estimates for the Discretized Reverse ODE

We derive global error bounds for the reverse ODE discretization error defined in Eq. (5.20). Our analysis builds on the mixture surrogate introduced in Assumption 5.1, which smooths the empirical distribution and ensures a well-behaved score function throughout the reverse diffusion process. To facilitate the analysis, we impose the following assumption on the boundedness of the training dataset $\mathcal{D}_{\text{train}}$.

Assumption 5.2. *The samples in the training dataset* $\mathcal{D}_{\text{train}}$ *in Eq.* (5.2) *satisfy*

$$\left\|z_0^j\right\| \leq M < \infty, \quad j = 1, \ldots, J,$$

where M *is the radius of the smallest* ℓ_2 *ball that can cover the entire training set.*

Under this assumption, the score function and drift field in Eq. (5.17) admit uniform Lipschitz bounds in space and controlled variation in time, enabling dimension-explicit global error estimates for the explicit Euler scheme. Our error bounds are derived in two steps. The first step, presented in Section 5.2.2, is to analyze the path-wise error for a single trajectory of the reverse ODE; the second step, presented in Section 5.2.2, is to analyze the error under expectation.

The Error Bound for a Single Trajectory of the Reverse ODE

Before deriving the global error of the discretization scheme in Eq. (5.31) for a given terminal value z_1, we first bound the entire exact path $t \mapsto z_t$ in terms of z_1. This pathwise envelope clarifies how the solution remains confined as it propagates backward from $t = 1$ to $t = 0$, and it prevents growth of constants that would otherwise obscure the accumulation of local discretization errors. With this bound in place, all subsequent estimates for the Euler iterates can be stated with constants that depend only on the ODE data and on $\|z_1\|$, ensuring a clean and stable comparison between the numerical trajectory and the exact one in the entire time domain $[0, 1]$.

Lemma 5.1. *Under Assumption 5.1 and 5.2, the solution z_t of the reverse ODE in Eq.* (5.30) *satisfies*

$$\|z_t\| \le \frac{\max\left\{\sqrt{\lambda_{\max}}, 1\right\}}{\min\left\{\sqrt{\lambda_{\min}}, \frac{1}{4}\right\}}\big(M + \|z_1\|\big), \quad \forall\, 0 \le t \le 1, \tag{5.32}$$

where M is the bound in Assumption 5.2, $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and the largest eigenvalues of the covariance matrix Σ of the Gaussian mixture model defined in Eq. (5.19)*.*

Proof. For notational simplicity, we rewrite the reverse ODE in Eq. (5.30) as the following form

$$dz_t = [A(t)z_t + g(z_t, t)]\, dt, \tag{5.33}$$

where the function $A(t)$ is defined by

$$A(t) = \frac{1}{2}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}, \tag{5.34}$$

and the function $g(z_t, t)$ is defined by

$$g(z_t, t) = -\frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\sum_{j=1}^{J} z_0^j w_j(z_t, t). \tag{5.35}$$

By the variation of parameters formula, the solution of reverse ODE in Eq. (5.33) can be written as

$$z_t = \exp\left(\int_1^t A(s)ds\right) z_1 - \int_t^1 \exp\left(\int_s^t A(u)du\right) g(z_s, s)\, ds. \tag{5.36}$$

Using the explicit form of the fundamental matrix associated with $A(t)$ in Eq. (5.34), we have

$$\begin{aligned} z_t =& \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} z_1 \\ &- \int_t^1 \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} \big((1-s)^2\Sigma + s\mathbf{I}_d\big)^{-\frac{1}{2}} g(z_s, s)\, ds. \end{aligned} \tag{5.37}$$

Substituting the definition of $g(z_s, s)$ in Eq. (5.35) into Eq. (5.37), we have

$$\begin{aligned} z_t =& \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} z_1 \\ &+ \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} \int_t^1 \frac{1+s}{2} \big((1-s)^2\Sigma + s\mathbf{I}_d\big)^{-\frac{3}{2}} \sum_{j=1}^{J} z_0^j\, w_j(z_s, s)\, ds. \end{aligned} \tag{5.38}$$

Taking the ℓ_2 norm on both sides of Eq. (5.38) and using the triangle inequality and sub-multiplicativity of matrix norms, we obtain

$$\begin{aligned} \|z_t\| \le& \left\| \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} \right\| \|z_1\| + \left\| \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} \right\| \\ &\cdot \int_t^1 \frac{1+s}{2} \left\| \big((1-s)^2\Sigma + s\mathbf{I}_d\big)^{-\frac{3}{2}} \right\| \sum_{j=1}^{J} \left\| z_0^j \right\| w_j(z_s, s)\, ds. \end{aligned} \tag{5.39}$$

Next we bound each term on the left of Eq. (5.39) uniformly in d. Since Σ is positive definite with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$ and $0 \le t \le 1$, it holds that

$$\left\| \big((1-t)^2\Sigma + t\mathbf{I}_d\big)^{\frac{1}{2}} \right\| \le \big((1-t)^2\lambda_{\max} + t\big)^{\frac{1}{2}},$$

and

$$\left\| \big((1-s)^2\Sigma + s\mathbf{I}_d\big)^{-\frac{3}{2}} \right\| \le \big((1-s)^2\lambda_{\min} + s\big)^{-\frac{3}{2}}.$$

Moreover, by Assumption 5.2, we have

$$\sum_{j=1}^{J} \left\| z_0^j \right\| \, w_j(z_s, s) \leq M \sum_{j=1}^{J} w_j(z_s, s) = M.$$

Substituting the above bounds into Eq. (5.39) gives

$$\begin{aligned} \|z_t\| \leq & \left((1-t)^2 \lambda_{\max} + t\right)^{\frac{1}{2}} \|z_1\| \\ & + \left((1-t)^2 \lambda_{\max} + t\right)^{\frac{1}{2}} \int_t^1 \frac{1+s}{2} \left((1-s)^2 \lambda_{\min} + s\right)^{-\frac{3}{2}} M \, ds. \end{aligned} \tag{5.40}$$

Finally, the integral term in Eq. (5.40) can be evaluated explicitly, leading to

$$\begin{aligned} \|z_t\| \leq & \left((1-t)^2 \lambda_{\max} + t\right)^{\frac{1}{2}} \|z_1\| \\ & + \left((1-t)^2 \lambda_{\max} + t\right)^{\frac{1}{2}} \left((1-t)^2 \lambda_{\min} + t\right)^{-\frac{1}{2}} (1-t) \, M. \end{aligned} \tag{5.41}$$

Since Σ is positive definite and $0 \leq t \leq 1$, we further have

$$\begin{aligned} \max_{0 \leq t \leq 1} \left((1-t)^2 \lambda_{\max} + t \right) &= \begin{cases} \lambda_{\max}, & \lambda_{\max} \geq 1, \\ 1, & 0 < \lambda_{\max} < 1, \end{cases} \\ &\leq \max\{\lambda_{\max}, \, 1\}. \end{aligned} \tag{5.42}$$

and we have

$$\begin{aligned} \max_{0 \leq t \leq 1} \left((1-t)^2 \lambda_{\min} + t \right)^{-1} &= \begin{cases} \dfrac{1}{\lambda_{\min}}, & 0 < \lambda_{\min} \leq \dfrac{1}{2}, \\ \dfrac{4\lambda_{\min}}{4\lambda_{\min} - 1}, & \lambda_{\min} > \dfrac{1}{2}, \end{cases} \\ &\leq \max \left\{ \frac{1}{\lambda_{\min}}, \, 2 \right\} = \frac{1}{\min\{\lambda_{\min}, \, \frac{1}{2}\}}. \end{aligned} \tag{5.43}$$

Using the bounds in Eq. (5.42) and Eq. (5.43), we immediately obtain

$$\begin{aligned}\big((1-t)^2\lambda_{\mathsf{max}}+t\big)^{\frac{1}{2}} &\le \mathsf{max}\left\{\sqrt{\lambda_{\mathsf{max}}},\,1\right\},\\ \big((1-t)^2\lambda_{\mathsf{min}}+t\big)^{-\frac{1}{2}} &\le \frac{1}{\mathsf{min}\{\sqrt{\lambda_{\mathsf{min}}},\,\frac{1}{4}\}}.\end{aligned} \tag{5.44}$$

Substituting these bounds into Eq. (5.41) and using $1-t\le 1$, we obtain

$$\begin{aligned}\|z_t\| \le& \mathsf{max}\{\sqrt{\lambda_{\mathsf{max}}},\,1\}\,\|z_1\| + \frac{\mathsf{max}\{\sqrt{\lambda_{\mathsf{max}}},1\}}{\mathsf{min}\{\sqrt{\lambda_{\mathsf{min}}},\,\frac{1}{4}\}}\,M\\ \le& \frac{\mathsf{max}\{\sqrt{\lambda_{\mathsf{max}}},1\}}{\mathsf{min}\{\sqrt{\lambda_{\mathsf{min}}},\,\frac{1}{4}\}}\big(\|z_1\|+M\big),\end{aligned} \tag{5.45}$$

which completes the proof. □

Next, we derive a uniform-in-time Lipschitz constant in z_t for the right-hand side of the reverse ODE in Eq. (5.30), and the bound of its second derivatives with respect to t. For this purpose we present two lemmas that provide derivative estimates for the weights $w_j(z_t,t)$ in Eq. (5.29).

Lemma 5.2. *Under Assumption 5.1 and 5.2, for each $j=1,\dots,J$, the gradient of $w_j(z_t,t)$ in Eq.* (5.29) *with respect to z_t satisfies*

$$\|\nabla_{z_t} w_j(z_t,t)\| \le \frac{2M}{\mathsf{min}\{\lambda_{\mathsf{min}},\,\frac{1}{2}\}} w_j(z_t,t), \tag{5.46}$$

where M is the bound in Assumption 5.2 and λ_{min} denotes the minimal eigenvalue of the covariance matrix Σ of the Gaussian mixture model in Eq. (5.19)*.*

Proof. From the definition of $w_j(z_t, t)$ given in Eq. (5.29), we have

$$
\begin{aligned}
&\nabla_{z_t} w_j(z_t, t) \\
=\ & \nabla_{z_t} \frac{\exp(-\frac{1}{2}(z_t - (1-t)z_0^j)^T((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}(z_t - (1-t)z_0^j))}{\sum_{j'=1}^J \exp(-\frac{1}{2}(z_t - (1-t)z_0^{j'})^T((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}(z_t - (1-t)z_0^{j'}))} \\
=\ & \nabla_{z_t} \frac{e_j(z_t, t)}{\sum_{j'=1}^J e_{j'}(z_t, t)} \\
=\ & \frac{((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}((1-t)z_0^j - z_t)e_j(z_t, t)}{\sum_{j'=1}^J e_{j'}(z_t, t)} \\
& - \frac{e_j(z_t, t)\left(\sum_{j'=1}^J ((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}((1-t)z_0^{j'} - z_t)e_{j'}(z_t, t)\right)}{\left(\sum_{j'=1}^J e_{j'}(z_t, t)\right)^2} \\
=\ & ((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} w_j(z_t, t)\left[((1-t)z_0^j - z_t) - \sum_{j'=1}^J ((1-t)z_0^{j'} - z_t)w_{j'}(z_t, t)\right].
\end{aligned}
$$

Noting that $\sum_{j'=1}^J w_{j'}(z_t, t) = 1$, we can simplify the above equation as

$$
\begin{aligned}
\nabla_{z_t} w_j(z_t, t) =& ((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} w_j(z_t, t) \\
& \cdot \left[\sum_{j'=1}^J ((1-t)z_0^j - z_t)w_{j'}(z_t, t) - \sum_{j'=1}^J ((1-t)z_0^{j'} - z_t)w_{j'}(z_t, t)\right] \\
=& (1-t)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} w_j(z_t, t) \sum_{j'=1}^J (z_0^j - z_0^{j'})w_{j'}(z_t, t).
\end{aligned}
$$

Taking the norm on both sides yields the estimate

$$
\|\nabla_{z_t} w_j(z_t, t)\| \le \left\|(1-t)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\right\| w_j(z_t, t) \sum_{j'=1}^J \left\|z_0^j - z_0^{j'}\right\| w_{j'}(z_t, t). \tag{5.47}
$$

Since Σ is positive definite and $0 \le t \le 1$, we have

$$
\max_{0\le t\le 1}\left\|\left(1-t)((1-t)^2\Sigma + t\mathbf{I}_d\right)^{-1}\right\| = \max_{0\le t\le 1} \frac{1-t}{(1-t)^2\lambda_{\min} + t} \le \frac{1}{\min\{\lambda_{\min}, \frac{1}{2}\}}, \tag{5.48}
$$

where the last inequality follows from Eq. (5.43). Furthermore, we have $\|z_0^j\| < M$ for all j in Assumption (5.2). Hence, for all j, j', $\|z_0^j - z_0^{j'}\| \leq 2M$. Substituting this into Eq. (5.47), and applying Eq. (5.48), we deduce

$$\|\nabla_{z_t} w_j(z_t, t)\| \leq \frac{2M}{\min\{\lambda_{\min}, \frac{1}{2}\}} w_j(z_t, t) \sum_{j=1}^{J} w_j(z_t, t) = \frac{2M}{\min\{\lambda_{\min}, \frac{1}{2}\}} w_j(z_t, t). \tag{5.49}$$

This completes the proof. □

Lemma 5.3. *Under Assumptions 5.1 and 5.2, for each $j = 1, \ldots, J$, the time derivative of $w_j(z_t, t)$ in Eq.* (5.29) *satisfies*

$$\left| \frac{\partial w_j(z_t, t)}{\partial t} \right| \leq \frac{2 + 2\lambda_{\max}}{\min\{\lambda_{\min}^2, \frac{1}{4}\}} (\|z_t\| + M)^2 w_j(z_t, t),$$

where M is the data bound in Assumption 5.2, $\lambda_{\min}$ and $\lambda_{\max}$ are respectively the smallest and largest eigenvalues of the covariance matrix Σ from the Gaussian mixture model in Eq. (5.19)*.*

Proof. We begin by estimating the time derivative of $e_j(z_t, t)$ in Eq. (5.28):

$$\begin{aligned}
\left| \frac{\partial e_j(z_t, t)}{\partial t} \right| &= \left| \frac{\partial}{\partial t} \exp\left(-\frac{1}{2} (z_t - (1-t) z_0^j)^T ((1-t)^2 \Sigma + t \mathbf{I}_d)^{-1} (z_t - (1-t) z_0^j) \right) \right| \\
&= e_j(z_t, t) \Big| - (z_t - (1-t) z_0^j)^T ((1-t)^2 \Sigma + t \mathbf{I}_d)^{-1} z_0^j \\
&\quad + \frac{1}{2} (z_t - (1-t) z_0^j)^T ((1-t)^2 \Sigma + t \mathbf{I}_d)^{-1} (\mathbf{I}_d - 2(1-t)\Sigma) \\
&\quad \cdot ((1-t)^2 \Sigma + t \mathbf{I}_d)^{-1} (z_t - (1-t) z_0^j) \Big|.
\end{aligned}$$

By bounding each term using $\|((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\| \leq \frac{1}{\min\{\lambda_{\min}, \frac{1}{2}\}}$ in Eq. (5.43) and $\|z_0^j\| \leq M$ in Assumption 5.2, we obtain

$$\begin{aligned}
\left|\frac{\partial e_j(z_t,t)}{\partial t}\right| \leq & e_j(z_t,t)\left(\frac{(\|z_t\|+M)M}{\min\{\lambda_{\min},\frac{1}{2}\}} + \frac{(\|z_t\|+M)^2(1+2\lambda_{\max})}{2\min\{\lambda_{\min}^2,\frac{1}{4}\}}\right) \\
\leq & e_j(z_t,t)\left(\frac{(\|z_t\|+M)^2}{2\min\{\lambda_{\min}^2,\frac{1}{4}\}} + \frac{(\|z_t\|+M)^2(1+2\lambda_{\max})}{2\min\{\lambda_{\min}^2,\frac{1}{4}\}}\right) \\
= & e_j(z_t,t)\frac{1+\lambda_{\max}}{\min\{\lambda_{\min}^2,\frac{1}{4}\}}(\|z_t\|+M)^2.
\end{aligned}$$

For the derivative of $w_j(z_t,t)$ in Eq. (5.29), we have

$$\begin{aligned}
\left|\frac{\partial w_j(z_t,t)}{\partial t}\right| &= \left|\frac{\partial}{\partial t}\frac{e_j(z_t,t)}{\sum_{j'=1}^J e_{j'}(z_t,t)}\right| \\
&\leq \left|\frac{\frac{\partial e_j(z_t,t)}{\partial t}}{\sum_{j'=1}^J e_{j'}(z_t,t)}\right| + \left|\frac{e_j(z_t,t)\sum_{j'=1}^J \frac{\partial e_{j'}(z_t,t)}{\partial t}}{(\sum_{j'=1}^J e_{j'}(z_t,t))^2}\right| \\
&\leq \frac{e_j(z_t,t)\frac{1+\lambda_{\max}}{\min\{\lambda_{\min}^2,\frac{1}{4}\}}(\|z_t\|+M)^2}{\sum_{j'=1}^J e_{j'}(z_t,t)} \\
&\quad + \frac{e_j(z_t,t)\sum_{j'=1}^J e_{j'}(z_t,t)\frac{1+\lambda_{\max}}{\min\{\lambda_{\min}^2,\frac{1}{4}\}}(\|z_t\|+M)^2}{(\sum_{j'=1}^J e_{j'}(z_t,t))^2} \\
&= \frac{2+2\lambda_{\max}}{\min\{\lambda_{\min}^2,\frac{1}{4}\}}(\|z_t\|+M)^2 w_j(z_t,t),
\end{aligned} \tag{5.50}$$

which completes the proof. □

With Lemmas 5.2 and 5.3, we have established explicit upper bounds on the partial derivatives of the weight function $w_j(z_t,t)$ in Eq. (5.29) with respect to both the spatial variable z_t and the temporal variable t. Equipped with these tools, we are now ready to analyze the global error of the reverse ODE in Eq. (5.30).

Theorem 5.1. *Under Assumption 5.1 and 5.2, Lemma 5.2 and 5.3, consider the probability flow ODE in Eq.* (5.30) *with initial condition* z_1 *at* $t=1$. *The global error between* z_0 *and*

numerical solution z^K in Eq. (5.31) *is bounded by*

$$\begin{aligned}\left\|z_0 - z^K\right\| \leq & C \exp\left(\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2, \frac{1}{4}\}}\right) \\ & \cdot \frac{(1+\lambda_{\max})^2(1+M)\left(1+\frac{\max\{\sqrt{\lambda_{\max}},1\}}{\min\{\sqrt{\lambda_{\min}},\frac{1}{4}\}}(M+\|z_1\|)+M\right)^2}{(1+2\lambda_{\max}+8M^2)\min\{\lambda_{\min},\frac{1}{2}\}} h,\end{aligned} \tag{5.51}$$

where M is the bound in Assumption 5.2, $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of the covariance matrix Σ from the Gaussian mixture model in Eq. (5.19)*, h is the time step in Eq.* (5.31)*, and C is a constant independent of $\lambda_{\max}, \lambda_{\min}, z_1, M, h$ and d.*

Proof. For notational simplicity, we define the following drift function

$$\begin{aligned}f(z_t, t) := & \frac{1}{2}(1-2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} z_t \\ & - \frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j w_j(z_t, t),\end{aligned}$$

such that the reverse ODE in Eq. (5.30) can be rewritten as

$$\frac{dz_t}{dt} = f(z_t, t).$$

We first estimate the Jacobian of f with respect to z_t:

$$\begin{aligned}\frac{\partial f(z_t,t)}{\partial z_t} = & \frac{1}{2}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \\ & - \frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j \frac{\partial w_j(z_t,t)}{\partial z_t}.\end{aligned}$$

Taking the ℓ_2 norm, we obtain

$$\begin{aligned}\left\|\frac{\partial f(z_t,t)}{\partial z_t}\right\| \leq & \frac{1}{2}\|(\mathbf{I}_d - 2(1-t)\Sigma)\|\|((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\| \\ & + \frac{1+t}{2}\left\|((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\right\|\left\|\sum_{j=1}^{J} z_0^j \frac{\partial w_j(z_t,t)}{\partial z_t}\right\| \\ \leq & \frac{1+2\lambda_{\max}}{2\min\{\lambda_{\min},\frac{1}{2}\}} + \frac{M}{\min\{\lambda_{\min},\frac{1}{2}\}}\sum_{j=1}^{J}\frac{2M}{\min\{\lambda_{\min},\frac{1}{2}\}} w_j(z_t,t),\end{aligned}$$

where the last inequality follows Eq. (5.48) and Lemma 5.2. Since $\sum_{j=1}^{J} w_j(z_t,t) = 1$, we have

$$\begin{aligned}\left\|\frac{\partial f(z_t,t)}{\partial z_t}\right\| \leq & \frac{1+2\lambda_{\max}}{2\min\{\lambda_{\min},\frac{1}{2}\}} + \frac{2M^2}{\min\{\lambda_{\min}^2,\frac{1}{4}\}} \\ \leq & \frac{1+2\lambda_{\max}}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}} + \frac{2M^2}{\min\{\lambda_{\min}^2,\frac{1}{4}\}} = \frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}},\end{aligned} \tag{5.52}$$

which shows $f(z_t,t)$ is Lipschitz continuous in z_t with Lipschitz constant $\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}}$. Next, we estimate the second-order derivative:

$$\begin{aligned}\left\|\frac{d^2 z_t}{dt^2}\right\| &= \left\|\frac{\partial f(z_t,t)}{\partial z_t}\cdot\frac{dz_t}{dt} + \frac{\partial f(z_t,t)}{\partial t}\right\| \\ &\leq \left\|\frac{\partial f(z_t,t)}{\partial z_t}\right\|\cdot\|f(z_t,t)\| + \left\|\frac{\partial f(z_t,t)}{\partial t}\right\|.\end{aligned} \tag{5.53}$$

We estimate the norm of $f(z_t,t)$ as follows:

$$\begin{aligned}\|f(z_t,t)\| = & \left\|\frac{1}{2}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}z_t\right. \\ & \left. - \frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\sum_{j=1}^{J} z_0^j w_j(z_t,t)\right\| \\ \leq & \frac{1}{2}\|\mathbf{I}_d - 2(1-t)\Sigma\|\|((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\|\|z_t\| \\ & + \frac{1+t}{2}\left\|((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}\right\|\left\|\sum_{j=1}^{J} z_0^j w_j(z_t,t)\right\| \\ \leq & \frac{(1+2\lambda_{\max})\|z_t\|}{2\min\{\lambda_{\min},\frac{1}{2}\}} + \frac{M}{\min\{\lambda_{\min},\frac{1}{2}\}} = \frac{(1+2\lambda_{\max})\|z_t\| + 2M}{2\min\{\lambda_{\min},\frac{1}{2}\}}.\end{aligned} \tag{5.54}$$

Additionally, we can derive the partial derivative of $f(z_t, t)$ with respect to t:

$$
\begin{aligned}
&\frac{\partial f(z_t,t)}{\partial t} \\
=&\Sigma((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} z_t \\
&- \frac{1}{2}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} z_t \\
&- \frac{1}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j w_j(z_t,t) \\
&+ \frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1}(\mathbf{I}_d - 2(1-t)\Sigma)((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j w_j(z_t,t) \\
&- \frac{1+t}{2}((1-t)^2\Sigma + t\mathbf{I}_d)^{-1} \sum_{j=1}^{J} z_0^j \frac{\partial w_j(z_t,t)}{\partial t}.
\end{aligned}
\tag{5.55}
$$

To bound the time derivative $\frac{\partial f(z_t,t)}{\partial t}$ in Eq. (5.55), we apply Assumption 5.2, Lemma 5.3 and Eq. (5.48), yielding

$$
\begin{aligned}
\left\| \frac{\partial f(z_t,t)}{\partial t} \right\| \leq & \frac{\lambda_{\max}\|z_t\|}{\min\{\lambda_{\min}, \frac{1}{2}\}} + \frac{(1+2\lambda_{\max})^2\|z_t\|}{2\min\{\lambda_{\min}^2, \frac{1}{4}\}} + \frac{M}{2\min\{\lambda_{\min}, \frac{1}{2}\}} \\
& + \frac{(1+2\lambda_{\max})M}{\min\{\lambda_{\min}^2, \frac{1}{4}\}} + \frac{(2+2\lambda_{\max})(\|z_t\|+M)^2 M}{\min\{\lambda_{\min}^3, \frac{1}{8}\}} \\
\leq & \frac{\lambda_{\max}\|z_t\|}{4\min\{\lambda_{\min}^3, \frac{1}{8}\}} + \frac{(1+2\lambda_{\max})^2\|z_t\|}{4\min\{\lambda_{\min}^3, \frac{1}{8}\}} + \frac{M}{8\min\{\lambda_{\min}^3, \frac{1}{8}\}} \\
& + \frac{(1+2\lambda_{\max})M}{2\min\{\lambda_{\min}^3, \frac{1}{8}\}} + \frac{(2+2\lambda_{\max})(\|z_t\|+M)^2 M}{\min\{\lambda_{\min}^3, \frac{1}{8}\}} \\
\leq & C\frac{(1+\lambda_{\max})^2(1+M)(\|z_t\|+M)^2}{\min\{\lambda_{\min}^3, \frac{1}{8}\}},
\end{aligned}
\tag{5.56}
$$

where C is a constant independent of $\lambda_{\max}, \lambda_{\min}, \|z_t\|$ and M. Combining Eqs. (5.52), (5.53), (5.54), and (5.56), we obtain

$$
\left\| \frac{d^2 z_t}{dt^2} \right\| \leq C\frac{(1+\lambda_{\max})^2(1+M)(1+\|z_t\|+M)^2}{\min\{\lambda_{\min}^3, \frac{1}{8}\}}. \tag{5.57}
$$

Therefore, by the global error of the Euler scheme, together with Eq. (5.52) and Eq. (5.57), we have

$$\begin{aligned}\left\|z_0 - z^K\right\| &\leq \frac{\exp\left(\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}}\right)-1}{2\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}}} C\frac{(1+\lambda_{\max})^2(1+M)(1+\|z_t\|+M)^2}{\min\{\lambda_{\min}^3,\frac{1}{8}\}}h \\ &\leq C\exp\left(\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}}\right)\frac{(1+\lambda_{\max})^2(1+M)(1+\|z_t\|+M)^2}{(1+2\lambda_{\max}+8M^2)\min\{\lambda_{\min},\frac{1}{2}\}}h.\end{aligned} \tag{5.58}$$

Applying the bound of $\|z_t\|$ in Lemma (5.1), we obtain

$$\begin{aligned}\left\|z_0 - z^K\right\| \leq & C\exp\left(\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2,\frac{1}{4}\}}\right) \\ & \cdot\frac{(1+\lambda_{\max})^2(1+M)(1+\frac{\max\{\sqrt{\lambda_{\max}},1\}}{\min\{\sqrt{\lambda_{\min}},\frac{1}{4}\}}(M+\|z_1\|)+M)^2}{(1+2\lambda_{\max}+8M^2)\min\{\lambda_{\min},\frac{1}{2}\}}h,\end{aligned} \tag{5.59}$$

where C is a constant independent of $\lambda_{\max}, \lambda_{\min}, \|z_1\|, M, h$ and dimension d. The proof is complete. □

The Expected Error Bound for the Reverse ODE

We now analyze the expected global error of the reverse ODE with a Gaussian initial condition $z_1 \sim \mathcal{N}(0, \mathbf{I}_d)$. Since the integration of Eq. (5.30) proceeds backward from $t = 1$ to $t = 0$, different realizations of z_1 produce different pathwise global errors, see Eq. (5.51) in Theorem 5.1. Therefore, it is natural to quantify the error in expectation with respect to the distribution of z_1. We are now ready to state our main result.

Theorem 5.2. *Under Assumptions 5.1, 5.2, consider the probability flow ODE in Eq.* (5.30) *with initial condition $z_1 \sim \mathcal{N}(0, \mathbf{I}_d)$. Let z_0 denote the exact solution at $t = 0$, and let z^K be the numerical solution produced by the Euler scheme in Eq.* (5.31)*. Then the expected ℓ_2-norm of the global error satisfies*

$$\mathbb{E}_{Z_1}\left\|z_0 - z^K\right\| \leq Cdh, \tag{5.60}$$

where C is a constant independent of the time step h and the dimension d.

Proof. By Theorem 5.1, for a fixed z_1, the resulting global approximation error for the discretized dynamics (5.31) applied to the probability flow ODE in Eq. (5.30) is bounded as follows:

$$\begin{aligned}\left\|z_0 - z^K\right\| \leq & C \exp\left(\frac{1+2\lambda_{\max}+8M^2}{4\min\{\lambda_{\min}^2, \frac{1}{4}\}}\right) \\ & \cdot \frac{(1+\lambda_{\max})^2(1+M)(1+\frac{\max\{\sqrt{\lambda_{\max}},1\}}{\min\{\sqrt{\lambda_{\min}},\frac{1}{4}\}}(M+\|z_1\|)+M)^2}{(1+2\lambda_{\max}+8M^2)\min\{\lambda_{\min},\frac{1}{2}\}} h,\end{aligned} \tag{5.61}$$

Taking the expectation with respect to Z_1, we obtain

$$\mathbb{E}_{Z_1}\left\|z_0 - z^K\right\| \leq Ch\mathbb{E}_{Z_1}\left[(\|z_1\|+1)^2\right], \tag{5.62}$$

where C is a constant independent of the time step h and the dimension d. Since $Z_1 \sim \mathcal{N}(0, \mathbf{I}_d)$, $\|Z_1\|$ has the chi distribution with d degrees of freedom, and

$$\mathbb{E}_{Z_1}[\|z_1\|^2] = d, \qquad \mathbb{E}_{Z_1}[\|z_1\|] = \sqrt{2}\,\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \leq \sqrt{d}. \tag{5.63}$$

Substituting these into Eq. (5.62) yields

$$\mathbb{E}_{Z_1}\|z_0 - z^K\| \leq Ch\left(d + 2\sqrt{d} + 1\right) \leq Cdh, \tag{5.64}$$

where C is independent of the time step h and the dimensionality d. □

Having established an expected global error bound in the ℓ_2 norm, we next derive an analogous estimate in the ℓ_∞ norm. The ℓ_∞ control is complementary to the ℓ_2 result and provides a coordinatewise guarantee and is particularly useful when one seeks uniform accuracy across components. In the following, we quantify the expected ℓ_∞-norm of the global discretization error under the same setting.

Theorem 5.3. *Assume Assumptions 5.1, 5.2 and the Gaussian mixture model in Eq.* (5.19) *has a diagonal covariance matrix Σ. Then the expected ℓ_∞-norm of the global error satisfies*

$$\mathbb{E}_{Z_1} \left\| z_0 - z^K \right\|_\infty \leq C(\ln(d) + 1)h, \tag{5.65}$$

where C is a constant independent of the time step h and the dimension d.

Proof. When the covariance matrix Σ in (5.19) is diagonal, all matrix-valued quantities that arise in the probability flow ODE and in the associated Euler analysis remain diagonal as well. For a diagonal matrix, the induced operator ℓ_2-norm and ℓ_∞-norm coincide. Moreover, by Assumption 5.2 we have, for each $j = 1, \ldots, J$,

$$\|z_0^j\|_\infty \leq \|z_0^j\| \leq M,$$

so the data bound can be stated equivalently in the ℓ_∞ norm. Combining these observations, every matrix norm estimate established earlier in the ℓ_2 setting continues to hold verbatim when $\|\cdot\|$ is replaced by $\|\cdot\|_\infty$. In particular, we may invoke Theorem 5.1 to obtain a pathwise ℓ_∞ norm bound for the Euler discretization error for a fixed initial condition z_1

$$\begin{aligned} \left\| z_0 - z^K \right\|_\infty \leq & C \exp\left(\frac{1 + 2\lambda_{\max} + 8M^2}{4 \min\{\lambda_{\min}^2, \frac{1}{4}\}} \right) \\ & \cdot \frac{(1 + \lambda_{\max})^2 (1 + M)(1 + \frac{\max\{\sqrt{\lambda_{\max}}, 1\}}{\min\{\sqrt{\lambda_{\min}}, \frac{1}{4}\}}(M + \|z_1\|_\infty) + M)^2}{(1 + 2\lambda_{\max} + 8M^2) \min\{\lambda_{\min}, \frac{1}{2}\}} h, \end{aligned} \tag{5.66}$$

Taking the expectation with respect to Z_1, we obtain

$$\mathbb{E}_{Z_1} \left\| z_0 - z^K \right\|_\infty \leq C h \mathbb{E}_{Z_1} \left[(\|z_1\|_\infty + 1)^2 \right] \tag{5.67}$$

where C is a constant independent of the time step h and the dimension d.

The next step is to estimate the expectations $\mathbb{E}_{Z_1}\|z_1\|_\infty$ and $\mathbb{E}_{Z_1}\|z_1\|_\infty^2$. Let z_1^i denote the i-th component of z_1, where $z_1^i, \ldots, z_1^d$ are independent and identically distributed as $\mathcal{N}(0,1)$. Applying Jensen's inequality, we have, for $t > 0$,

$$\begin{aligned}
\exp(t\mathbb{E}_{Z_1}\|z_1\|_\infty) \leq & \mathbb{E}_{Z_1}[\exp(t\|z_1\|_\infty)] = \mathbb{E}_{Z_1}[\exp(t \max_{1\leq i\leq d} |z_1^i|)] = \mathbb{E}_{Z_1}[\max_{1\leq i\leq d} \exp(t|z_1^i|)] \\
\leq & \mathbb{E}_{Z_1}\left[\sum_{i=1}^d \left(\exp(tz_1^i) + \exp(-tz_1^i)\right)\right] \\
= & \sum_{i=1}^d \left(\mathbb{E}_{Z_1}[\exp(tz_1^i)] + \mathbb{E}_{Z_1}[\exp(-tz_1^i)]\right) = 2d\exp(t^2/2).
\end{aligned}$$

Taking the natural logarithm on both sides, we obtain

$$\mathbb{E}_{Z_1}\|z_1\|_\infty \leq \frac{\ln(2d)}{t} + \frac{t}{2}, \quad \forall\, t > 0.$$

Choosing the optimal value $t = \sqrt{2\ln(2d)}$ then yields

$$\mathbb{E}_{Z_1}\|z_1\|_\infty \leq \sqrt{2\ln(2d)}. \tag{5.68}$$

Similarly, we estimate the second moment by noting

$$\begin{aligned}
& \exp\left(\frac{1}{4}\mathbb{E}_{Z_1}\|z_1\|_\infty^2\right) \leq \mathbb{E}_{Z_1}\left[\exp\left(\frac{1}{4}\|z_1\|_\infty^2\right)\right] = \mathbb{E}_{Z_1}\left[\exp\left(\frac{1}{4}\max_{1\leq i\leq d}|z_1^i|^2\right)\right] \\
= & \mathbb{E}_{Z_1}\left[\max_{1\leq i\leq d}\exp\left(\frac{1}{4}|z_1^i|^2\right)\right] \leq \mathbb{E}_{Z_1}\left[\sum_{i=1}^d \exp\left(\frac{1}{4}(z_1^i)^2\right)\right] = \sum_{i=1}^d \mathbb{E}_{Z_1}\left[\exp\left(\frac{1}{4}(z_1^i)^2\right)\right] \\
= & \sqrt{2}d.
\end{aligned}$$

Taking the natural logarithm of both sides in the above, we find

$$\mathbb{E}_{Z_1}\|z_1\|_\infty^2 \leq 4\ln(\sqrt{2}d). \tag{5.69}$$

Combining Eqs. (5.67), (5.68), and (5.69), we conclude

$$\mathbb{E}_{Z_1} \left\| z_0 - z^K \right\|_\infty \leq C(\ln(d) + 1)h,$$

as desired. This completes the proof. □

5.2.3 Discussion on the Error Bounds

The choice of the covariance matrix Σ

In practice, choosing Σ of the Gaussian mixture model in Eq. (5.19) with either extremely small or extremely large eigenvalues tends to degrade numerical accuracy. For example, when the covariance matrix is in a diagonal form $\Sigma = \sigma \mathbf{I}_d$, the analytical expression of the Lipschitz constant $L(\sigma)$ (as a function of σ) of the linear part of the reverse ODE in Eq. (5.30) is plotted in Fig. 5.1. We observe that

the Lipschitz constant initially decreases as σ grows, reaches a favorable plateau, and then increases again. This captures the intrinsic trade-off in tuning the covariance scale, i.e., too small spread leads to stiffness, while too large spread weakens the contraction structure of the flow. Our numerical experiments in Section 5.3.1 verify this qualitative behavior and indicate that an intermediate range of σ achieves the most stable and accurate integration.

The Dimensionality Dependence

A typical origin of the curse of dimensionality in global error estimates for numerical ODE solvers is the exponential amplification from Grönwall-type bounds when the drift has a large Lipschitz constant. If the effective Lipschitz constant L grows with the ambient dimension d, the error prefactor can behave like $\exp(CL)$, where C is a positive constant independent of time step h.

In our setting, the drift of Eq. (5.30) depends on the Gaussian–mixture weights $w_i(z_t, t)$. Lemma 5.2 bounds the spatial sensitivity, which yields a uniform control of the Jacobian of the drift with respect to z_t that depends on M and on spectral quantities of Σ but does not blow up exponentially with d. Specifically, the estimate Eq. (5.52) inside Theorem 5.1 provides a bounded Lipschitz constant for the drift, thereby preventing the exponential in d amplification in the Grönwall factor and ensuring a first-order global error bound with a dimension-stable prefactor. The only remaining d-dependence appears when taking expectation over the Gaussian initial condition $z_1 \sim \mathcal{N}(0, \mathbf{I}_d)$: one has $\mathbb{E}\|z_1\|^2 = d$ and $\mathbb{E}\|z_1\| \leq \sqrt{d}$, which leads to the linear dependence in d of Theorem 5.2. Thus the expected error scales like $\mathcal{O}(dh)$ rather than $O(\exp(d)h)$. We note that the constants deteriorate as $\lambda_{\min} \downarrow 0$, reflecting ill-conditioning of the mixture components; under Assumptions 5.2, spectral control keeps the drift's Lipschitz modulus bounded and averts dimension-induced exponential growth. The same mechanism extends to non-diagonal Σ under comparable spectral bounds.

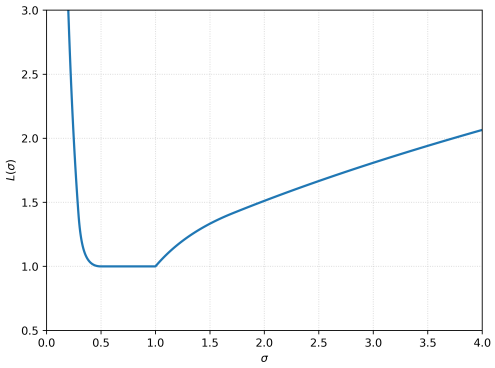

Figure 5.1: Upper bound of the Lipschitz constant $L(\sigma)$ of the linear part of the reverse ODE in Eq. (5.30).

5.3 Numerical Experiments

We conduct a set of numerical experiments to validate the error bounds derived in Section 5.2. Since the analysis in Section 5.2 accounts only for the reverse ODE discretization error under Assumption 5.1, we choose the target distribution $p_X(x) = q_{Z_0}(z_0)$

to be a Gaussian mixture model defined as

$$q_{Z_0}(z_0) := \frac{1}{J}\sum_{j=1}^{J} \phi\big(z_0; z_0^j, \Sigma\big)\,, \tag{5.70}$$

where $\phi(\cdot)$ denotes the Gaussian density function, J is the number of mixture components, $\{z_0^j\}_{j=1}^J$ are the component means, and Σ is a shared diagonal covariance matrix. The specific choices of $\{z_0^j\}_{j=1}^J$ and Σ are specified in each experiment. By adopting the Gaussian mixture model in (5.70) as the target distribution, we eliminate the contribution of the data error term in (5.20). This allows us to isolate the reverse ODE discretization error and directly assess the sharpness of the theoretical error bounds established in Section 5.2.

5.3.1 Verification of the Discretization Error Bounds of the Reverse ODE

We verify the error bounds of the reverse ODE, focusing on how the error depends on the dimension, the time step size, and the covariance structure of the Gaussian mixture model. To quantify the discretization error, we use second-order Heun method with a small step size $h_{\mathsf{ref}} = 10^{-3}$ as a reference solution. In all experiments we consider a Gaussian mixture with $J = 10$ modes. The mean vectors $\{z_0^j\}_{j=1}^J$ in Eq. (5.70) are drawn differently depending on the evaluation norm. For the experiments verifying error bounds in ℓ_2 norm, we sample $\{z_0^j\}_{j=1}^J$ independently and uniformly from the ball of radius M in $\mathbb{R}^d$, while for the experiments verifying the error bounds in ℓ_∞ norm, we sample independently and uniformly from the hypercube $[-M, M]^d$. The expectation of the error bounds in Theorems 5.2 and 5.3 is approximated by averaging the errors of 1000 trajectories.

Verifying the dimension dependence Fig. 5.2 demonstrates the dimension dependence estimated in Theorems 5.2 and 5.3. The constant in Assumption 5.2 is set to $M = 1$, and the covariance matrix in Eq. (5.70) is chosen as a full symmetric positive definite (SPD) matrix $\Sigma = U\Lambda U^\top$, where $U \in \mathbb{R}^{d\times d}$ is a random orthogonal matrix and $\Lambda = \mathsf{diag}(\sigma_1, \ldots, \sigma_d)$ collects the eigenvalues of Σ. We generate $\{\sigma_k\}_{k=1}^d$ by periodically repeating the pattern

$(0.1, 0.2, 0.3, 0.4, 0.5)$ until reaching dimension d. We observe that the ℓ_2 error increases approximately linearly with the dimension d, and the ℓ_∞ error exhibits an $\mathcal{O}(\log d)$ growth rate. These scaling behaviors are consistent with the dimension dependence predicted by Theorems 5.2 and 5.3.

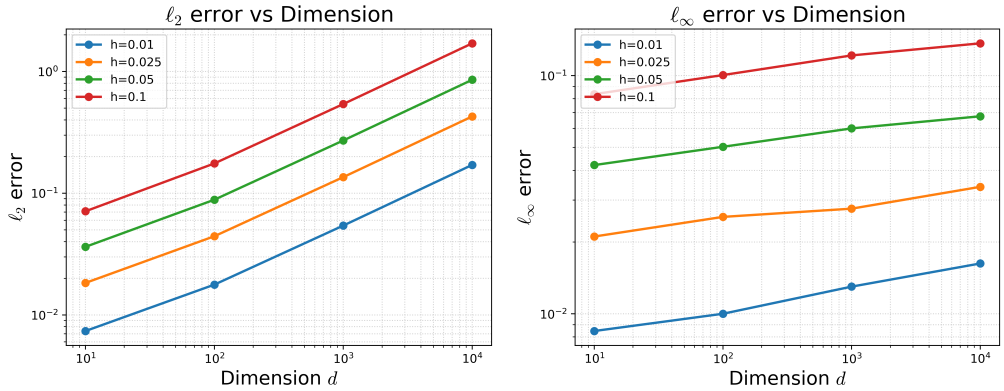

Figure 5.2: Expected error versus dimension d.

Verifying the first-order temporal convergence rate We now verify the convergence rate with respect to the time step size h presented in Theorem 5.2 and Theorem 5.3. The time step size is set to $h \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$, and the dimension is set to $d \in \{10, 100, 1000, 10000\}$. The bound defined in Assumption 5.2 is set to $M = 1$, and we use the same full SPD covariance setting as above: $\Sigma = U\Lambda U^\top$ with $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_d)$, where $\{\sigma_k\}_{k=1}^d$ repeats the pattern $(0.1, 0.2, 0.3, 0.4, 0.5)$. Fig. 5.3 shows that our method achieves a perfect first-order convergence rate with respect to the time step size h in both ℓ_2 and ℓ_∞ norm, which demonstrates the sharpness of our error bounds.

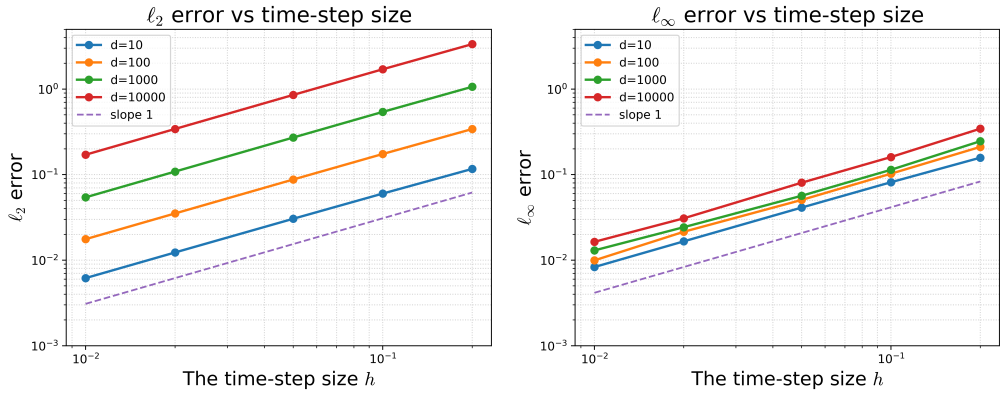

Figure 5.3: Expected error versus time-step size h.

Verifying the dependence on the smoothing constant σ We now verify the dependence of the error bound derived in Theorem 5.1 on the smoothing constant σ. The covariance matrix in Eq. (5.70) is chosen to be isotropic, $\Sigma = \sigma I_d$. We vary $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ while fixing $d = 10$, $h = 0.01$, and $M = 1$. Fig. 5.4 shows that the ℓ_2 and ℓ_∞ errors decrease for small σ and then increase as σ grows, which is consistent with the discussion in Section 5.2.3 and the plot in Fig. 5.1.

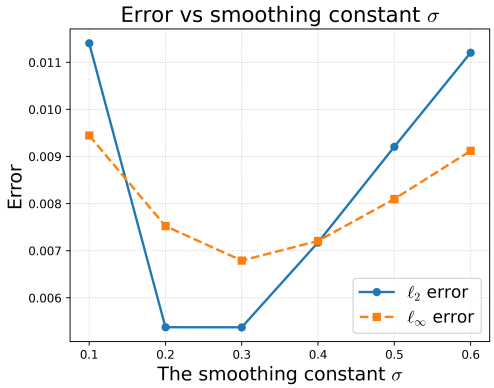

Figure 5.4: Expected error versus smoothing constant σ in Eq. (5.70).

5.3.2 Verification of the Supervised Learning Error

To demonstrate the practical effectiveness of the supervised learning strategy used to construct the generative model in (5.3), we present additional numerical experiments to assess the relative contributions of the reverse ODE discretization error and the supervised learning error in Eq. (5.20), and to determine which error source is dominant in practice.

Experiment setup. We use the reverse ODE with the exact score function to generate training pairs, and then train neural networks to directly learn the noise-to-sample map $z_1 \mapsto z_0$. We still use Eq. (5.70) as the target distribution while setting $J = 10$. Each Gaussian component's mean vector has coordinates drawn independently from $\{-1, 1\}$ with equal probability, and we use a shared diagonal covariance matrix with diagonal entries drawn uniformly from $[0.2, 0.4]$. We test across dimensions $d \in \{8, 16, 32, 64, 128, 256, 512,$

$1024, 2048, 4096, 8192\}$. To generate labeled data in the training set $\mathcal{D}_{\text{train}}$, we sample $z_1 \sim \mathcal{N}(0, I_d)$ from the standard Gaussian and solve for the corresponding z_0 through the reverse ODE. The reference solution is generated with $h = 0.001$ and the approximate solution is generated with $h = 0.01$.

For each dimension, we generate 100,000 samples and split them into 80,000 training samples, 10,000 validation samples, and 10,000 test samples. We repeat each experiment 5 times with different random splits to assess variability. We compare two Multi-layer Perceptron (MLP) architectures with different capacities: MLP-1024 and MLP-2048, which have hidden dimensions of 1024 and 2048, respectively. Both architectures consist of two hidden layers with GELU activation functions. The models are trained using the AdamW optimizer with learning rate 10^{-3} and weight decay 10^{-4}. Training continues until the validation loss fails to improve for 50 consecutive epochs (early stopping), at which point the model with the lowest validation loss is selected.

Results and discussion

Fig. 5.5 shows the root mean squared error (RMSE) for different model configurations across dimensions. For each dimension, we compare two types of errors on both training and test sets, i.e., the reverse ODE discretization error and the neural network approximation error. We observe that the reverse ODE discretization error (shown in red) remains consistently small, ranging from approximately 0.01 to 0.05 in RMSE. This error grows slowly with dimension and remains nearly flat across the tested scenarios.

Across all test cases, the total sampling error with respect to the true distribution is dominated by neural network approximation error, which exceeds the ODE discretization error by one to two orders of magnitude, as clearly visible in the log-scale plot (right panel). This demonstrates that improving ODE solvers beyond a reasonable accuracy threshold yields diminishing returns. Instead, research efforts should prioritize three directions. First, scaling network architectures to handle high-dimensional noise to sample mappings. Second, improving data efficiency through better training procedures or data

augmentation. Third, developing architectures that maintain strong generalization as the dimension increases.

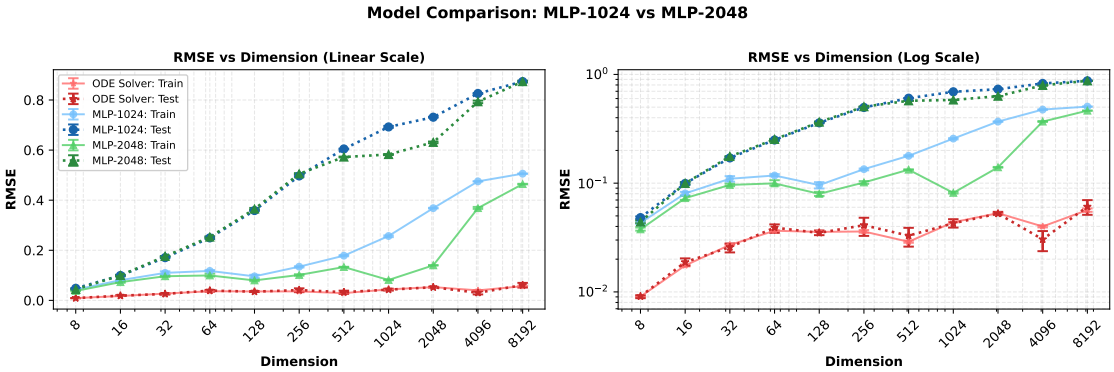

Figure 5.5: Root mean squared error (RMSE) comparison across dimensions for supervised learning of the noise-to-sample mapping.

Chapter 6 Summary and Future Work

6.1 Summary

The main contributions of this dissertation are the development and analysis of pseudo-reversible normalizing flows and training-free diffusion models for generative modeling in scientific computing. These methods construct generative surrogate models that transform samples from a simple reference distribution into samples from target probability distributions. The pseudo-reversible normalizing flow framework is developed for conditional sampling in stochastic dynamical systems and uncertainty quantification. The training-free diffusion framework is studied for high-dimensional sampling and the numerical analysis of probability-flow ordinary differential equations.

Chapter 3 proposed a pseudo-reversible normalizing flow for stochastic dynamical systems with different initial distributions. The novelty of our normalizing flow approach can be viewed from two perspectives. Firstly, it enables learning the distribution of the final state based on any given initial state. This allows the model to be trained just once, and then applied to handle various initial distributions effectively. Secondly, we developed a pseudo-reversible neural network architecture that relaxes the strict reversibility constraint and facilitates the convergence analysis. Under suitable assumptions, the convergence of the learned distribution to the target distribution was established in the Kullback–Leibler divergence. Numerical experiments for Brownian motion, runaway-electron dynamics, and advection–diffusion transport demonstrated that the learned model can serve as a surrogate for conditional sampling in these settings.

Chapter 4 extended the pseudo-reversible normalizing flow framework to uncertainty quantification for noisy physical models. The conditional generative model was designed to learn probability distributions in both forward and inverse uncertainty propagation directly from input-output data. In particular, the method does not require separate knowledge of the deterministic model component or the distribution of the additive

noise. The conditional PR-NF architecture retained the approximate reversibility principle while incorporating the conditioning variable into both transport maps. The analysis established convergence for the forward conditional distribution, and the numerical examples considered synthetic problems, high-dimensional uncertainty propagation, and a geologic carbon storage application. These results showed that conditional generative surrogate models can represent uncertainty beyond a deterministic input-output approximation, provided that the conditioning variables remain within the range represented by the training data.

Chapter 5 developed a rigorous and numerically verifiable error analysis for a class of training-free diffusion models used to generate labeled data for supervised learning of end-to-end generative samplers. By decomposing the total error into data, discretization, and supervised learning components, we isolated the key source of theoretical uncertainty and focused on the discretization error arising from the reverse-time diffusion process when a Gaussian mixture model is treated as the target distribution. A central feature of our analysis is the exploitation of analytically available score functions for Gaussian mixture models, which allows us to completely avoid the propagation of score-function approximation errors through the diffusion process. As a result, we recover classical convergence rates associated with standard ODE discretization schemes and obtain error bounds with favorable dependence on the problem dimensionality. Importantly, the derived estimates are not only theoretically rigorous but also directly verifiable through numerical experiments, both with respect to time-step size and dimensionality, thereby addressing a key gap between existing diffusion model theory and observed numerical behavior.

6.2 Future Work

A natural direction is to extend the pseudo-reversible normalizing flow framework to time-dependent and higher-dimensional stochastic systems. The models developed in

Chapter 3 learn the terminal distributions of SDE. Future work may incorporate time as an additional conditioning variable or construct models for temporally consistent trajectories.

Another direction is to develop conditional generative surrogates for more general uncertainty quantification and Bayesian inverse problems. The conditional PR-NF framework can be extended to high-dimensional inputs and outputs through latent-space representations. It is also of interest to establish theoretical guarantees for inverse conditional distributions, especially when these distributions are multimodal. In recursive Bayesian estimation, a further goal is to reuse a trained conditional surrogate as new observations arrive, rather than repeatedly solving the full forward problem or rerunning a sampling procedure.

For training-free diffusion models, an important extension is to move beyond Gaussian mixture approximations. More general mixture components or distributions concentrated near lower dimensional structures may better represent complex target distributions, but they also require new score representations and stability analysis for the associated probability flow dynamics. It would also be useful to investigate higher order or adaptive discretization methods and to design probability flows with smoother and better conditioned trajectories. Such developments may lead to sharper error estimates for generated samples, densities, and quantities of interest.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

[2] Juan Alcalde, Stephanie Flude, Mark Wilkinson, Gareth Johnson, Katriona Edlmann, Clare E Bond, Vivian Scott, Stuart MV Gilfillan, Xènia Ogaya, and R Stuart Haszeldine. Estimating geological CO_2 storage security to deliver on climate mitigation. *Nature communications*, 9(1):2201, 2018.

[3] Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[4] Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Nearly d-linear convergence bounds for diffusion models via stochastic localization. In *International Conference on Learning Representations*, volume 2024, pages 36916–36936, 2024.

[5] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[6] Adam Block, Youssef Mroueh, and Alexander Rakhlin. Generative modeling with denoising auto-encoders and langevin sampling. *arXiv preprint arXiv:2002.00107*, 2020.

[7] Vanessa Böhm, François Lanusse, and Uroš Seljak. Uncertainty quantification with generative models. *arXiv preprint arXiv:1910.10046*, 2019.

[8] Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research*, 2022.

[9] Gert-Jan Both and Remy Kusters. Temporal normalizing flows. *arXiv preprint arXiv:1912.09092*, 2019.

[10] Boris N Breizman, Pavel Aleynikov, Eric M Hollmann, and Michael Lehnen. Physics of runaway electrons in tokamaks. *Nuclear Fusion*, 59(8):083001, 2019.

[11] Russel E Caflisch. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7:1–49, 1998.

[12] Bailian Chen, Dylan R Harp, Youzuo Lin, Elizabeth H Keating, and Rajesh J Pawar. Geologic co2 sequestration monitoring design: A machine learning and uncertainty quantification based approach. *Applied energy*, 225:332–345, 2018.

[13] Hongrui Chen, Holden Lee, and Jianfeng Lu. Improved analysis of score-based generative modeling: User-friendly bounds under minimal smoothness assumptions. In *International Conference on Machine Learning*, pages 4735–4763. PMLR, 2023.

[14] Sitan Chen, Sinho Chewi, Holden Lee, Yuanzhi Li, Jianfeng Lu, and Adil Salim. The probability flow ode is provably fast. *Advances in Neural Information Processing Systems*, 36:68552–68575, 2023.

[15] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Yuan Chen and Dongbin Xiu. Learning stochastic dynamical system via flow map operator. *Journal of Computational Physics*, 508:112984, 2024.

[17] CMG. Cmg gem user's guide. *Calgary, Alberta, Canada:Computer Modeling Group Ltd.*, 2022.

[18] Diego Del-Castillo-Negrete, Minglei Yang, Matthew Beidler, and Guannan Zhang. Generation and mitigation of runaway electrons: spatio-temporal effects in dynamic scenarios. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2021.

[19] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

[20] Thierry Dombre, Uriel Frisch, John M Greene, Michel Hénon, A Mehr, and Andrew M Soward. Chaotic streamlines in the abc flows. *Journal of Fluid Mechanics*, 167:353–391, 1986.

[21] Geir Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, Berlin, Heidelberg, 2nd edition, 2009.

[22] Geir Evensen, Femke C. Vossepoel, and Peter Jan van Leeuwen. *Data Assimilation Fundamentals: A Unified Formulation of the EnKF and Particle Filter*. Springer, Cham, Switzerland, 2022.

[23] M. Fan, Z. Zhang, D. Lu, and G. Zhang. GenAI4UQ: A software for inverse uncertainty quantification using conditional generative ai models. *SoftwareX*, 31:102232, 2025.

[24] Ming Fan, Dan Lu, and Siyan Liu. A deep learning-based direct forecasting of co2 plume migration. *Geoenergy Science and Engineering*, 221:211363, 2023.

[25] Ming Fan, Hongsheng Wang, Jing Zhang, Seyyed A Hosseini, and Dan Lu. Advancing spatiotemporal forecasts of co2 plume migration using deep learning networks with transfer learning and interpretation analysis. *International Journal of Greenhouse Gas Control*, 132:104061, 2024.

[26] Li Feng, Xiaodong Zeng and Tao Zhou. Solving time dependent fokker-planck equations via temporal normalizing flow. *Communications in Computational Physics*, 32(2):401–423, 2022.

[27] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[28] Khashayar Gatmiry, Jonathan Kelner, and Holden Lee. Learning mixtures of gaussians using diffusion models. In *The Thirty Eighth Annual Conference on Learning Theory*, pages 2403–2456. PMLR, 2025.

[29] Raoof Gholami, Arshad Raza, and Stefan Iglauer. Leakage risk assessment of a co2 storage site: A review. *Earth-Science Reviews*, 223:103849, 2021.

[30] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[31] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

[32] Ling Guo, Hao Wu, and Tao Zhou. Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models. *Journal of Computational Physics*, 461:111202, 2022.

[33] Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[34] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(40):1303–1347, 2013.

[35] Daniel Zhengyu Huang, Jiaoyang Huang, and Zhengjiang Lin. Convergence analysis of probability flow ode for score-based generative models. *IEEE Transactions on Information Theory*, 2025.

[36] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[37] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[38] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*. Springer, 2014.

[39] Sharmila Karumuri and Ilias Bilionis. Learning to solve bayesian inverse problems: An amortized variational inference approach using gaussian and flow guides. *Journal of Computational Physics*, 511:113117, 2024.

[40] Rachael T Keller and Qiang Du. Discovery of dynamics using linear multistep methods. *SIAM Journal on Numerical Analysis*, 59(1):429–455, 2021.

[41] Thomas A Keller, Jorn WT Peters, Priyank Jaini, Emiel Hoogeboom, Patrick Forré, and Max Welling. Self normalizing flows. In *International Conference on Machine Learning*, pages 5378–5387. PMLR, 2021.

[42] Pierric Kersaudy, Bruno Sudret, Nadège Varsier, Odile Picon, and Joe Wiart. A new surrogate modeling technique combining kriging and polynomial chaos expansions–application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics*, 286:103–117, 2015.

[43] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[44] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*, 2014.

[45] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[46] Gen Li, Yu Huang, Timofey Efimov, Yuting Wei, Yuejie Chi, and Yuxin Chen. Accelerating convergence of score-based diffusion models, provably. In *Forty-first International Conference on Machine Learning*, 2024.

[47] Y. Liu, Y. Chen, D. Xiu, and G. Zhang. A training-free conditional diffusion model for learning stochastic dynamical systems. *SIAM Journal on Scientific Computing*, 47(5):C1144–C1171, 2025.

[48] Yanfang Liu, Minglei Yang, Zezhong Zhang, Feng Bao, Yanzhao Cao, and Guannan Zhang. Diffusion-model-assisted supervised learning of generative models for density estimation. *Journal of Machine Learning for Modeling and Computing*, 5(1):25–38, 2024.

[49] D. Lu, Y. Liu, Z. Zhang, F. Bao, and G. Zhang. A diffusion-based uncertainty quantification method to advance E3SM model calibration. *Journal of Geophysical Research: Machine Learning and Computation*, 1:e2024JH000234, 2024.

[50] Yubin Lu, Romit Maulik, Ting Gao, Felix Dietrich, Ioannis G Kevrekidis, and Jinqiao Duan. Learning the temporal evolution of multivariate densities via normalizing flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3):033121, 2022.

[51] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[52] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[53] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[54] Yu V Petrov, PB Parks, and RW Harvey. Numerical simulation of the hot-tail runaway electron production mechanism using cql3d and comparison with smith–verwichte analytical model. *Plasma Physics and Controlled Fusion*, 63(3):035026, 2021.

[55] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[56] Apostolos F Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023.

[57] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538. PMLR, 2015.

[58] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

[59] Christian P Robert and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 2004.

[60] Raghavendra Selvan, Frederik Faye, Jon Middleton, and Akshay Pai. Uncertainty quantification in medical image segmentation with normalizing flows. In *International workshop on machine learning in medical imaging*, pages 80–90. Springer, 2020.

[61] Murat Sensoy, Lance Kaplan, Federico Cerutti, and Maryam Saleki. Uncertainty-aware deep classifiers using generative models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5620–5627, 2020.

[62] Kulin Shah, Sitan Chen, and Adam Klivans. Learning mixtures of gaussians using the ddpm objective. *Advances in Neural Information Processing Systems*, 36:19636–19649, 2023.

[63] Michael F Shlesinger, George M Zaslavsky, and Joseph Klafter. Strange kinetics. *Nature*, 363(6424):31–37, 1993.

[64] Ronald W Shonkwiler and Franklin Mendivil. *Explorations in monte carlo methods*. Springer Science & Business Media, 2009.

[65] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*. SIAM, 2024.

[66] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[67] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[68] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of the International Conference on Learning Representations*, 2021.

[69] A Stahl, O Embréus, G Papp, M Landreman, and T Fülöp. Kinetic modelling of runaway electrons in dynamic scenarios. *Nuclear Fusion*, 56(11):112009, 2016.

[70] Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[71] Yuankai Teng, Zhu Wang, Lili Ju, Anthony Gruber, and Guannan Zhang. Level set learning with pseudoreversible neural networks for nonlinear dimension reduction in function approximation. *SIAM Journal on Scientific Computing*, 45(3):A1148–A1171, 2023.

[72] Pengjun Wang, Zezhong Zhang, Minglei Yang, Feng Bao, Yanzhao Cao, and Guannan Zhang. Error estimates of a training-free diffusion model for high-dimensional sampling. *arXiv preprint arXiv:2601.19740*, 2026.

[73] Yuqing Wang, Ye He, and Molei Tao. Evaluating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 37:19307–19352, 2024.

[74] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

[75] D Wirtz, N Karajan, and B Haasdonk. Surrogate modeling of multiscale models using kernel methods. *International Journal for Numerical Methods in Engineering*, 101(1):1–28, 2015.

[76] M. Yang, Y. Liu, D. del Castillo-Negrete, Y. Cao, and G. Zhang. Generative AI models for learning flow maps of stochastic dynamical systems in bounded domains. *Journal of Computational Physics*, 544:114434, 2026.

[77] Minglei Yang, Pengjun Wang, Diego del Castillo-Negrete, Yanzhao Cao, and Guannan Zhang. A pseudoreversible normalizing flow for stochastic dynamical systems with various initial distributions. *SIAM Journal on Scientific Computing*, 46(4):C508–C533, 2024.

[78] Minglei Yang, Pengjun Wang, Ming Fan, Dan Lu, Yanzhao Cao, and Guannan Zhang. Conditional pseudo-reversible normalizing flow for surrogate modeling in quantifying uncertainty propagation. *Journal of Machine Learning for Modeling and Computing*, 6(4), 2025.

[79] Minglei Yang, Guannan Zhang, Diego del Castillo-Negrete, and Miroslav Stoyanov. A feynman-kac based numerical method for the exit time probability of a class of transport problems. *Journal of Computational Physics*, 444:110564, 2021.

[80] Zongzhao Zhou, Yew Soon Ong, My Hanh Nguyen, and Dudy Lim. A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In *2005 IEEE congress on evolutionary computation*, volume 3, pages 2832–2839. IEEE, 2005.